

AUTOMATED DETECTION OF SECURITY AND
PRIVACY THREATS IN PEER-TO-PEER
NETWORKS

User manual for
IDPT: Intrusion Detection For Peer-to-Peer Threats

Submitted To:
Department of Electronics & Information Technology
(DeitY)
e-Security Division
Ministry of Communications & IT
Govt. of India

Submitted by:
Dr. Chittaranjan Hota
Dept. of Computer Sc. & Info. Systems,
BITS – Pilani, Hyderabad campus
Jawahar Nagar, Shameerpet Mandal
Telangana – 500078

Contents

1	Introduction	2
2	Prerequisites	2
3	System setup	4
3.1	Dashboard	5
3.2	Server-side scripts	6
3.3	Set up cron jobs	7
4	IDPT in action	10
	Acknowledgments	14
	Credits	14

1 Introduction

This guide will walk you through installation of IDPT as a network intrusion detection framework meant for Peer-to-Peer overlays. A web-based graphical user interface is also provided for easy viewing and analysis by a network administrator. Our system has been tested on Ubuntu 12.04 Server x64.

2 Prerequisites

Before installing the list of packages, make sure your operating system is up to date. Run the following commands on your terminal. Once all the updates are installed, reboot your machine.

Getting started ...

```
sudo apt-get update  
sudo apt-get upgrade  
sudo reboot now
```

IDPT requires libpcap library, Java, LAMP, PHP, GNU Parallel and Python's numpy and scipy installed. Let's install them next:

Install packages


```
sudo apt-get install libpcap-dev
sudo apt-get install openjdk-7-jre
sudo apt-get install parallel
sudo apt-get install python-numpy python-scipy
sudo apt-get install lamp-server
sudo apt-get install phpmyadmin
```

Please note that the package ‘Parallel’, by default, is installed in ‘tollef’ mode. We need ‘GNU’ mode for our modules. To get rid of the ‘tollef’ mode, delete the `--tollef` flag from the file `/etc/parallel/config`, or simply delete the file itself.

For MySQL and phpmyadmin, you will be required to set your passwords (for root as well as ‘user’ account). For this documentation, we use the default password ‘deity’ for all purposes. You should select your own password instead of this password.

3 System setup

Let us fetch the entire source code for server-side scripts as well as the web modules. You may download the zip file of source code from GitHub and unzip it, or clone/fork the repository. The commands provided below are for downloading the zip file. Our working directory will be called ‘idpt_source-master’.

 **Get source code**

```
wget https://github.com/NetClique/idpt_source/archive/master.zip  
unzip idpt_source-master.zip  
cd idpt_source-master
```

After unzipping, the structure should look similar to the one given in GitHub (as shown in Figure 1).

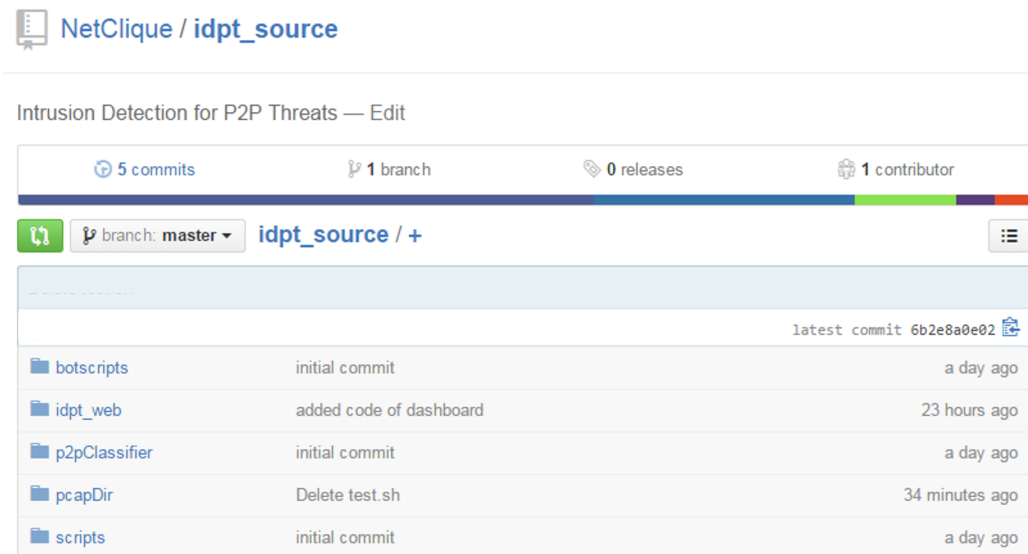


Figure 1: Directory structure of idpt_source-master

3.1 Dashboard

While you are inside 'idpt_source-master' directory, move the folder named 'idpt_web' to /var/www/:

Directory for Dashboard

```
mv idpt_web /var/www/
```

This directory now hosts the web module for IDPT. If apache is running fine on your system, you should be able to access the web module at:

<<host machine's IP>>/idpt_web/

If you access the dashboard, you should see the homepage as given in Figure 2.

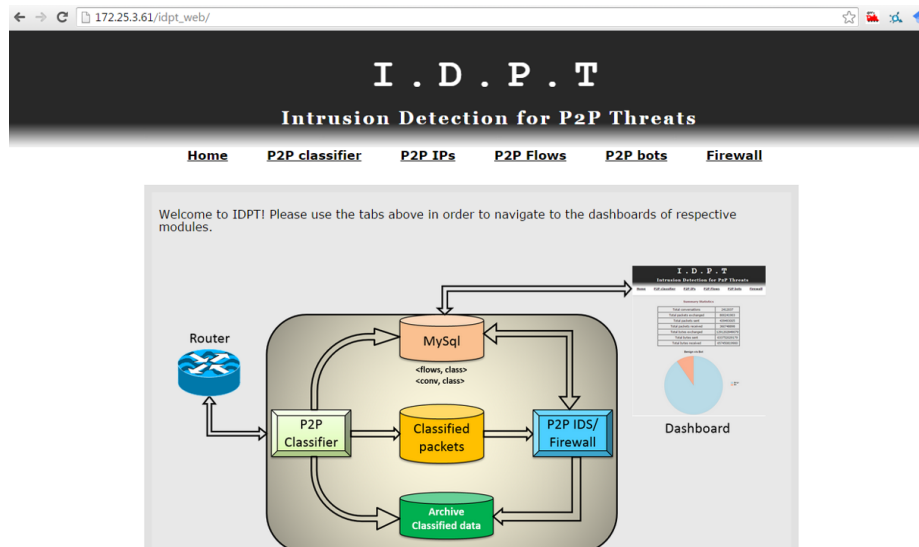


Figure 2: Homepage of IDPT dashboard

3.2 Server-side scripts

All other folders contain scripts required for server-side execution. Move all the other folders to the user's 'home' directory. For example, if the username is 'bits', all these folders should now reside under '/home/bits/'.

Run the command given below while you are inside 'idpt_source-master' directory:

Directories for Server-side scripts

```
mv * /home/bits/ #Remember to give the actual path of your user
```

The results of server-side scripts are stored in a database. Create the database and respective tables by running the script as shown below.

Database and tables

```
cd scripts  
bash createDB.sh
```


`createDB.sh` script will create the database named `idpt` for user `idpt`. Please note that user `idpt` should already exist. In case you are using some other username, please change that in `createDB.sh`. Also, you should grant all permissions to this user. `createDB.sh` will internally invoke `idpt.sql`, which will create the required tables for the `idpt` database.

Once you have created the database, you can access it using phpMyAdmin or the MySQL command line. The view of your database and its tables with phpMyAdmin should look similar to Figure 3.

<div><div>Structure</div><div>SQL</div><div>Search</div><div>Query</div><div>Export</div><div>Import</div><div>Operations</div><div>Privileges</div><div>Tracking</div></div>												
	Table	Action					Rows	Type	Collation	Size	Overhead	
<input type="checkbox"/>	convo_tbl	<div><div><div>Browse</div></div></div>	<div><div><div>Structure</div></div></div>	<div><div><div>Search</div></div></div>	<div><div><div>Insert</div></div></div>	<div><div><div>Empty</div></div></div>	<div><div><div>Drop</div></div></div>	~47,468	InnoDB	latin1_swedish_ci	9.5 MiB	-
<input type="checkbox"/>	flow_ensem	<div><div><div>Browse</div></div></div>	<div><div><div>Structure</div></div></div>	<div><div><div>Search</div></div></div>	<div><div><div>Insert</div></div></div>	<div><div><div>Empty</div></div></div>	<div><div><div>Drop</div></div></div>	~54,684	InnoDB	latin1_swedish_ci	8.5 MiB	-
<input type="checkbox"/>	flow_tbl	<div><div><div>Browse</div></div></div>	<div><div><div>Structure</div></div></div>	<div><div><div>Search</div></div></div>	<div><div><div>Insert</div></div></div>	<div><div><div>Empty</div></div></div>	<div><div><div>Drop</div></div></div>	~54,712	InnoDB	latin1_swedish_ci	8.5 MiB	-
	3 tables	Sum					~156,864	InnoDB	latin1_swedish_ci	26.5 MiB	0 B	

Figure 3: A snapshot of IDPT database from phpMyAdmin

Next, we will download jNetPcap library. Untar it. You will need to manually copy `libjnetpcap.so` into `/usr/lib/` directory using root privileges.


jNetPcap

```

wget http://slytechs.com/downloads/dist/a130819os/jnetpcap-1.4.
r1425-1.linux64.x86_64.tgz

tar -zxvf jnetpcap-1.4.r1425-1.linux64.x86_64.tgz

cd jnetpcap-1.4.r1425-1

sudo cp *.so /usr/lib/

sudo cp *.jar /usr/share/java/

```

3.3 Set up cron jobs

Crons automate the functioning of our server-side scripts which are used to parse network traces, extract relevant features, upload files to database, etc. The Crontab

for a particular user can be accessed by running `crontab -e`. Certain crons will be created for the 'bits' user; those which require root access to run will be created for the 'root' user.

Let us set-up the crons for the 'bits' user.

Crons for user 'bits'

Do `crontab -e`

If you are using `crontab` for the first time, some basic set-up (such as choosing your text editor) will follow.

Next, enter the following lines at the end of the crontab:

```
*/6 * * * * bash /home/bits/pcapDir/classifier.sh
```

```
*/6 * * * * bash /home/bits/pcapDir/dbUpload.sh
```

```
00 00 * * * bash /home/bits/scripts/truncateTables.sh
```

Save and exit.

The last script (`truncateTables.sh`) empties the database every day at midnight. If you do not want to empty it, you can change this action and take a back-up instead.

Next, let's set-up crons for the 'root' user. Please note that this requires an active 'root' user account.

Crons for user 'root'

```
sudo -i #become root user
```

```
Do crontab -e
```

If you are using crontab for the first time, some basic set-up (such as choosing your text editor) will follow.

Next, enter the following lines at the end of the crontab:

```
*/10 * * * * /home/bits/botscripts/run.in
```

```
*/6 * * * * bash /home/bits/pcapDir/fileMove.sh
```

```
*/10 * * * * /home/bits/scripts/blockIPs.sh
```

Save and exit.

With the crons in place, we are almost done with the set-up of the IDPT module. As soon as we start feeding data, the processing will begin. We will perform the task of initializing data captures and get a feel of the run of IDPT in the next section.


Summary Statistics	
Total conversations	7060963
Total packets exchanged	2823766885
Total packets sent	1620555372
Total packets received	1203211513
Total bytes exchanged	4686267107586
Total bytes sent	2490472835629
Total bytes received	2195794271957

Figure 4: Summary statistics viewed in the dashboard

4 IDPT in action

IDPT module receives network traffic in the form of `.pcap` files captured using `dumppcap`.

Let's begin this process:



Initialize data capture

```
cd scripts
sudo bash capture.sh
```

The script `capture.sh` invokes `dumppcap` with several pre-defined parameters. You are strongly suggested to go through these parameters and customize them for your network. For example, make sure `dumppcap` is listening on the right interface (specified using the `-i` option).

The server-side scripts of IDPT are automated as cron jobs. Once the crons and data capture are initialized, the module will begin its functioning. If the database is

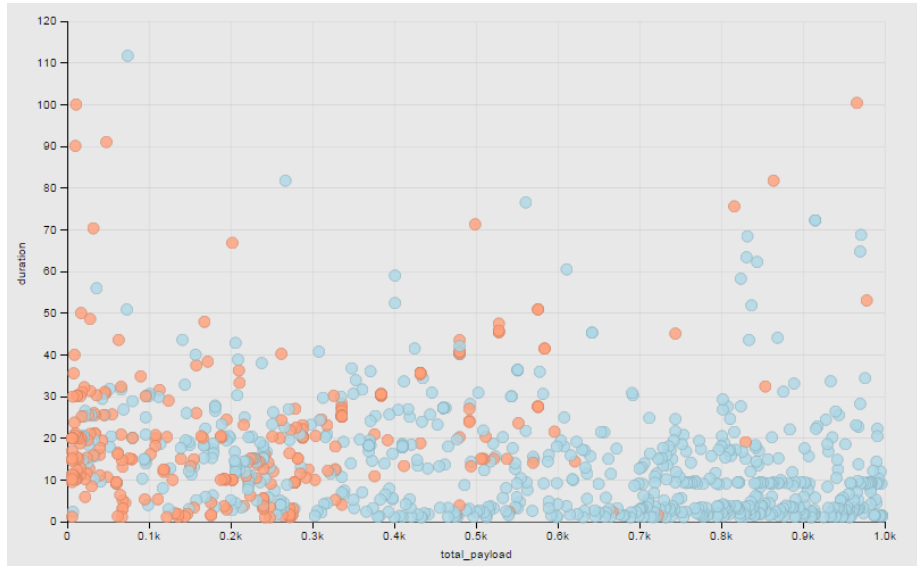


Figure 5: Scatter plot viewed in dashboard

properly set-up, you should soon see entries being populated into the dashboard. With the help of the dashboard, you can see the summary statistics and visualize various attributes of your traffic. Figure 4 shows the ‘summary statistics’ as displayed on the ‘P2P bots’ page.

Figure 5 shows a scatter plot of the ‘duration’ of conversations plotted against the ‘volume’. We track low-volume, high-durations conversations for the detection of P2P bots.

Since the database contains a large number of conversations, we provide a random sample of other features for visualization. Every time you refresh the page, a different sample will be fetched.

The ‘firewall’ tab on the dashboard displays top 50 IPs (outside your LAN) which were identified as malicious by IDPT. A snapshot is shown in Figure 6.

Blocking of traffic is performed by the `blockIPs.sh` script, which was set-up as a

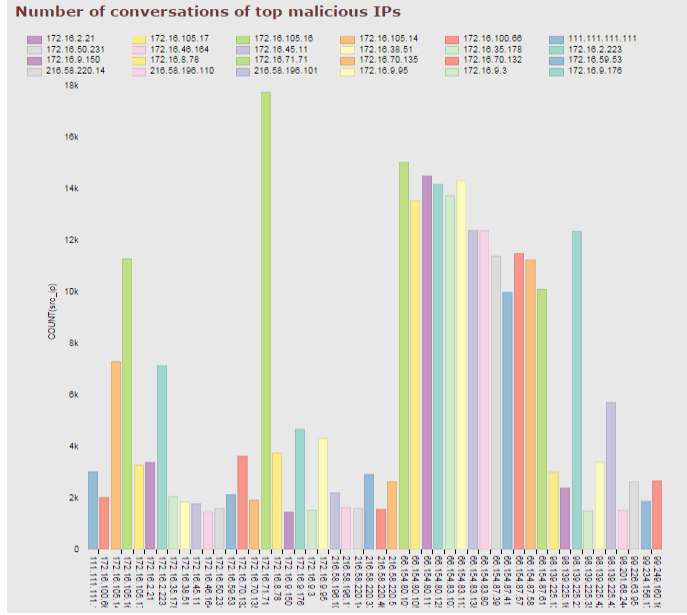


Figure 6: Top malicious IPs viewed in the ‘firewall’ tab

cron job. This ‘firewall’ module utilizes IPtables to create automated and dynamic firewall rules for hosts identified as running P2P apps and for hosts infected with bots. For the prototype implementation of our module, we have implemented firewall rules for top 20 IPs outside the LAN identified as P2P, top 20 IPs outside the LAN identified as bot-infected, and top 20 IPs inside the LAN identified as bot-infected. Apart from creating firewall rules pertaining to these set of hosts, we also created generic firewall rules to block popular P2P applications. Our firewall rules have a dynamic nature. We do not block/reject P2P traffic on weekends. On working days, P2P traffic is disallowed during the working hours (9:00 AM – 5:00 PM). Apart from working hours, P2P traffic is allowed. Bot traffic, however, is shown zero tolerance and is rejected under all circumstances. You can view all these rules in `blockIPs.sh` file, and you are free to modify, add or delete any of these rules to suit the needs of your network.



Attention!

The script 'blockIPs.sh' creates firewall rules for blocking/dropping the traffic.

This script should be used on a server which functions as a 'Gateway' machine.

If the machine is not a Gateway, the firewall rules will serve no purpose.

While the module is running, you can view the java and python modules being executed by running `top`, `htop` or `atop` on your command prompt. A sample output (obtained through `htop`) is given in Figure 7. The portion marked in red indicates the multiple processes of the P2P classification module.

```

1  [|||||] 100.0% 7  [|||||] 100.0% 13 [|||] 9.7% 19 [|||] 1.3%
2  [|||] 3.3% 8  [|||||] 76.0% 14 [|||||] 100.0% 20 [|||||] 22.7%
3  [|||] 2.6% 9  [|||||] 48.7% 15 [|||||] 100.0% 21 [|||||] 51.6%
4  [|||] 1.3% 10 [|||||] 24.2% 16 [|||||] 100.0% 22 [|||||] 78.1%
5  [|||] 0.7% 11 [|||||] 99.4% 17 [|||||] 99.4% 23 [|||] 1.9%
6  [|||||] 100.0% 12 [|||||] 23.4% 18 [|||] 1.3% 24 [|||||] 78.4%
Mem[|||||] 37490/64384MB Tasks: 90; 13 running
Swp[|||||] 17/65487MB Load average: 16.02 16.57 13.50
Uptime: 6 days, 04:28:56

RES S CPU% MEM% TIME+ Command
11152 2324 S 0.0 0.0 0:00.35 /usr/bin/perl -w /usr/bin/parallel -i 15 -a inputDir java -jar classifierDump.jar
612 512 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/14/
3027M 10748 S 101. 4.7 4:15.01 java -jar classifierDump.jar /home/bits/pcapDir/14/
612 512 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/10/
2564M 10748 S 100. 4.0 4:17.74 java -jar classifierDump.jar /home/bits/pcapDir/10/
612 512 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/9/
4188M 10748 S 99.0 6.5 4:13.50 java -jar classifierDump.jar /home/bits/pcapDir/9/
608 508 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/8/
3063M 10744 S 101. 4.8 4:14.59 java -jar classifierDump.jar /home/bits/pcapDir/8/
604 508 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/7/
3348M 10744 S 100. 5.2 4:28.24 java -jar classifierDump.jar /home/bits/pcapDir/7/
608 512 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/6/
1683M 10744 S 103. 2.6 4:18.04 java -jar classifierDump.jar /home/bits/pcapDir/6/
608 512 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/5/
1771M 10744 S 101. 2.8 4:23.74 java -jar classifierDump.jar /home/bits/pcapDir/5/
608 512 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/4/
3056M 10744 S 99.0 4.7 4:25.01 java -jar classifierDump.jar /home/bits/pcapDir/4/
608 512 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/3/
3224M 10744 S 100. 5.0 4:24.63 java -jar classifierDump.jar /home/bits/pcapDir/3/
612 512 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/2/
5314M 10748 S 100. 8.3 4:22.85 java -jar classifierDump.jar /home/bits/pcapDir/2/
608 512 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/1/
3028M 10744 S 101. 4.7 4:17.20 java -jar classifierDump.jar /home/bits/pcapDir/1/
608 512 S 0.0 0.0 0:00.00 /bin/sh -c java -jar classifierDump.jar /home/bits/pcapDir/0/
2152M 10744 S 105. 3.3 4:16.00 java -jar classifierDump.jar /home/bits/pcapDir/0/

```

Figure 7: A snapshot of the P2P classification module invoked in parallel

Acknowledgments

This work was supported by Grant number 12(13)/2012-ESD for scientific research under Cyber Security area, e-Security division, from the Department of Electronics & Information Technology (DeitY), Govt. of India, New Delhi, India.

Credits

IDPT was developed by the NetClique research group at BITS–Pilani, Hyderabad campus, Hyderabad, India. This implementation is a part of a research project funded by DeitY, Govt. of India.

Contributors:

- Prof. Chittaranjan Hota (Chief Investigator)
- Abhishek Thakur (Co-investigator)
- D. Jagan Mohan Reddy (Research scholar)
- Pratik Narang (Research scholar)