IPTA

iptables Log Analyzer

Anders Sikvall

ichimusai.org

June 10, 2017

Version 0.1

http://ichimusai.org/projects/ipta
ichi@ichimusai.org

Contents

License

Copyright (c) 2015, Anders ”Ichimusai” Sikvall.
Latest revision June 10, 2017

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the ”Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

1. The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. It is not allowed to modify this license in any way.

2a. The original header must accompany all software files and are not removed in redistribution. It is allowed to add to the headers to describe modifications of the software and who has made these modifications. I claim no right to your modifications, they stand on their own merits.

2b. The license shown when the software is invoked with the ”–license” option is not removed or modified but must remain the same. You may add your name to the bottom of the credit part if you have contributed or modified the software.

3. You are allowed to add to the header files and the license information described in (2b) with changes and your own name, but only as an addition at the bottom of the file.

4. Distributing the software must be done in a way so that the software archive is intact and all necessary files (except external libraries such as MySQL) is always included. The software should be compilable after a reasonable set-up of the tool chain and compiler.

5. If you are making modifications to this software i humbly suggest you send a copy of your modifications to me on ichi@ichimusai.org and I may include your modifications (if you allow it) as well as give credit to you (if you want) in the next release of the software. This is only a suggestion to keep the code base in a single location but it is in no way a restriction to your right to modify and redistribute this software.

1 Introduction

I have searched the Internets for a while to find a good logging and reporting tool for IP tables but most tools were either too complex or not good enough. I am a firm believer in that security comes from being clear, simple and transparent and complexity is a very dangerous path. Therefore I decided to write my own version of a logging tool that can be used for anyone deploying a Linux server with IP Tables to get some kind of statistics for what is going on with their machine.

Now, this project is still in its initial phase, call it Alpha software if you like, so there may be lots of bugs and problems, missing functionality and other things. Some parts of this manual describes aims and goals rather than actually implemented features. I hope I will have marked those sections clearly enough.

If you would like to contribute to the software by debugging, writing code or otherwise maintain, improve upon or develop new features and so on, please let me know. You can always reach me by email to <anders@sikvall.se>.

The project is hosted on GitHub.com for the moment so it is rather easy to get started and contribute to it.

1.1 Project aims

The aims of the project is to create a tool to be able to analyze logs from the Linux kernel iptables logger and create reports on various issues, which types of packets are most commonly rejected, suspicious IP addresses in the world and many other things. It is a rather ambitious goal and my time is limited but I hope to spend some time on this and perhaps get a couple of other people involved in this.

We should rely as little as possible on external software. Not use obscure libraries that we are not able to distribute together with the source or the software and so on. Some things are unavoidable such as dependence on mySQL in this project wherefore those libraries needs to be installed but that should be easy to do.

Minimal configuration should be required to get up and running. A quick read through the manual and anyone really that knows their way around a terminal in Linux should be able to use this.

We do rely on a few outside components, mainly iptables itself and we will use MySQL as the database to collect the data into for easier processing. This requires the user to install MySQL if it is not already installed in the system, and set up usernames and a database for ipta to be using.

Regarding syslog it is also best to set it up so that it will process the log for iptables in a separate log file, but we will explain how to configure the most common syslogd in this manual also. Most syslog daemons have similar configuration files.

1.2 Bug reports

This is just an early pre-release really and there is a lot of things that are not included yet. You may find a few that are just stubs or that are not working properly.

Please report any other bugs you find on the issue tracker at Github[1]. It is much appreciated if you would take the time and describe how to re-create the bug.

You can of course also email me and I will create an issue for the bug and see what we can do for a resolution. Please email to <anders@sikvall.se> for reporting bugs and put “ipta: [description]” in the header.

2 Installation

This chapter describes the installation of the source package, the tools necessary to build it, compiling and installation of the software on the target machine.

2.1 Installing mysql server and client

The system relies on mysql server and client software installed on the local machine or another machine. Consult your system documentation on how to install the necessary tools.

For Ubuntu you would likely install something like this:

```
$ sudo apt-get install mysql-server mysql-client mysql-common
```

[1] https://github.com/sikvall/ipta

2.2 Installing the necessary tools

The ipta software is delivered as a software package and needs to be built for the platform you want to run it on. However the result is a single executable file which can easily be packaged and deployed in any software distribution you see fit.

The license in chapter grants you the right to deploy the software in any situation you seem fit, you are also free to modify the software and I only ask that if you make substantial improvements or bug fixes that you send them to me for consideration into the main software stream so others may benefit from this.

We need to install the GNU Compiler suite (GCC) and the C development libs for mysql as well as the version control system git. This is done like this in Ubuntu:

```
$ sudo apt-get install gcc libmysqlclient-dev git make
```

2.3 Download the source

Get the latest software package from the github repository, currently hosted at GitHub.com by issuing the following command.

```
$ git clone https://github.com/sikvall/ipta
```

This will give you a directory called ”ipta” in the current working directory which will bet git initialized and ready for you to work with or just compile.

2.4 Building the source and installing

You can now compile the software on you local machine by doing the following:

```
$ cd src
$ make all
$ make install
```

You should by now if no errors have been shown have a local binary of ”ipta” that you can start testing with. The last command will install the ipta binary under /usr/bin in your system with owner root and proper permissions.

The make install command will ask for your sudo password in order to install the binary to /usr/bin. If you do not want that you can run ipta straight from the src directory by just typing the following:

```
$ ./ipta <options> [arguments]
```

2.5 Making the manual

The manual is delivered as a PDF and as a LaTeX source this means you can compile it just like you do with the source code for the binary.

You need a different set of tools to make the manual from scratch but for most modern Linux systems you should probably have most of these tools already installed.

```
$ sudo apt-get install texlive
```

The TeX-Live distribution is the one recommended by TeX User Group and is the default one in Ubuntu and others. But other LaTeX can of course be used.

Go to the manual directory and type

```
$ make
```

This will start the make process, it will run "pdflatex ipta.tex" twice and build you a new pdf with the manual. Most users would however not need to build the manual but can just look in the provided pre-built PDF file instead.

You may also make postscript versions, or device independetn versions for printing and such with LaTeX but it is outside the scope of this manual to describe how this works.

2.5.1 Updating to the latest version

You can always update to the latest version by pulling the changes from the github repository at any time. This is done from the command line by changing to the ipta directory and issuing the following command:

```
$ git pull
```

This will pull the latest changes. When done you can always compile and install an updated version of the binary as described earlier. You will also receive the updated manual and any other accompanying file.

2.6 Setting up MySQL

Setting up MySQL for ipta is rather simple. You need to create a user, a database and grant all privileges on that database for the ipta user. If you do not configure ipta differently it will attempt to connect using the following credentials:

Username: ipta
Password: ipta

This can be overruled on the command line. With no other arguments it will try to access an existing database called ”ipta” and if no other name is given the default table is named ”logs”.

2.6.1 Create MySQL User

This assumes you already have MySQL installed and have access to the root account. During most installs you would have set up the root account when the MySQL database server was installed.

First you need to log in to MySQL:

```
$ mysql -u root -p
```

This will connect a shell to the local MySQL database with user root asking you to provide a password. Once logged in you would be presented with the MySQL prompt. It is then time to create a new user, a database for ipta and set the permissions to use it.

```
mysql> CREATE DATABASE ipta;
mysql> GRANT ALL ON ipta.* TO ’ipta’@’localhost’ IDENTIFIED BY ’ipta’;
mysql> FLUSH PRIVILEGES;
mysql> quit
Bye.
```

You can now verify that it works by invoking ipta to create the table needed for analysis. This is done by the following command:

2.7 Initiate the database

The next step is to set up the database table needed for ipta to work properly. The default table is “logs” in the database “ipta” but you can use command line switches to override this as fit. There is also a configuration file you may store this in so you don’t have to type it in every time.

```
$ ipta --create-table
* Table 'logs' created in database which you may now use.
```

If you get the mssage that the table was successfully created we are fine. If you get a message it already exists you are probably fine. If you get an error, check your MySQL statements again and see that you did create the ipta user and password is correct and so on.

Now we have everything set up in MySQL and can move on the next part, setting up iptables and syslog.

2.8 Setting up syslog

In many systems the iptables messages will be logged to the same system log as every other kernel message. This means we will have logs cluttered with all sorts of messages making it hard to see what is what.

By breaking out the iptables messages to a separate file we will not clutter the standard system log but can keep it separate and it also means the processing of the log file will be more efficient by ipta.

The ipta tool always looks for the ”IPT: ” prefix on the log line and will ignore any line that do not start with this. In order to set up syslog properly for our purpose we need to make some changes to its configuration file.

In this case we are working with *rsyslogd* which is the standard syslogger in Ubuntu. If you have a different syslog daemon you may want to consult the manual to do thesame thing. Most have fairly similar syntax in their configuration files.

2.8.1 Make syslog use a separate log file for iptables

In Ubuntu 14.04 you find the configuration in /etc/rsyslog.d where you can create a new file called 20-iptables.conf and the content of this file should be:

```
# Syslog iptables to separate file
:msg, contains, "IPT: " -/var/log/iptables.log
# Remove from the other logs
&~
```

Save this to the disk as the file /etc/rsyslog.d/20-iptables.conf and then issue a service restart for the syslog daemon by the command:

```
$ sudo service rsyslog restart
```

It should now restart with the new configuration and hopefully log all iptables messages to a new file.

2.9 Enable log-rotate

Since the logfile has a potential of becoming big, I suggest rotating it on a daily basis and just keeping what you need. It is probably a good idea to zip the files older than today and yesterday also to save space on the disk.

To set this up you need to find your logrotate.d directory, it usually resides in /etc/logrotate.d/ and if you go there you will find the configuration files for logrotate. One of them is probably already made for syslog, in Ubuntu 14.04 it will be called rsyslog. Open and edit this file carefully with an editor of your choice.

```
/var/log/iptables.log
{
    rotate 30
    daily
    missingok
    notifempty
    compress
    delaycompress
    sharedscripts
    postrotate
        reload rsyslog >/dev/null 2>&1 || true
    endscript
}
```

This will rotate your iptables.log every day and keep them 30 days back. It will perform some housekeeping as well as restarting the rsyslog (you may need to change that to the syslog in your system) in order to keep sytems nice. It will also compress all files from day before yesterday and back while keeping the two latest days, today and yesterday, uncompressed.

Of course if you want to rotate on a weekly basis or something else, you can change this accordingly. Insert it into the rsyslog or create a new one called "ipta" or something like that, then save it. Check it out a couple of days later that it is actually rotating your logs.

2.10 Setting up iptables

Now it is time to setup the iptables to log what we want. You should be familiar a little bit with iptables already, you do not have to be an expert but some familiarity with the basic functions is required here. If you feel you don't have that please take some time to read through a few of the various good HOWTO documents that are out there and familiarize yourself with iptables.

2.10.1 Creating the logchains

Basically what we will do is create two new chains in iptables that can be used for targets from other rules. One is to log packets that we want to DROP and the other is to log packets we accept but still wants to log.

We can also elaborate on this by creating a log chain for packets that are invalid and packets from certain ranges of IP addresses we wish to block and so on, the possibilities are really endless.

Let's start with creating a log chain for packets that are dropped. We will at the beginning somewhere, before any rules but after the policy setting in the iptables script insert something like this:

```
iptables -N LOGDROP
iptables -A LOGDROP -j LOG --log-prefix "IPT: DROP " --log-level 7
iptables -A LOGDROP -j DROP
```

Now every time in your firewall that you will drop a packet, instead of using the usual target DROP you will replace it with LOGDROP instead. If you still want to drop some packets without logging you can of course still use the DROP target.

We will also create a chain to handle packets that are accepted but logged and it can look like this

```
iptables -N LOGACCEPT
iptables -A LOGACCEPT -j LOG --log-prefix "IPT: ACCEPT " --log-level 7
iptables -A LOGACCEPT -j ACCEPT
```

Now we can see that the prefix we use tells ipta something on what is going on. First the marker ”IPT:” is recognized as this is a line that should be processed, the next word is a signal word that tells ipta what has happend. This should be ACCEPT, DROP, INVALID or something similar. ACCEPT and DROP should be used mainly as they have special meaning in the processor.

The signal word will be put into the database as the action field and used in statistics so create as many as you like, keep them short and to the point and prefer to use the ACCEPT and DROP unless you really want to distinguish between drops for different reasons.

Some action words used in the past that may give you some idea here:

Action	**Description**
DROP	Used generally for dropped packets
ACCEPT	Used generally for accepted packets
INVALID	Packets specifically dropped as invalid
CBLK	Packets that are CIDR blocked (country block)

2.10.2 A full iptables script

Normally a firewall configuration is not input manually but rather there is a script that is executed just after boot or sometimes by a cron job or similar. This script can take many forms but they generally do something like this:

1. Clear all previous rules
2. Set default policies
3. Set rules for accepting traffic to specific services
4. Deny all traffic not accepted

Below is an example iptables script, adapted to the ipta tool as well that you can use and expand upon yourself for your system. This should give you a general idea on how to set it up.

To fully explain all things with iptables is outside the scope of this manual but there are many tutorials and howto-docs on the net that will help you get started. The one below is rather well documented. It will also assume anything related to the loop-back interface (lo) is considered safe, if not, you can take that part out.

```
#!/bin/bash

# This is the IP Tables script used for ipta (IP Tables Analyzer)
# version x.x.x If your binary for iptables is located in a different
# directory change this definition

IPTABLES="/sbin/iptables"

# Setting default policies if you want to log dropped packets the
# default policy should be ACCEPT so that you can send them to the log
# facility, otherwise they will be dropped silently

\$IPTABLES -P INPUT   ACCEPT
\$IPTABLES -P OUTPUT  ACCEPT
\$IPTABLES -P FORWARD DENY #Change to ACCEPT ifyou use forwarding

# Flushing existing rules to make sure our iptables are empty
\$IPTABLES -F

# Create the logging chains for our iptables

# What happens here is that we add two rules to each chain. One that
# sends the packet to be logged via syslog using a recognizable prefix
# as well as a single word telling what happened to the packet (DROP
# or ACCEPT) and then the packet is either dropped or accepted.

\$IPTABLES -N LOGDROP
\$IPTABLES -A LOGDROP -j LOG --log-prefic "IPT: DROP " --log-level 7
\$IPTABLES -A LOGDROP -j DROP

\$IPTABLES -N LOGACCEPT
\$IPTABLES -A LOGACCEPT -j LOG --log-prefix "IPT: ACCEPT " --log-level 7
\$IPTABLES -A LOGACCEPT -j ACCEPT

\$IPTABLES -N LOGINVALID
\$IPTABLES -A LOGINVALID -j LOG --log-prefix "IPT: INVALID " --log-level 7
\$IPTABLES -A LOGINVALID -j DROP

# Drop early all malformed packets
```

```
\$IPTABLES -A INPUT -m state --state INVALID -j LOGINVALID

# Allow all traffic on local interfalce "lo" without logging
\$IPTABLES -A INPUT  -i lo -j ACCEPT
\$IPTABLES -A OUTPUT -i lo -j ACCEPT

# Enable ICMP but log them
\$IPTABLES -A INPUT -p icmp -j LOGACCEPT
\$IPTABLES -A OUTPUT -p icmp -j LOGACCEPT

# Enable DNS server connections from local host, no logging here DNS
# is specified for both tcp and udp connections but almost always only
# udp is used so you may disable the tcp parts if you like

\$IPTABLES -A INPUT  -p tcp --sport domain -j ACCEPT
\$IPTABLES -A INPUT  -p udp --sport domain -j ACCEPT
\$IPTABLES -A OUTPUT -p tcp --dport domain -j ACCEPT
\$IPTABLES -A OUTPUT -p udp --dport domain -j ACCEPT

# If you are using NTP for time synchronization you should have these
# enables, otherwise comment them out. I like to log them also.

\$IPTABLES -A INPUT  -p udp --sport 123 -m state --state ESTABLISHED -j LOGACCEPT
\$IPTABLES -A OUTPUT -p udp --sport 123 -m state --state NEW,ESTABLISHED -j LOGACCEPT

# This enables incoming SSH connections to the local host we will log
# this activity. Logging the ESTABLISHED state usually renders a lot
# of packets in the log since that would be file transfers and so on,
# therefore we just log the new packets.

\$IPTABLES -A INPUT  -p tcp --dport 22 -m state --state NEW -j LOGACCEPT
\$IPTABLES -A INPUT  -p tcp --dport 22 -m state --state ESTABLISHED -j ACCEPT
\$IPTABLES -A OUTPUT -p tcp --sport 22 -m state --state ESTABLISHED -J ACCEPT

# Enable traffic to your web server on port 80, no logging

\$IPTABLES -A INPUT  -p tcp --dport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
\$IPTABLES -A OUTPUT -p tcp --sport 80 -m state --state ESTABLISHED -j ACCEPT

# Everything else that falls through here will be considered
# non-wanted traffic and will therefore be logged also, then dropped.

\$IPTABLES -A INPUT  -j LOGDROP
\$IPTABLES -A OUTPUT -j LOGDROP
```

The above should give you a good starting point that you can modify into your own needed rules for the firewall you are deploying. You may of course simplify it or make it as complex as needed but this should show the geist of how you can use the different log chains to log packets that you want to analyze later.

Now, this script above creates an iptables firewall, make sure you have a way to connect to your computer before you run it in case you loose the connection with it. Don't update firewalls remotely unless you are certain of what you are doing.

This version makes use of the following actions:

- INVALID - packets that are invalid

- DROP - when a packet is dropped
- ACCEPT - when a packet is accepted in or out of the system

You can add others if you like so that you can distinguish between packets from known hosts etc, the limits are endless. We do recommend you have these three log chains however since the statistict module in ipta looks for these three actions.

2.11 Logging loops caution

As with any complex system there is the potential to have unintended consequences depending on how you set it up. During development I found out that it is easy to get into a ”logging loop” with the `--follow` mode engaged. The problem here is that if you log certain types of packets, such as DNS requests (port 53) or ssh (port 22) and you use the `--rdns` option or view the traffic over SSH the ipta may generate it's own traffic that in its turn creates even more traffic as it attempts to display the already generated traffic.

To avoid this you should take care to not log packets for an established SSH session if that is how you are running ipta, or DNS queries if you are using the `--rdns` switch.

3 Usage and Syntax

This tool is used to parse log files from syslogd regarding IP-tables data. Depending on how you set your iptables up to log you may log only denied, accepted or both types of packets. This is entirely up to you and you must configure the iptables on your machine to do some logging first in order to get some data for the ipta import module to work with.

You need to have your MySQL setup for this tool to work. Also, remember that you may want to clear your existing tables when starting a new import, otherwise it will just be appended to the earlier data that you already have in your logs. If you import the same file over you would skew the results as you would have twice the entrys for the same packets being logged.

You can change the settings for the database in the ipta-system configuration file in your home folder. It should be called .ipta-conf and will be read upon start to find your database names, passwords etc.

3.1 Options

Any settings in the configuration file can be overridden with command line parameters. They way ipta decides on the configuration is that first it will use the ones given on the command line. If no options are given it will check the configuration file and if there is no config file it will use default settings.

3.1.1 Mode switches

Mode switch	Description
`-c, --clear`	Clears all entries in the database. Can be used in front of the import directive to clear the database before importing new data.
`-i, --import <file>`	Import a file into the database. If the database already contains data the new data will be appended to the existing. If you do not wish to add to the data use the -c or –clear directive in front of the import directive in order to clear first, then import.
`-a, --analyze`	This is a mode switch and tells ipta to do the automatic analysis that's built in to the software. The output is sent to standard output.
`-f, --folow <file>`	This changes the behaviour of ipta into a real-time analyzer. It will follow the log file and any new packets that are logged will be presented as a line on the screen to stdandard output. Every 20 lines a new header is written detailing the fields to aid the operator idetinfyin what he sees on the screen. This is useful for real-time monitoring of any logged packets. It will not affect the database in any way or store any data so all the switches and parameters dealing with that has no effect when this mode is employed.
`-ai, --analyze-interactive`	Not yet implemented. In the future this will allow you to open a shell and put custom queries to the database in such a way that you can create your own analysis.

3.1.2 Database options

Database options	Description
`-d, --db-name <name>`	Use a different database called name instead of the one in the configuration file or defaul one (ipta). Useful when you are running multiple analysis in the same MySQL database or for some other reason can not use the default name.
`-h, --db-host <host>`	Use a different host name for the MySQL database. If not, ipta assumes localhost as the host for the MySQL database.
`-u, --db-user <name>`	Username that ipta uses to connecto to the MySQL database.. This overrides the name in specification file and the default name "ipta".
`-p, --db-pass <pass>`	Password to use for connecting to the MySQL database. If this is not given, the configuration file password will be used or if that is not designed the default password "ipta" will be tried.
`-pi, --db-pass-i`	Asks for the pasword interactively, otherwise the same thing as `-p, --db-pass` will do. The reason to use this one rather than the other one is that passwords entered on a command line may show up in the process table and rea not considered secure at all.
`-t, --db-table <table>`	Use the table named "table" instead of the default one or the one in the configuration file. The default table name that ipta works with would be "logs".
`-lt, --list-tables`	List all the tables in the database ipta is using. This is useful if you work through different tables and want to list their names.
`-dt, --delete-table <table>`	Remove any unwanted "table" from the database. The table and all information is erased. This can not be undone.

`-da, --delete-all-tables`	Remove all tables from the database. This will unceremoniously delete all the data in the ipta database.
`-ct, --create-table [name]`	Create a new ipta table. This could be the default table or you can supply an argument to give the new table a different name.
`-s, --save-db`	Not yet implemented. This option will write out a configuration file to the user's home directory named .ipta with the current settings. By providing configuration of database name, user, password and table to use as default these setting will be written to a file and used as default settings next time you run ipta.

3.1.3 Format switches

Format switches	**Description**
`-r, --rdns`	Do a reverse DNS lookup on the IP addresses that are being analyzed. This will in the reports replace the IP numbers usually seen with the DNS names for those addresses (if they exist). If no reverse DNS record exist, the IP address will be shown instead. The –rdns flag will sometimes give you good information about the host sending the packets.
`-l, --limit` <num>	The standard number of lines presented in the analyzer module of ipta is 10 lines per analyzer submodule. With this parameter you can change that to <num> lines instead of the default.
`--license`	Print the license information to standard output. The license is a modified version of the MIT license giving you the right to do anything with the software as long as you keep the attribution.
`--no-header`	This switch removes the headers from the output when the `--follow` mode is being employed.
`--no-counter, -C`	Removes the line counter in front of the lines in `--follow` mode.
`--no-local`	This switch removes the local inteface "lo" from the statistics when interpreting the data in the analyzer mode.
`--no-accept`	This switch removes all packets marked with the action "ACCEPT" from the analyzer output and focuses the analysis on the dropped and invalid packets.

3.1.4 DNS Cache switches

DNS Cache options	**Description**
`--ttl <hours>`	This switch sets the number of hours for the entries in the DNS cache to be considered valid. The default (no switch given) is 14 days. That's a pretty long time but most IP:s reverse DNS changes very seldom. Using this switch you can set the maximum allowed time before they have to refresh to `<hours>` instead of the default which is 336 hours.
`--dns-prune`	Remove all entries in the system that are older than the dns ttl which is default 14 days (336 hours). If you want to change the ttl you can combine this with the `--dns-ttl` switch in order to use a longer or shorter time to live for the entries.
`--dns-create-table`	Create the DNS table in the database. Default the table is simply called "dns" but you can override this by using the switch to change th name of the dns table or set it in the configuration file.
`--dns-table [table]`	Change from the default database table "dns" to a new one for this run. Don't forget it must be created with the `--dns-create-table` also before it can be used.
`--dns-dump`	Dumps the DNS table as it is now with ip address and host name formatted in an easy to read fashion. Can be used for further analysis.

3.1.5 Miscellaneous switches

Miscellaneous	Description
`--license`	Print the license and then exit.
`--version`	Print version information then exit.
`--usage`	Print quick usage instructions then exit.

4 How to use ipta

The first thing you need to do is set up your database, iptables log and logrotate as previously described in this manual. Once you have that working it is time to use the tool.

We can start by initializing the database:

```
\$ ipta --create-table
```

This will create the default table in the database and prime it ready for use. The next step is to import the current iptables log file into the MySQL database where it can be later easily analyzed.

This is done by the import mode and triggered by the mode switch `--import` followed by the file name. It could be something like this:

```
\$ ipta --import /var/log/iptables.log
```

Once the log has been properly imported, and the progress will be shown during the import which will sometimes take several minutes if it is a big iptables logfile.

Now we can analyze the imported data by taking a look at it with ipta's buildt in analyzer module. This is triggered by the mode switch `--analyze` and will show you various outputs depending on the packets you have logged.

5 Analyzer modules output

The analyzer runs a number of automated queries on the MySQL database and then collects the results and formats it in a visual pleasing way so it is easy to contrast and compare and keep track of what goes on in the system. This part will describe the different modules used today.

5.1 Denied traffic grouped by IP, destport and action

This module shows everything that is NOT marked as "ACCEPT" and therefore is assumed denied. It then groups it by IP address, destination port number and which action was taken with the packet. It is useful for an overlook of what kind of traffic we get that gets denied.

By default the top ten lines are shown, this can be changed with the `--limit` option.

```
Showing denied traffic grouped by IP, destination port, action taken and protocol.
 Count Source IP                SPort Dest IP                  DPort Proto  Action
------ ------------------------ ----- ------------------------ ----- ------ ----------
   138 213.136.38.8             41757 188.126.93.160              80 TCP    INVALID
    14 125.16.128.122           62215 188.126.93.160              80 TCP    INVALID
    14 204.124.183.98           53512 188.126.93.160              80 TCP    INVALID
    14 91.200.12.11             56310 188.126.93.160              80 TCP    INVALID
    10 194.71.19.244            48725 188.126.93.160              80 TCP    INVALID
     9 176.111.61.12            50139 188.126.93.160              80 TCP    INVALID
     9 117.21.176.95             6000 188.126.93.160            1433 TCP    DROP
     9 71.176.122.34             8235 188.126.93.160              80 TCP    INVALID
     9 176.61.140.118            4080 188.126.93.160              25 TCP    DROP
     8 194.63.142.101           36697 188.126.93.160             110 TCP    DROP
     8 201.157.0.78             50608 188.126.93.160              80 TCP    INVALID
     7 121.155.129.180          48461 188.126.93.160           10142 UDP    DROP
     6 171.96.241.142           45479 188.126.93.160              25 TCP    DROP
     6 194.153.119.59           63395 188.126.93.160              80 TCP    INVALID
     5 200.5.112.174             3163 188.126.93.160              80 TCP    INVALID
     4 222.186.31.137            6000 188.126.93.160            1433 TCP    DROP
     4 176.37.170.58            40466 188.126.93.160            8080 TCP    DROP
     4 162.244.35.24            41695 188.126.93.160           21320 TCP    DROP
     3 212.162.17.234            3371 188.126.93.160              23 TCP    DROP
     3 222.186.190.157          43484 188.126.93.160           20022 TCP    DROP
```

5.2 ICMP module

The next module shows the ICMP traffic grouped by IP addres and action. This is usefull since it is both an indication if there is a problem to a certain host as well as if there is somebody spraying the host with ICMP packets such as various ping scans and so on.

This module shows all traffic logged with protocol ICMP no matter the status.

```
Showing ICMP traffic statistics
 Count Source IP                Dest IP                  Action
------ ------------------------ ------------------------ ----------
    14 188.126.93.160           129.82.138.44            ACCEPT
     3 128.9.168.98             188.126.93.160           ACCEPT
     3 203.178.148.19           188.126.93.160           ACCEPT
     3 129.82.138.44            188.126.93.160           ACCEPT
     2 195.251.255.69           188.126.93.160           ACCEPT
     1 190.7.215.194             188.126.93.160            ACCEPT
     1 14.8.249.133              188.126.93.160            ACCEPT
     1 124.65.193.150            188.126.93.160            ACCEPT
```

5.3 Most denied ports module

This part shows grouped on port number and action what goes on and is sorted top-down showing the most denied ports first. This is useful to determine if any certain services are being attacked or what is going on in connection attempts.

```
Most denied ports
 Count   DPort   Proto    Action
------   -----   ------   ----------
   234        80   TCP      INVALID
    56        23   TCP      DROP
    22        25   TCP      DROP
    14      1433   TCP      DROP
    13        21   TCP      DROP
    10      8080   TCP      DROP
     9       110   TCP      DROP
     9      3389   TCP      DROP
     7      5060   UDP      DROP
     7     10142   UDP      DROP
```

5.4 Invalid packets source

This module sorts invalid packets (action = “INVALID”) based on the number of packets, their origin and also shows source and destination ports for the packets.

It is useful to determine network problems as well as malformed packets or source routed spoofed packets.

```
Most invalid packets comes from
 Count   Source IP                   SPort   Dest IP                      DPort   Proto
------   -------------------------   -----   -------------------------   -----   ------
   138   213.136.38.8                41757   188.126.93.160                 80   TCP
    14   125.16.128.122              62215   188.126.93.160                 80   TCP
    14   91.200.12.11                56310   188.126.93.160                 80   TCP
    14   204.124.183.98              53512   188.126.93.160                 80   TCP
    10   194.71.19.244               48725   188.126.93.160                 80   TCP
     9   176.111.61.12               50139   188.126.93.160                 80   TCP
     9   71.176.122.34                8235   188.126.93.160                 80   TCP
     8   201.157.0.78                50608   188.126.93.160                 80   TCP
     6   194.153.119.59              63395   188.126.93.160                 80   TCP
     5   200.5.112.174                3163   188.126.93.160                 80   TCP
     3   8.2.122.80                  28016   188.126.93.160              38267   TCP
     3   109.190.104.220             52561   188.126.93.160                 80   TCP
     3   177.91.97.84                50385   188.126.93.160                 80   TCP
     2   104.28.11.14                   80   188.126.93.160              65399   TCP
     1   178.63.22.198               19505   188.126.93.160              54699   TCP
   1   195.198.171.84            49473   188.126.93.160               80   TCP
   1   108.167.143.218              80   188.126.93.160            28498   TCP
```

5.5 Interface statistics

This module shows the packets count per action and interface. It is useful as a marker and keep an eye on things in Interface statistics general.

```
 Count   IF In        Action       Proto
------   ----------   ----------   -----
   241   eth0         INVALID      TCP
   217   eth0         DROP         TCP
    43   eth0         DROP         UDP
```

6 Follow mode

The ipta follow mode is special because it shows real-time, or at least close to real-time what goes on when the system is running. Every packet logged will be subject to ipta's formatting functions and displayed on a single line.

You need a terminal with wide lines to make proper use of this but that should generally not be too much of a problem in this day and age.

The follow mode is iniated by the following command:

```
$ ipta --follow /var/log/iptables.log
```

You may also want to combine this with the switches `--no-lo` to remove the packets to and from the local interface as this is traffic that is internal on the host and perhaps also combine it with the `--no-accept` switch if you only wish to see denied traffic.

Another very useful switch would of course be the `--rdns` switch which will reverse look-up the host name if it exists of the source and destination IP of the address.

The follow mode shows most of the interesting parameters in the table such as source IP and port as well as destination IP and port and the action taken on the packet. With the `--no-accept` flag the system will skip any line with the action set to ACCEPT of course.

```
Count    IF       Source                          Port Destination                     Port Proto      Action
-------- -------- ------------------------------ ----- ------------------------------ ----- ---------- ----------
    1951 eth0     *6f44.cust.bredbandsbolaget.se 35396 zathras.ichimusai.org          15003 TCP        DROP
    1952 eth0     *6f44.cust.bredbandsbolaget.se 50758 zathras.ichimusai.org           6001 TCP        DROP
    1953 eth0     *6f44.cust.bredbandsbolaget.se 45937 zathras.ichimusai.org           1580 TCP        DROP
    1954 eth0     *6f44.cust.bredbandsbolaget.se 48118 zathras.ichimusai.org          42510 TCP        DROP
    1955 eth0     *6f44.cust.bredbandsbolaget.se 47564 zathras.ichimusai.org           3871 TCP        DROP
    1956 eth0     *6f44.cust.bredbandsbolaget.se 36463 zathras.ichimusai.org            563 TCP        DROP
    1957 eth0     *6f44.cust.bredbandsbolaget.se 41547 zathras.ichimusai.org           1066 TCP        DROP
    1958 eth0     *6f44.cust.bredbandsbolaget.se 41956 zathras.ichimusai.org           1287 TCP        DROP
    1959 eth0     *6f44.cust.bredbandsbolaget.se 50023 zathras.ichimusai.org           9917 TCP        DROP
    1960 eth0     *6f44.cust.bredbandsbolaget.se 59197 zathras.ichimusai.org          50001 TCP        DROP
    1961 eth0     *6f44.cust.bredbandsbolaget.se 53897 zathras.ichimusai.org           1272 TCP        DROP
    1962 eth0     *6f44.cust.bredbandsbolaget.se 34387 zathras.ichimusai.org           1112 TCP        DROP
    1963 eth0     *6f44.cust.bredbandsbolaget.se 52943 zathras.ichimusai.org           1218 TCP        DROP
    1964 eth0     *6f44.cust.bredbandsbolaget.se 37205 zathras.ichimusai.org           1185 TCP        DROP
    1965 eth0     *6f44.cust.bredbandsbolaget.se 42648 zathras.ichimusai.org           1175 TCP        DROP
    1966 eth0     *6f44.cust.bredbandsbolaget.se 40920 zathras.ichimusai.org            683 TCP        DROP
    1967 eth0     *6f44.cust.bredbandsbolaget.se 36787 zathras.ichimusai.org          15660 TCP        DROP
    1968 eth0     *6f44.cust.bredbandsbolaget.se 41813 zathras.ichimusai.org           2525 TCP        DROP
    1969 eth0     *6f44.cust.bredbandsbolaget.se 57365 zathras.ichimusai.org           1108 TCP        DROP
    1970 eth0     *6f44.cust.bredbandsbolaget.se 48566 zathras.ichimusai.org           8443 TCP        DROP
```

7 Scan mode

Scan mode is a lot like the follow mode. It is invoked with the following syntax:

```
$ ipta --scan <logfile>
```

Options such as -l and -r are valid. The scan mode is a special form of follow mode that scans the whole log file and outputs every line from it.

This can be used together with other system tools to produce blocking lists or other records of what has happend over sime time back.

An example of this is to produce a blocking file for those who are attempting to connect on a port you suspect is for hacking such as port 23 (telnet) which you are not using.

By running `ipta --scan <logfile> | grep " 23 TCP"` you may find all attempts on port 23 for example.

8 Configuration file

The configuration file is a simple text file that contains key = value pairs. The file should be in the users home directory and be called ".ipta" for it to be found. All the keys in the file are optional. Keys missing from the file will means ipta use the default value instead. Keys in the config file can be overridden on the command line using arguments such as `--db-host` and similar.

8.1 Keywords

This describes the keywords in the configuration file.

Key	Value
analyzer limit	Sets the maximum lines to ouput in the analyzer module when ipta is being called with the `--analyze` argument. Some modules in the analyzer may output less lines however, but never more.
db_host	Sets the hostname for the MySQL server. Usually localhost for best performance but you may just as well use a separate database server if you prefer.
db_user	Sets the username when connecting to the MySQL database.
db_pass	Sets the password when connecting to the MySQL database.
db_name	Sets the database name to use, default is ”ipta”.
db_table	Sets the name of the table to use, default is ”logs”.
rdns	Takes ’yes’ or ’no’ depending on if you wish to use reverse DNS lookups as default or not. Omitted no reverse DNS lookup is performed.
counter	Takes ’yes’ or ’no’ depending on if you wisth to turn on or off the counter in –follow mode. Omitted it shows counters.
show-lo	Decides if you want to show activities on local interface ’lo’ (set to ’yes’) or not (set to ’no’). Omitted means yes.
header	Show header every 20 lines in –follow mode (set to ’yes’). If omitted it shows headers, if you don’t want headers set to ’no’.

8.2 Config example

This is an example of a configuration file.

```
## ipta configuration file

# Misc settings
analyzer limit = 15

# Database settings
db_host = localhost
db_user = ipta
```

```
db_pass = ipta
db_name = ipta
db_table = logs

# Flags
show-lo = no
header = yes
counter = yes
rdwn = yes

# End
```

You may copy and paste this to your init file.

9 Database format

Coming in next release

10 DNS Cache functions

To be implemented in next release.

The built-in DNS caching function will save you a number of lookups as it will first query the local database and only if the TTL has expired or if there is no record of the IP again will the operating system be queried to do a reverse DNS lookup on the IP number.

This save a large number of unnecessary requests but it also introduces the risk that we show old information to the user. Therefore a time to live should be used so that when we have old records we will still query the DNS subsystem rather than pulling old names out of our proverbial hat.

The defalt time to live is 14 days. While this is quite okay for most cases it may not work well if you have a lot of dynamically assigned names to IP addresses (although it is rarer, the reverse is much more common) you can modify this by using `--dns-cache-ttl <hours>` to expire it faster or slower.

14 days equals 336 hours anyhow and anything from sayt 6 hours to 900 hours should be pretty sane to use here. Using a ttl of 0 means the records will never expire. However they can be removed manually by using the `--dnsc-clear` switch that will remove all records irregardless of their time to live.

You can also clear the entire DNS cache by issuing the option `--dns-cache-clear` of course. This will remove all entries from the dns cache and start caching things anew.

10.1 Using BIND for local cache

If you have your own BIND system running on the local machine (for caching all DNS queries) you do not, and probably should not, employ the dns cache in ipta. The reason for that is that the proper BIND system have it's own TTL and will use the TTL set by the DNS servers upstream that it will query. It will likely be more efficient to use what you already have especially since other tools such as `traceroute` etc will be able to benefit from the cached BIND records.

So regard the ipta internal cache as something that will aid you in case you are not running your own caching DNS server.