

Un enfoque práctico para engañar la detección remota de SO de Nmap

David Barroso Berrueta

<http://voodoo.somoslopeor.com>

tomac@somoslopeor.com

Copyright © 2003 David Barroso Berrueta. Palencia (Spain)

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

La detección remota de SO es está haciendo cada vez más popular, no sólo para los auditores de redes, sino también para cualquier atacante. Ya que Nmap es cada vez más popular como la herramienta utilizada para adivinar qué SO está corriendo en un sistema remota, están surgiendo algunas herramientas de seguridad para engañar a Nmap en su propósito de la detección de SO. Este artículo describe diferentes métodos para engañar a Nmap y comportarse como cualquier otro sistema operativo elegido, así como demostraciones de cómo se realiza.

1. Introducción

El propósito de este artículo es intentar enumerar y describir brevemente todas las aplicaciones y técnicas desarrolladas para engañar la detección remota de SO de Nmap, pero en cualquier caso, la seguridad por oscuridad no es ninguna buena solución; puede ser una buena medida de seguridad, pero por favor, ten en cuenta que es más importante tener un entorno verdaderamente seguro (parches, firewalls, ids, ...) que intentar esconder tu SO.

Saber que Sistema Operativo está corriendo en un sistema remoto puede ser muy valioso tanto para los auditores de redes como para cualquier atacante. Supón que encuentran un puerto abierto en su (aprobada o no) penetración; saber el SO hace que encontrar y ejecutar un *exploit* contra ese servicio sea más fácil, ya que a menudo los *exploit*son para versiones específicas de SO, y un *exploit* para Sendmail sobre HP-UX no funcionará para Sendmail sobre AIX, o para ser más exactos, un *exploit* para AIX 4.3.3 puede no funcionar en un sistema corriendo 4.3.3 con los últimos parches de mantenimiento aplicados. Fyodor (autor de Nmap) ha escrito un detallado artículo (<http://www.insecure.org/nmap/nmap-fingerprinting-article.html>) sobre la detección remota de SO, describiendo algunos métodos diferentes de detectar con éxito el SO remoto, desde los métodos más básicos, hasta los más complejos.

Al principio, adivinar el SO remoto se hacía mirando el *banner* de un servicio específico. Por ejemplo, un típico *banner* de telnet o ftp siempre era mostrado a todo el mundo, diciendo que SO estaba siendo ejecutado; o si el *banner* había sido quitado o cambiado, se podían ejecutar algunos comandos para conocer el SO (recuerda el SYST en el FTP). Otros métodos básicos para conocer el SO pueden ser buscar entradas HINFO en el servidor DNS, o intentar conseguir información usando snmp (un montón de dispositivos tienen habilitado por defecto el acceso snmp usando la cadena 'public'). Incluso buscar anuncios de trabajo publicados en Internet, buscar en su basura manuales de SO, o hacer uso de la ingeniería social son métodos válidos de intentar averiguar el SO remoto.

Después, se desarrollaron algunas soluciones más avanzadas, aprovechándose de que cada fabricante de SO tiene una pila TCP/IP distinta. La idea es mandar algunos paquetes 'creados' al sistema remoto y esperar su respuesta. Esos paquetes son paquetes 'malvados', contruidos con opciones TCP poco comunes u opciones 'imposibles'. Cada SO tiene su propia pila TCP/IP, no existe una implementación común para todos los SO, y este hecho permite crear una clasificación de los SO y versiones de acuerdo a sus respuestas. De esta forma es como funcionan las herramientas de detección remota de SO; algunas de ellas usando el protocolo TCP/IP, y otras usando el protocolo ICMP.

Hay un artículo sobre 'Defeating TCP/IP Stack Fingerprinting (<http://www.usenix.org/publications/library/proceedings/sec2000/smart.html>)' que describe en un modo teórico el diseño y la implementación de un programa de recogida de implementaciones de pilas TCP/IP. Ese artículo describe cómo y porqué puedes vencer a la detección de SO, por lo que no voy hablar mucho sobre ello; por lo tanto, me centraré en la soluciones que podemos usar y están disponibles.

2. Razones para ocultar tu SO al mundo

Quizás te estas preguntando porqué quieres perder tu valioso tiempo cambiando tu kernel de Linux para ocultar tu verdadero SO a los usuarios 'malvados' de Nmap. Puede que las siguientes razones te convezcan:

- Al revelar tu SO hace que encontrar y ejecutar con éxito un *exploit* contra cualquiera de tus dispositivos sea más fácil.
- Tener un SO no parcheado o una versión antigua no es muy conveniente para el prestigio de tu compañía. Imagina que tu compañía es un banco y algunos usuarios se dan cuenta de que tienes varias máquinas sin parchear. No creo que confíen más en ti. Además, este tipo de 'malas' noticias siempre salen a la luz pública.
- El conocer el SO que tienes puede llegar a ser más peligroso, porque la gente puede adivinar que aplicaciones estás ejecutando en ese SO (inferencia de datos). Por ejemplo, si tu sistema es MS Windows, y estás ejecutando un base de datos, es muy probable que estés ejecutando MS-SQL.
- Puede llegar a ser conveniente para otras compañías de programas, ya que te pueden ofrecer un nuevo entorno de SO (ya que saben lo que tienes).
- Y finalmente, privacidad; nadie necesita saber los sistemas que estás ejecutando.

3. Nmap

Nmap (<http://www.insecure.org>) es una de estas herramientas. Manda siete paquetes TCP/IP (llamados tests) y espera la respuesta. Los resultados se comparan en una base de datos de resultados conocidos (fichero de firmas de SO). Esta base de datos es un fichero de texto que contiene el resultado respondido (firma) de cada SO

conocido. Así, si la respuesta coincide con alguna de las entradas de la base de datos, podemos adivinar que el SO remoto es el mismo que el de la base de datos. Algunos paquetes Nmap se mandan a un puerto abierto y otros a un puerto cerrado; dependiendo de los resultados, se averigua el SO remoto. Una entrada de ejemplo puede ser:

```
/* Comentario sobre el SO. Sí, queremos ser una consola Sega Dreamcast */
Fingerprint Sega Dreamcast

/* Predicibilidad del ISN; TD: dependiente del tiempo */
TSeq(Class=TD%gcd=<780%SI=<14)

/* Resultado del Test 1: paquete SYN con algunas opciones a un puerto abierto. Obtenemos
un SYN+ACK, reconocimiento seq +1, tamaño de ventana 0x1d4c, bit no fragmentar
no activado, y sólo el MSS devuelto */
T1(DF=N%W=1D4C%ACK=S++%Flags=AS%Ops=M)

/* Resultado del Test 2: paquete Null con
algunas opciones a un puerto abierto. Obtenemos un ACK+RST, reconocimiento seq,
tamaño de ventana 0x0, bit no fragmentar no activado */
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)

/* Resultado del Test 3: SYN, FIN, URG, PSH con opciones a un puerto abierto.
Obtenemos un SYN+ACK, reconocimiento seq +1, tamaño de ventana 0x1d4c, el
bit no fragmentar no activado, y sólo devuelve el MSS */
T3(Resp=Y%DF=N%W=1D4C%ACK=S++%Flags=AS%Ops=M)

/* Resultado del Test 4: paquete ACK a un puerto abierto. Obtenemos un RST,
reconocimiento seq, tamaño de ventana 0x0, bit de no fragmentar no activado */
T4(DF=N%W=0%ACK=S%Flags=R%Ops=)

/* Resultado del Test 5: paquete SYN con opciones a un puerto cerrado. Obtenemos
un ACK+RST, reconocimiento seq, tamaño de ventana 0x0, bit de no fragmentar no
activado */
T5(DF=N%W=0%ACK=S%Flags=AR%Ops=)

/* Resultado del Test 6: ACK con opciones a un puerto cerrado. Obtenemos un RST,
reconocimiento seq, tamaño de ventana 0x0, bit no fragmentar no activado */
T6(DF=N%W=0%ACK=S%Flags=R%Ops=)

/* Resultado del Test 7: FIN, PSH, URG con opciones a un puerto cerrado. Obtenemos
un ACK+RST, reconocimiento seq+1, tamaño de ventana 0x0, bit no fragmentar no
activado */
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)

/* Resultado de Port unreachable. No hay respuesta */
PU(Resp=N)
```

Así que, si queremos engañar a Nmap y decir al atacante que estamos corriendo un sistema operativo diferente, sólo necesitamos engañar las respuestas a los tests de Nmap. La solución que voy a describir es sólo válida para engañar Nmap pero no a todas las herramientas de detección remota de SO. En la Conclusión otras herramientas serán mencionadas, así como algunas recomendaciones para el auditor de sistemas y/o el atacante.

4. Soluciones de Linux

Los métodos para engañar la detección remota de SO de Nmap están escritos como módulos del kernel, o al menos, como parches al kernel de Linux. La razón es que si el objetivo es cambiar el comportamiento de la pila TCP/IP, necesitamos hacerlo en la capa del kernel.

Se van a describir tres módulos del kernel, todos ellos independientes del árbol del kernel de Linux; tienes que descargarlos y parchear tu kernel para añadirlos. El primero requiere que tengas *netfilter* activado en tu kernel (que pienso que es necesario si quieres empezar a tener un sistema seguro), pero los otros dos no lo necesitan.

4.1. IP Personality

El primero, y probablemente, mejor opción es IP Personality (<http://ippersonality.sourceforge.net/>). Es un módulo de *netfilter* (por lo tanto, sólo disponible para kernels 2.4) que permite cambiar el comportamiento de la pila IP, pudiendo tener múltiples personalidades dependiendo de los parámetros que le especifiques en una regla de **iptables**. De hecho, podemos cambiar las siguientes opciones:

- Número de Secuencia Inicial TCP (ISN)
- Tamaño inicial de ventana TCP
- Opciones TCP (sus tipos, valores y orden en el paquete)
- IP números de ID
- respuestas a algunos paquetes TCP 'anómalos'
- respuestas a algunos paquetes UDP

Un resumen general de *IP Personality* es que podemos cambiar la forma de responder a algunos paquetes, y podemos especificar qué paquetes queremos responder de esa forma (puede ser dependiendo de la dirección ip origen, el puerto destino, o, y es lo que vamos a usar, esos paquetes especiales que manda Nmap)

La instalación es bastante sencilla y bien explicada en el fichero `INSTALL` que viene en el paquete; para nuestros propósitos, nuestra máquina de pruebas es una *Debian stable* con un kernel 2.4.19. Por defecto, el módulo de *netfilter* de *IP Personality* no está disponible en los últimos kernels, así que tenemos que parchear el código fuente del kernel. El parche para añadir *IP Personality* a nuestras fuentes de *netfilter* está disponible en la página del programa. También necesitamos parchear nuestro comando **iptables** para que reconozca nuestra nueva opción. Una vez que el kernel ha sido parcheado y compilado, necesitamos reiniciar la máquina ya que el parche modifica otros ficheros de *netfilter* (*connection tracking*)

El siguiente paso es incluir nuestras reglas de **iptables** relacionadas con *IP Personality* en nuestro kernel. Antes de hacerlo, ejecutamos Nmap para ver que SO tenemos:

```
# nmap (V. 3.10ALPHA4) scan initiated Wed Feb 19 20:26:52 2003 as: nmap -sS -O -oN nmap1.log 1
Interesting ports on 192.168.0.19:
(The 1597 ports scanned but not shown below are in state: closed)
Port      State      Service
22/tcp    open      ssh
25/tcp    open      smtp
80/tcp    open      http
143/tcp   open      imap2
```

```
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 106.832 days (since Tue Nov 5 00:29:33 2002)
# Nmap run completed at Wed Feb 19 20:26:58 2003 -- 1 IP address (1 host up) scanned in 7.957
```

Ahora, podemos reiniciar en nuestro nuevo kernel parcheado, y añadir las reglas **iptables** para engañar a Nmap:

```
voodoo:~/ippersonality-20020819-2.4.19/samples#/usr/local/sbin/iptables -t mangle -A PREROUTING
voodoo:~/ippersonality-20020819-2.4.19/samples#/usr/local/sbin/iptables -t mangle -A OUTPUT -s
```

Lo que estamos haciendo con estas reglas es:

- La primera significa que todos los paquetes que vengan de 192.168.0.50 (yo) contra 192.168.0.19 (servidor) tienen que ser modificados y reescritos para simular una Dreamcast. La cadena de *PREROUTING* es la que lo tiene que hacer.
- La segunda significa que todos los paquetes que vengan de 192.168.0.19 (servidor) contra 192.168.0.50 (yo) tienen que ser modificados y reescritos para simular una Dreamcast. Como son paquetes que salen del servidor, tenemos que usar la cadena *OUTPUT*

Comprobamos nuestra configuración:

```
voodoo:~/ippersonality-20020819-2.4.19/samples#/usr/local/sbin/iptables -L -t mangle
Chain PREROUTING (policy ACCEPT)
target     prot opt source                destination
PERS       all  192.168.0.50          192.168.0.19    tweak:dst local id:Dreamcast
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
Chain FORWARD (policy ACCEPT)
target     prot opt source                destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
PERS       all  192.168.0.19          192.168.0.50    tweak:src local id:Dreamcast
Chain POSTROUTING (policy ACCEPT)
target     prot opt source                destination
```

Ahora podemos ver si Nmap todavía nos dice que tenemos Linux kernel 2.4.0-2.5.20 o quizás averiguamos que nuestro SO ha cambiado:

```
# nmap (V. 3.10ALPHA4) scan initiated Wed Feb 19 21:49:18 2003 as: nmap -sS -O -oN nmap2.log 1
Interesting ports on 192.168.0.19:
(The 1597 ports scanned but not shown below are in state: closed)
Port      State      Service
22/tcp    open      ssh
25/tcp    open      smtp
80/tcp    open      http
143/tcp   open      imap2
Remote operating system guess: Sega Dreamcast
# Nmap run completed at Wed Feb 19 21:49:23 2003 -- 1 IP address (1 host up) scanned in 5.886
```

Como puedes ver, hemos engañado a Nmap con nuestra respuesta. Es fácil elegir qué SO queremos 'ejecutar' en el fichero de firmas de SO de Nmap y decir a *IP Personality* que se comporte como el SO elegido. Vamos a ver el fichero `dreamcast.conf` que hemos especificado al añadir nuestras reglas de **iptables**:

```

/* Nuestra nueva identificación de SO */
id "Dreamcast";

/* sólo los paquetes que entran serán cambiados pero el tamaño de ventana TCP no será cambiado */
tcp {
    incoming yes;
    outgoing no;
    max-window 32768;
}

/* Necesitamos emular el generador de ISN de la Dreamcast que es dependiente del tiempo; esto
podemos hacer usando el generador fixed-inc y un pequeño incremento */
tcp_isn {
    type fixed-inc 2;
    initial-value random;
}

tcp_options {
    keep-unknown yes;
    keep-unused no;
    isolated-packets yes;
    code { copy(mss); }
}

/* ahora podemos seguir la firma de la Dreamcast y comportarnos como ella */
tcp_decoy {
    code {
        if (option(mss)) { /* nmap tiene mss en todos sus paquetes */
            set(df, 0);
            if (listen) {
                if (flags(syn&ece)) { /* nmap test 1 */
                    set(win, 0x1D4C);
                    set(ack, this + 1);
                    set(flags, ack|syn);
                    insert(mss, this+1);
                    reply;
                }
                if (flags(null)) { /* nmap test 2 */
                    set(win, 0);
                    set(ack, this);
                    set(flags, ack|rst);
                    reply;
                }
                if (flags(syn&fin&urg&push)) { /* nmap test 3 */
                    set(win, 0x1D4C);
                    set(ack, this + 1);
                    set(flags, ack|syn);
                    insert(mss, this+1);
                    reply;
                }
                if (ack(0) && flags(ack) && !flags(syn|push|urg|rst)) { /* nmap test 4 */
                    set(win, 0);
                    set(ack, this);
                }
            }
        }
    }
}

```

```

        set(flags, rst);
        reply;
    }
} else {
    set(win, 0);
    if (flags(syn) && !flags(ack)) { /* nmap test 5 */
        set(ack, this);
    }
    set(flags, ack|rst);
    reply;
}
if (ack(0) && flags(ack) && !flags(syn|push|urg|rst)) { /* nmap test 6 */
    set(ack, this);
    set(flags, rst);
    reply;
}
if (flags(fin&push&urg)) { /* nmap test 7 */
    set(ack, this + 1);
    set(flags, ack|rst);
    reply;
}
}
}
}

/* No respondemos con ICMP a conexiones a puertos UDP cerrados */
udp_unreach {
    reply no;
    df no;
    max-len 56;
    tos 0;

    mangle-original {
        ip-len 32;
        ip-id same;
        ip-csum zero;
        udp-len 308;
        udp-csum same;
        udp-data same;
    }
}

```

IP Personality es aún más poderoso. Puedes configurar un firewall/router Linux que cambiará la respuesta de las máquinas detrás de él. Todos las máquinas que están detrás de tu router Linux pueden parecer consolas Sega Dreamcast para tus atacantes!.

También hay un interesante parche para Nmap en el mismo sitio, llamado *osdet* (<http://ippersonality.sourceforge.net/download.html>), que nos permite hacer una detección remota de SO usando el motor de Nmap, pero con la interesante característica de que vemos los paquetes que mandamos y recibimos en formato de tcpdump. Algunas veces viene muy bien y nos ayuda a comprender la técnica de detección remota de SO el ver los paquetes corriendo por tu pantalla (todos los tests de Nmap y sus respuestas).

4.2. Parche Stealth

La solución que se va a describir ahora es el parche *stealth*, disponible en Security Technologies (<http://www.innu.org/%7Esean/>). Está disponible como módulo del kernel para kernels de Linux 2.2.x (y en un futuro cercano para 2.4.x), y como un parche del kernel (para kernels de Linux 2.4.x sin soporte de módulos). Para comprobar su funcionamiento, usamos el parche para el kernel 2.4.19; una vez parcheado, aparecen dos nuevas opciones en nuestro fichero de configuración:

- *IP: TCP Stack Options*: opción que tenemos que seleccionar si queremos usar el parche *stealth*. Si seleccionas esta opción, se activa por defecto cuando arrancas el sistema. Para deshabilitarlo, necesitas ejecutar:

```
echo 0 > /proc/sys/net/ipv4/tcp_ignore_ack
echo 0 > /proc/sys/net/ipv4/tcp_ignore_bogus
echo 0 > /proc/sys/net/ipv4/tcp_ignore_synfin
```

- *Log all dropped packets*: guarda todos los paquetes con opciones extrañas.

Este parche simplemente ignora los paquetes TCP/IP recibido que cumplan las siguientes características:

1. Paquetes con SYN y FIN activado (*tcp_ignore_synfin*) (test de QueSO).
2. Paquetes *Bogus*: si la cabecera TCP tiene el bit res1 activo (uno de los bits reservados, entonces es un paquete *bogus*) o si no tiene nada de lo siguiente activado: ACK, SYN, RST o FIN (test 2 Nmap).
3. Paquetes con FIN, PUSH y URG activado (test 7 Nmap).

Esta es una solución más simple que la que hemos descrito antes. No podemos comportarnos como cualquier otro Sistema Operativo, ya que simplemente ignoramos todos los paquetes 'extraños' que se supone que están dirigidos a adivinar nuestro SO, y esperamos que sea suficiente para engañar a nuestro atacante, o al menos, poner las cosas más difíciles. Estas modificaciones al kernel son fáciles de comprender, y es relativamente fácil añadir nuestras fórmulas caseras para la detección de 'paquetes malvados'.

4.3. Fingerprint Fucker

Fingerprint Fucker es un módulo del kernel disponible para kernels 2.2.x que puede también esconder tu SO y comportarse como otro. Es un módulo que acepta parámetros desde la línea de comandos para configurar su respuesta. Por defecto, simula un VAX. También contiene un fichero, llamado `finger_parsers.c`, que recorre un fichero de firmas de Nmap y carga el módulo de *Fingerprint Fucker* con los parámetros correctos (al ejecutar *finger_parsers*, tienes que especificar que SO quieres emular). Entonces espera a recibir los paquetes de Nmap, y responde cómo le hayas configurado. Hasta donde he podido comprobar, sólo algunos tests de Nmap son tratados (T1, T2 y T7).

Atención

El código no es muy estable. Momentos después de cargar el módulo mi máquina Linux se colgó.

4.4. IPlog

IPlog (<http://ojnk.sourceforge.net/stuff/iplog.readme>) es un *logger* de TCP/IP que también detecta algunos >*scans* (XMAS, FIN, SYN, ...). Para nuestros propósitos, tiene una opción (-z) que permite engañar a los intentos de Nmap, y, aun cuando no podemos comportarnos como otro SO, podemos engañar totalmente a Nmap en su intento de adivinar remotamente nuestro SO.

Al ejecutar *iplog*, observamos los resultados:

```
voodoo:~#iplog -o -L -z -i eth0
```

Las opciones son las siguientes: -o (quédate en primer plano), -L (resultados a la salida estándar), -z (engaña a Nmap), -i eth0 (escucha en eth0). Si ejecuto un Nmap contra la máquina, *iplog* empieza a escribir un montón de información a la salida estándar, sobre todas las conexiones establecidas, e incluso sobre qué tipo de *scan* está siendo llevado a cabo; he incluido sólo la información relevante sobre la identificación de SO de Nmap en la salida de *iplog*

```
Feb 20 13:20:54 TCP: SYN scan detected [ports 10082,1430,770,815,440,86,848,797,560,5998,
Feb 20 13:20:56 TCP: Bogus TCP flags set by 192.168.0.50:49054 (dest port 22)
Feb 20 13:20:56 UDP: dgram to port 1 from 192.168.0.50:49047 (300 data bytes)
Feb 20 13:20:56 ICMP: 192.168.0.50: port is unreachable to (udp: dest port 1, source port 49047)
Feb 20 13:20:58 UDP: dgram to port 1 from 192.168.0.50:49047 (300 data bytes)
Feb 20 13:20:58 ICMP: 192.168.0.50: port is unreachable to (udp: dest port 1, source port 49047)
Feb 20 13:21:01 UDP: dgram to port 1 from 192.168.0.50:49047 (300 data bytes)
Feb 20 13:21:01 ICMP: 192.168.0.50: port is unreachable to (udp: dest port 1, source port 49047)
Feb 20 13:21:04 TCP: Xmas scan detected [ports 1,9,49055,49056,49054] from 192.168.0.50 [ports 49055,49056,49054]
Feb 20 13:21:05 UDP: dgram to port 1 from 192.168.0.50:49047 (300 data bytes)
Feb 20 13:21:05 ICMP: 192.168.0.50: port is unreachable to (udp: dest port 1, source port 49047)
Feb 20 13:21:12 TCP: null scan detected [ports 9,49056,49060,49054] from 192.168.0.50 [ports 49055,49056,49054]
Feb 20 13:21:13 TCP: FIN scan detected [ports 49060,49054,9,1] from 192.168.0.50 [ports 1,9,49055,49056,49054]
Feb 20 13:21:56 TCP: SYN scan mode expired for 192.168.0.50 - received a total of 1647 packets (300 data bytes)
Feb 20 13:21:56 TCP: Xmas scan mode expired for 192.168.0.50 - received a total of 33812 packets (300 data bytes)
Feb 20 13:22:03 TCP: null scan mode expired for 192.168.0.50 - received a total of 16462 packets (300 data bytes)
Feb 20 13:22:04 TCP: FIN scan mode expired for 192.168.0.50 - received a total of 16343 packets (300 data bytes)
```

Iplog reconoce los paquetes TCP *bogus*, los paquetes TCP 'vacíos', todos los intentos de reconocimiento remoto de Nmap. Esa es la razón por la que puede actuar consecuentemente y mandar una respuesta errónea para engañar a Nmap. La salida de Nmap es la siguiente:

```
# nmap (V. 3.10ALPHA4) scan initiated Thu Feb 20 13:20:54 2003 as: nmap -vv -sS -O -oN nmap
Insufficient responses for TCP sequencing (1), OS detection may be less accurate
Insufficient responses for TCP sequencing (1), OS detection may be less accurate
Insufficient responses for TCP sequencing (1), OS detection may be less accurate
Interesting ports on voodoo (127.0.0.1):
(The 1599 ports scanned but not shown below are in state: closed)
Port      State      Service
22/tcp    open      ssh
25/tcp    open      smtp
80/tcp    open      http
143/tcp   open      imap2
No exact OS matches for host (If you know what OS is running on it, see http://www.insecure.org/c
```

```

TCP/IP fingerprint:
SInfo(V=3.10ALPHA4%P=i586-pc-linux-gnu%D=2/20%Time=3E54C833%O=9%C=1)
T1(Resp=Y%DF=Y%W=7FFF%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=Y%DF=N%W=0%ACK=0%Flags=BA%Ops=)
T2(Resp=Y%DF=Y%W=100%ACK=0%Flags=BARF%Ops=)
T2(Resp=Y%DF=Y%W=100%ACK=0%Flags=BPF%Ops=)
T3(Resp=Y%DF=N%W=0%ACK=0%Flags=BA%Ops=)
T4(Resp=Y%DF=Y%W=0%ACK=0%Flags=R%Ops=)
T5(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(Resp=Y%DF=Y%W=0%ACK=0%Flags=R%Ops=)
T7(Resp=Y%DF=N%W=0%ACK=0%Flags=BA%Ops=)
T7(Resp=Y%DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(Resp=Y%DF=N%TOS=C0%IPLen=164%RIPTL=148%RID=E%RIPCK=E%UCK=E%ULEN=134%DAT=E)

# Nmap run completed at Thu Feb 20 13:21:07 2003 -- 1 IP address (1 host up) scanned in 13.633 se

```

Como podéis ver, *iplog* responde a todos los paquetes que envía Nmap; veamos el código fuente de *iplog*

file *iplog_tcp.c*, line 99:

```

if (opt_enabled(FOOL_NMAP) &&
    ((tcp_flags & TH_BOG) || (tcp_flags == TH_PUSH) || (tcp_flags == 0) ||
    ((tcp_flags & (TH_SYN | TH_FIN | TH_RST)) && (tcp_flags & TH_URG)) ||
    ((tcp_flags & TH_SYN) && (tcp_flags & (TH_FIN | TH_RST)))))

```

Esa sentencia 'if' significa que si hemos ejecutado *iplog* con la opción '-z' (engañar Nmap), y si las opciones de la cabecera TCP son:

- *bogus* (uso de los bits reservados), o
- sólo PUSH, o
- NULL (sin opciones), o
- SYN+URG, FIN+URG, RST+URG, o
- SYN+FIN, SYN+RST

entonces creará un nuevo paquete para responder con las opciones que queramos (algunas opciones dependen en la hora de la máquina, por ejemplo DF, esta es la razón por la que algunas veces es 1 y otras 0, o el tamaño de ventana, que está definido como `current_time & 1`).

Naturalmente podemos cambiar el fichero *iplog_tcp.c* para que *iplog* siempre se comporte como una Sega Dreamcast contra esos paquetes malvados, pero no tenemos la flexibilidad de tener múltiples personalidades o de especificar que queremos comportarnos como una Dreamcast sólo para un tráfico específico o una dirección ip específica. Es una buena idea responder de esta manera a estos paquetes 'anormales', pero es mejor tener el control sobre ello, y que sea más granular.

5. Soluciones *BSD

5.1. Blackhole

Blackhole es una opción especial presente en el kernel *BSD para controlar el comportamiento del sistema cuando alguien se conecta a puertos TCP o UDP cerrados. Hay dos opciones que podemos cambiar:

```
sysctl -w net.inet.tcp.blackhole=[0 | 1 | 2]
sysctl -w net.inet.udp.blackhole=[0 | 1]
```

Blackhole TCP se comporta de la forma siguiente: si el valor es 0, cuando un paquete se conecta a un puerto TCP cerrado, se devuelve un RST. Si el valor es 1, si un paquete SYN se llega un puerto TCP cerrado, se ignora; y si el valor es 2, también se ignora.

Blackhole UDP es similar; si el valor es 0, cualquier conexión a un puerto UDP cerrado, devuelve un paquete *ICMP port unreachable*; si el valor es 1, entonces no lo devuelve.

Si activamos estas opciones, los tests 5, 6 y 7, y el test del *port unreachable* no funcionarán, por lo que no seremos capaces de conocer el SO.

5.2. Fingerprint Fucker

Existe también otro *Fingerprint fucker* (<http://packetstormsecurity.org/UNIX/misc/bsdffpf.tar.gz>) para los sistemas FreeBSD, escrito por Darren Reed, que simplemente reescribe la pila TCP/IP y manda paquetes con otras opciones (diferente tamaño de ventana, ttl, ...) intentando ocultar su verdadero SO.

5.3. OpenBSD packet filter

El *OpenBSD packet filter* puede ser también configurado para engañar la identificación remota de SO. Hay algunas opciones en el fichero de configuración `ip.conf` (Normalización del tráfico) (<http://www.openbsd.org/cgi-bin/man.cgi?query=pf.conf&sektion=5&arch=i386&apr>) donde puedes cambiar algunos campos del paquete IP (bit DF, TTL, MSS, ID), tal como se puede ver en la página de manual de `ip.conf`:

```
no-df
Pone a cero el bit de no fragmentar del paquete ip.

min-ttl _numero_
Asigna un ttl mínimo para el paquete ip.

max-mss _numero_
Asigna un máximo mss para el paquete ip.

random-id
Reemplaza el campo de identificación IP con valores aleatorios para
compensar los valores predecibles generados por muchas máquinas. Esta
```

opción sólo se aplica a los paquetes de salida que no son fragmentados después del opciones reensamblado de paquetes.

5.4. *FreeBSD TCP_DROP_SYNFIN*

El kernel de FreeBSD tiene una opción especial (http://www.freebsd.org/doc/en_US.ISO8859-1/articles/dialup-firewall/kernel.html), `TCP_DROP_SYNFIN`, que ignora todos los paquetes con SYN y FIN activados (el test #3 de Nmap manda un paquete TCP con SYN+FIN+PSH+URG) ; esta opción puede ser también un método válido para engañar a Nmap cuando realiza sus tests (acuérdate de activarlo en el arranque en el fichero `/etc/rc.conf`).

6. Soluciones generales

Hemos visto al hablar de *IP Personality* que podemos configurar un router linux protegiendo a nuestra red interna, y ese router podría engañar a Nmap y otras herramientas de detección de SO en sus intentos de adivinar remotamente el SO de nuestras máquinas de la red interna. Si no tenemos una máquina linux, pero tenemos un *Checkpoint FW-1* , entonces podemos hacer algo similar gracias al lenguaje *INSPECT*. Usando este lenguaje, es fácil crear nuestro propio 'inspector de paquetes' para los paquetes que atraviesan nuestro fw-1. Existe una referencia (<http://www.phoneboy.com/fom-serve/cache/82.html>) en la lista de correo de FW-1 describiendo un servicio fw-1 para tratar estos paquetes *bogus*.

7. Más cosas para jugar

La siguiente solución no nos permite ocultar o cambiar nuestro SO, pero seremos capaces de crear tantos dispositivos virtuales como queramos con cualquier Sistema Operativo que podamos imaginar. Esta idea está siendo aplicada en el tema de los *honeypots*, simplemente porque puedes crear una clase C entera virtual con un montón de diferentes SO ejecutándose; el atacante puede ser fácilmente atraído por todas esas máquinas corriendo tantos servicios vulnerables...Puede ser la panacea que busca todo atacante.

Los *honeypots* en general, y esta aproximación en particular, es altamente recomendable no sólo para aprender las herramientas y tácticas de los atacantes, sino también para redirigir a los atacantes contra tu *honeynet* y no contra tus máquinas en producción. Puede también hacer creer a a los atacantes que tienes muchas máquinas de un SO específico (el virtual) y ocultar tu SO real.

El programa que voy a describir brevemente es *honeyd* (<http://www.citi.umich.edu/u/provos/honeyd/>), de Niels Provos. Una de sus grandes características es que podemos asignar a cada una de nuestros dispositivos virtuales el SO que queramos. Esa personalidad también se especifica con un fichero normal de firmas de SO de Nmap, permitiéndonos convertirnos en el SO que queramos. No voy a describir con detalles esta gran herramienta, simplemente voy a ejecutar la configuración que viene como ejemplo para demostrar de lo que es capaz.

Después de instalarlo, hay un fichero que se llama `config.localhost` con un montón de dispositivos configurados. Por ejemplo, si vemos la definición del dispositivo 10.0.0.1:

```

    route entry 10.0.0.1
route 10.0.0.1 link 10.0.0.0/24
[snip]
create routerone
set routerone personality "Cisco 7206 running IOS 11.1(24)"
set routerone default tcp action reset
add routerone tcp port 23 "router-telnet.pl"
[snip]
bind 10.0.0.1 routerone
[snip]

```

La explicación a grandes rasgos es que tenemos un dispositivo cuya dirección ip es 10.0.0.1, que se comportará como un Cisco 7206 ejecutando una IOS 11.1(24), reseteará todas las conexiones TCP menos las que vayan al puerto 23, ya que entonces el script `router-telnet.pl` (una emulación del demonio telnet) será ejecutado. Bien, ejecutemos Nmap para comprobar el SO que se ejecuta en el dispositivo virtual que acabamos de crear:

```

# nmap (V. 3.10ALPHA4) scan initiated Thu Feb 20 16:17:44 2003 as: nmap -v -sS -oN nmap4.
Warning: OS detection will be MUCH less reliable because we did not find at least 1 open and 1 c
Interesting ports on 10.0.0.1:
(The 1604 ports scanned but not shown below are in state: filtered)
Port      State      Service
23/tcp    open       telnet
Remote OS guesses: Cisco 7206 running IOS 11.1(24), Cisco 7206 (IOS 11.1(17))
TCP Sequence Prediction: Class=random positive increments
Difficulty=26314 (Worthy challenge)
IPID Sequence Generation: Incremental

# Nmap run completed at Thu Feb 20 16:20:42 2003 -- 1 IP address (1 host up) scanned in 178.847 s

```

De nuevo, cuando recibimos los paquetes *bogus* de Nmap, *honeyd* responde con la personalidad que hemos escogido.

8. Conclusión

Tal como se dice en IP Personality Limitations (<http://ippersonality.sourceforge.net/doc/ippersonality-en-2.html>), el cambiar nuestra pila TCP/IP nos puede crear algunos problemas:

- algunas características de los SO están relacionadas con la arquitectura de la máquina (por ejemplo el tamaño de página en varias CPU), lo que podría conllevar problemas de eficiencia.
- algunos de estos cambios son cambios 'políticos' de la pila IP (números iniciales de secuencia, tamaño de ventana, opciones TCP disponibles, ...). Modificarlos permite engañar a un *scanner* pero puede romper nuestra conectividad en la red. Puede incluso hacer que el sistema sea menos seguro, al cambiar nuestra pila IP por otra menos segura.

En mi opinión, está bastante claro que no podemos fiarnos sólo de una sola herramienta de seguridad para adivinar el Sistema Operativo. Este artículo ha mostrado que es muy fácil engañar a Nmap (y otras herramientas similares) cuando intentan detectar un SO, y que todos esos intentos pueden ser convenientemente guardados por el administrador remoto. Para detectar con éxito el SO remoto, se tienen que ejecutar todos los posibles métodos,

empezando desde los más simples (captura de *banner*, buscar anuncios de trabajo, ingeniería social, ...) hasta los más complejos (identificación por red). Cada servicio abierto en un dispositivo remoto tiene que ser debidamente analizado (*banner*, respuestas, comportamiento ante ataques, DoS, errores conocidos) y documentado. Puede que sea incluso posible (aunque no ético) ejecutar algunas herramientas que se sabe que 'cuelgan' versiones específicas de SO (nuke, land, teardrop, ...) para clarificar nuestra suposición.

Aun cuando todas estas soluciones descritas pueden ser modificadas para detectar y engañar cualquier otra herramienta de detección de SO basada en TCP/IP (simplemente sabiendo qué paquetes manda), es bastante recomendable usar varias herramientas al intentar averiguar un SO remoto. Nmap es quizás la más usada, pero también existe otra herramienta que funciona bien: Xprobe (<http://www.sys-security.com/html/projects/X.html>). Xprobe tiene también una base de datos de firmas (no actualizada muy a menudo), y la suposición final es una suposición probabilística (*fuzzy matching*) dependiendo de varias respuestas. Uno de los mayores problemas de Xprobe es que no se actualiza muy a menudo, e incluye muy pocas firmas. Nmap detecta el SO remoto si el resultado de sus tests es igual a la firma de ese SO en la base de datos, pero también puedes ejecutar Nmap con la opción `--osscan_guess o --fuzzy`, y entonces realiza una búsqueda de SO más agresiva encontrando la firma que más se parece en su base de datos de firmas. Existe un artículo (<http://www.sys-security.com/archive/papers/Xprobe2.pdf>) sobre la especificación y uso de Xprobe donde se explica porqué su idea e implementación parece ser tan buena y válida. Pienso que se debe ejecutar conjuntamente con Nmap, en caso de que puedas mandar tanto paquetes TCP como ICMP. Xprobe puede ser una herramienta muy efectiva en redes no muy seguras, ya que manda paquetes *ICMP timestamps request* y *ICMP netmask request*, lo cual puede resultar muy sospechoso para un administrador de red. No envía paquetes *bogus* (paquetes TCP no muy comunes, ya que los bits reservados raramente son utilizados) para detectar el SO remoto, sino que simplemente manda tráfico ICMP 'normal' contra la máquina de destino, haciendo más difícil (si no imposible) detectar esos paquetes (y de esta forma, actuar consecuentemente). Este método fue utilizado primero en sing (<http://sing.sourceforge.net>) (*Send Internet Nasty Garbage*), que puede ser ejecutado con la opción '-O' para realizar la detección de SO (con el tipo de ICMP que elijas). Es difícil para cualquier IDS detectar que estos paquetes ICMP tienen una función extraña, ya que existe una gran cantidad de estos paquetes ICMP diariamente en nuestras redes. Por otro lado, ICMP cada vez se bloquea más por defecto en casi todos los entornos de red, haciendo imposible hacer una detección remota de SO basada en ICMP, pero normalmente en esos casos puedes encontrar algunos servicios TCP y disparar tus paquetes Nmap.

Para ser más exactos, existe otra herramienta de detección remota de SO, llamada p0f (<http://www.stearns.org/p0f/>); p0f escucha en tu red buscando por el primer SYN de una conexión TCP y guarda las opciones del paquete. Si concuerda con alguno de su base de datos de firma, entonces puede averiguar el SO: de nuevo, si cambiamos cualquiera de las opciones que busca p0f, podemos engañarlo. Si, por ejemplo, usamos *IP Personality*, podemos cambiar el tamaño de ventana de los paquetes, y haremos que p0f no sepa qué SO tenemos.

Los administradores deberían configurar con cuidado todos sus dispositivos para no mostrar ninguna información que pueda ser usada para identificarlos (*banners*, *issue*, servicios comunes abiertos por defecto, ...) y ejecutar alguna de estas herramientas que pueden guardar los intentos de detección remota de SO, ya que es muy probable que, esas direcciones ip que quieren saber tu SO, te estarán atacando tu red en breve tiempo. Además, configurar un router linux usando *IP Personality* y engañar a todo el mundo de fuera de tu red diciendo que tienes un SO diferente (con cualquiera de las opciones presentadas en este artículo), puede ser una buena medida de seguridad.

Referencias

Matthew Smart, Robert Malan, y Farnan Jahanian, *Defeating TCP/IP Stack Fingerprinting*, Usenix Security Symposium 2000, URL: <http://www.usenix.org/publications/library/proceedings/sec2000/smart.html> .

- Fyodor, *Remote OS Detection via TCP/IP Stack Fingerprinting*, June 11, 2002, URL: <http://www.insecure.org/nmap/nmap-fingerprinting-article.html> .
- Gael Roualland y Jean-Marc Saffroy, *IP Personality*, URL: <http://ippersonality.sourceforge.net/> .
- Sean Trifero y Derek Callaway, *Stealth*, URL: <http://www.innu.org/%7Esean/> .
- Ryan McCabe, *IPlog*, URL: <http://ojnk.sourceforge.net/stuff/iplog.readme> .
- Fusys y |CyRaX|, *Fingerprint Fucker*, URL: <http://www.s0ftpj.org/tools/fingfuck.tgz> .
- FreeBSD, *Blackhole*, URL: <http://www.gsp.com/cgi-bin/man.cgi?section=4&topic=blackhole> .
- Darren Reed, *Fingerprint Fucker*, URL: <http://packetstormsecurity.org/UNIX/misc/bsdspf.tar.gz> .
- OpenBSD, *ip.conf manual*, URL: <http://www.openbsd.org/cgi-bin/man.cgi?query=pf.conf&sektion=5&arch=i386&apr> .
- FreeBSD, *Kernel Options*, URL: http://www.freebsd.org/doc/en_US.ISO8859-1/articles/dialup-firewall/kernel.html .
- Alfredo Andrés Omella, *Trying to stop the security tool queSO*, URL: <http://www.phoneboy.com/fom-serve/cache/82.html> , October, 6th, 1998.
- Niels Provos, *Honeyd - Network Rhapsody for You*", URL: <http://www.citi.umich.edu/u/provos/honeyd/> .
- Gael Roualland y Jean-Marc Saffroy, *IP Personality Limitations*, URL: <http://ippersonality.sourceforge.net/doc/ippersonality-en-2.html> .
- Fyodor Yarochkin y Ofir Arkin, *Xprobe*, URL: <http://www.sys-security.com/html/projects/X.html> .
- Fyodor Yarochkin y Ofir Arkin, *Xprobe2 - A 'Fuzzy' Approach to Remote Active Operating System Fingerprinting*, URL: <http://www.sys-security.com/archive/papers/Xprobe2.pdf> .
- Alfredo Andrés Omella, *Sing*, URL: <http://sing.sourceforge.net> , October, 6th, 1998.
- Michael Zalewski y William Stearns, *p0f*, URL: <http://www.stearns.org/p0f/> .

A. GNU Free Documentation License

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

A.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

A.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

A.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

A.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

A.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.12. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.