

IRC-Client

IRC Group: Arne Bochém, Benjamin Maas, David Weiss

July 6, 2007

- 1 Introduction
- 2 iroffer bot behaviour
- 3 Implementation
- 4 Results
 - Regions
 - Operating Systems

Outline

We will, in the following 30 minutes present the work and results of the IRC-Client group as follows:

- Introduction
- iroffer bot behaviour
- Implementation
- Results

Purpose

The purpose of this project was to implement an IRC client, which is able to connect to IRC servers, join channels and fully automatically retrieve the IP addresses of, and other information about XDCC bots.

XDCC bots are used to serve files (mostly software/movies etc.) and usually run on hacked computers. By collecting their IP addresses, some insight can be gained which parts of the world are most often targeted by botnet owners.

Bot statistics

- Bots periodically advertise their files in the channels (which are usually moderated, but bots have "voice" privileges).
- First: Information about the bot's bandwidth: Statistics (e.g. peak bandwidth) and information about current transfers.

Example:

```
PRIVMSG #CX-ALLMP3S : ** 27 packs ** 1 of 1 slot  
open, Max: 2000.0KB/s, Record: 51.5KB/s  
PRIVMSG #CX-ALLMP3S : ** Bandwidth Usage **  
Current: 0.0KB/s, Cap: 6000.0KB/s, Record:  
199.3KB/s
```

Triggers

- Then a "trigger" message is announced. All bots use the same one:

```
PRIVMSG #CX-ALLMP3S : ** To request a file, type  
"/msg cX-AllMp3s-XXAA-12345 xdcc send #x" **
```

- Finally the available files ("warez") are listed:

```
PRIVMSG #CX-ALLMP3S : #1 6x [617K] 13Fill Me Nice  
50k 35g
```

...

```
PRIVMSG #CX-ALLMP3S : #23 0x [ 63M]  
4(cX).Lori.McKenna.Unglamorous.  
[Advance].2007.(1ar).rar
```

DCC connection setup

- After identifying a bot, our client contacts it using the trigger:
`PRIVMSG cX-AllMp3s-XXAA-12345 :xdcc send #1`
- If the bot is busy, we may have to wait in its queue.
Eventually, the bot will send a DCC request (via CTCP),
specifying its IP address and a port number:
`DCC SEND 5MB.bin 3190071222 1024 632100`

Closing the connection

- Now we are supposed to establish a TCP connection with the bot. As we already have its IP address, we are no longer interested in the bot.
- Additionally, the bot may keep us updated on the status of the transfer via NOTICE. We can ignore this.

Example:

```
NOTICE unfr3k :** Sending you pack #1 (" 13Fill  
Me Nice 50k 35g"), which is 617KB (resume  
supported)
```

```
NOTICE unfr3k :** Closing Connection: Connection  
Lost (Connection reset by peer)
```


Controller

The Controller is basically the main function of the program. Its responsibilities are instantiating the ConfigReader and StatisticsAggregator, as well as launching a single thread for each server in the configuration.

Each such thread will instantiate an IRCWrapper and a BotLogic and give control to the IRCWrapper.

BotLogic

The BotLogic implements the IRCListener class. It receives events from the IRCWrapper, processes them and takes action as necessary. It will send IRC command via the IRCWrapper, when a bot was identified and tell the StatisticsAggregator to record relevant events.

IRCWrapper

The IRCWrapper implements the IRC protocol. For the basic TCP connectivity, it uses an instance of the Socket class. Incoming lines are parsed and accordingly, event handlers in a given class implementing the IRCListener interface will be called. This class also offers functions to send IRC messages and other "active" IRC functionality.

Socket

The Socket class offers both client and server socket functionality, with a simple interface.

ConfigReader

The ConfigReader does pretty much what one would assume it to do. It reads configuration files and offers a somewhat convenient way to access the information from it.

Most importantly, it offers a list of client identities (nickname, username, realname), a list of servers and for each server a list of channels.

StatisticsAggregator

The StatisticsAggregator is used to collect information about bots in an XML file. It tries to identify known bots in multiple ways, to make sure that information concerning a single bot is stored only in the entry for said bot. It also handles a couple of signals, so statistics are not lost on abnormal termination. On catching SIGHUP, statistics are saved, but the program is not stopped. This offers a convenient way to save statistics during normal operation.

AdditionalInformationCollector

Given an IP address and port, the AdditionalInformationCollector tries to identify the country a host is in. It will also try to use the SinFP fingerprinting engine to determine the OS of said host, if an open port was provided.

Regions

At this time, we observed a total of 1789 unique IPs running bots. Now some of the statistics, which can be gleaned from the collected data shall be show.

The "country" UNKNOWN represents all bots, where no country information could be retrieved from the IP address. Those bots are most likely misconfigured in some way and send an internal LAN IP or other bad data, like inverse endianness of the integer representing the IP in a DCC send request.

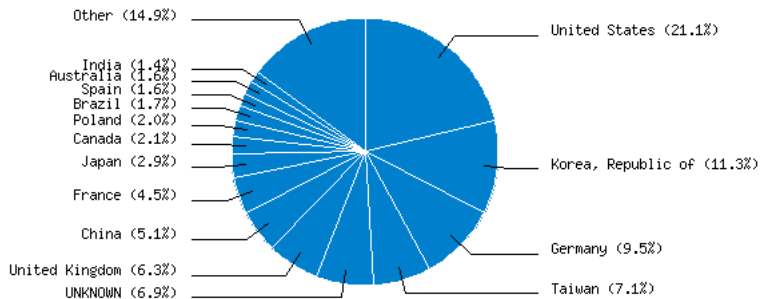


Figure: Distribution of XDCC bots by country (Top 15)

Regions (continued)

The USA and South Korea have the largest share of the bot-pie, followed by Germany, Taiwan and the UK.

Now for some more detailed information about offered/transferred data per country.

It is worth noting that, even though South Korea doesn't have the most bot-infested machines, those bots still store and transfer more data in total, than those of any other country.

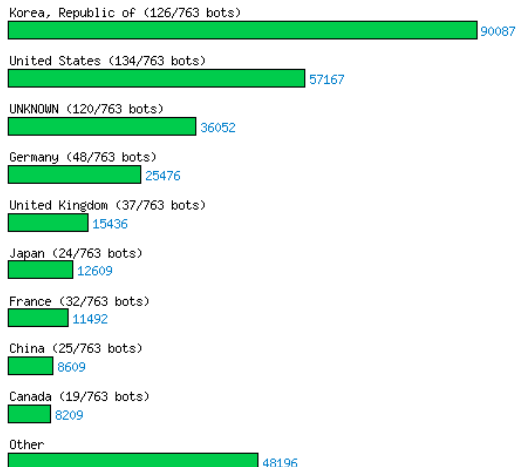


Figure: Total amount of data offered by country in MiB (Top 10)

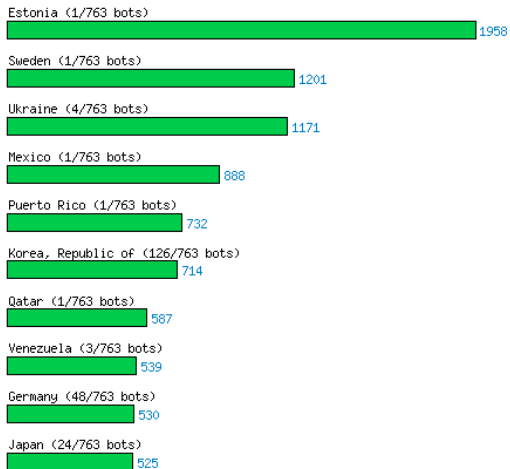


Figure: Mean amount of data offered per bot in MiB (Top 10)

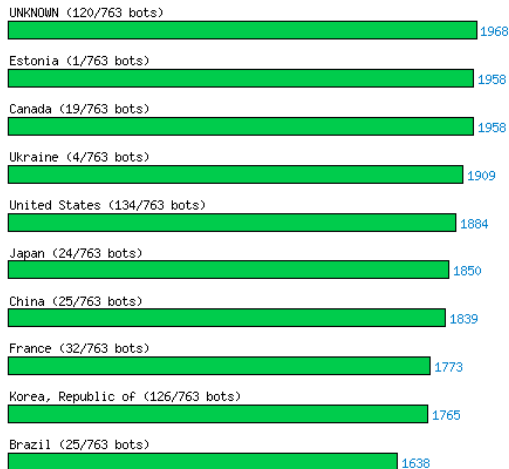


Figure: Maximum amount of data offered on single bot in MiB (Top 10)

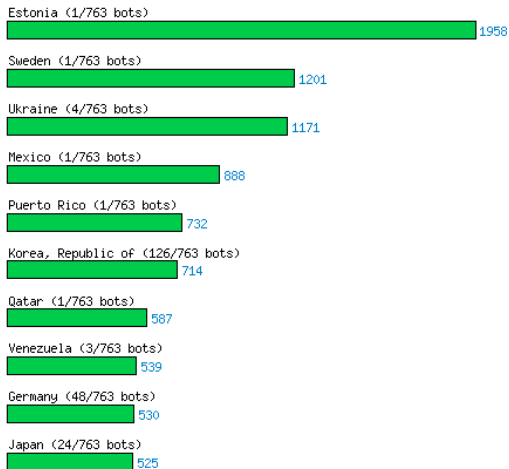


Figure: Mean amount of data transferred per bot in MiB (Top 10)

Regions (continued)

Examining the statistics for mean amount of data offered/transferred, it becomes obvious, that those charts are populated mainly with countries with few bots (1-4 bots). Disregarding these countries with few bots and looking for something a bit more statistically relevant, we observe that in both cases South Korea, Germany and Japan lead the chart, when only considering countries with non-insignificant amounts of bots.

Operating Systems

Now we shall take a look at the OSes run by the bot hosts.

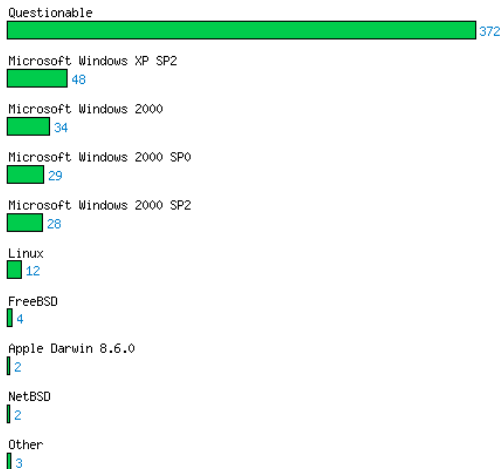


Figure: OSes of computers running XDCC bots (Top 10)

Operating Systems (continued)

Obviously the OS fingerprinting was less than perfect. The questionable results were mostly Linksys routers, which really shouldn't run XDCC bots and were probably misidentifications. Still, it is quite obvious that most of the "owned" machines run MS Windows (esp. Win2K).

There are also a few bots running Linux and other UNIX like OSes. These might be just slightly unusual botnet zombies, or they could be machines of people voluntarily running an XDCC bot for some reason.