



About ENISA

The European Union Agency for Network and Information Security (ENISA) is a centre of network and information security expertise for the EU, its member states, the private sector and Europe's citizens. ENISA works with these groups to develop advice and recommendations on good practice in information security. It assists EU member states in implementing relevant EU legislation and works to improve the resilience of Europe's critical information infrastructure and networks. ENISA seeks to enhance existing expertise in EU member states by supporting the development of cross-border communities committed to improving network and information security throughout the EU. More information about ENISA and its work can be found at www.enisa.europa.eu.

Acknowledgements

Contributors to this report

We would like to thank all our ENISA colleagues who contributed with their input to this report and supervised its completion, especially Lauri Palkmets, Cosmin Ciobanu, Andreas Sfakianakis, Romain Bourgue, and Yonas Leguesse. We would also like to thank the team of Don Stikvoort and Michael Potter from S-CURE, The Netherlands, Mirosław Maj and Tomasz Chlebowski from ComCERT, Poland, and Mirko Wollenberg from PRESECURE Consulting, Germany, who produced the second version of this documents as consultants.

Agreements or Acknowledgements

ENISA wants to thank all institutions and persons who contributed to this document. A special 'Thank You' goes to the following contributors: Anna Felkner, Tomasz Grudzicki, Przemysław Jaroszewski, Piotr Kijewski, Mirosław Maj, Marcin Mielniczek, Elżbieta Nowicka, Cezary Rzewuski, Krzysztof Silicki, Rafał Tarłowski from NASK/CERT Polska, who produced the first version of this document as consultants and the countless people who reviewed this document.

Contact

For contacting the authors please use CERT-Relations@enisa.europa.eu

For media enquires about this paper, please use press@enisa.europa.eu.



Legal notice

Notice must be taken that this publication represents the views and interpretations of the authors and editors, unless stated otherwise. This publication should not be construed to be a legal action of ENISA or the ENISA bodies unless adopted pursuant to the Regulation (EU) No 526/2013. This publication does not necessarily represent state-of-the-art and ENISA may update it from time to time.

Third-party sources are quoted as appropriate. ENISA is not responsible for the content of the external sources including external websites referenced in this publication.

This publication is intended for information purposes only. It must be accessible free of charge. Neither ENISA nor any person acting on its behalf is responsible for the use that might be made of the information contained in this publication.

Copyright Notice

© European Union Agency for Network and Information Security (ENISA), 2013

Reproduction is authorised provided the source is acknowledged.



Table of Contents

1	Introduction	1
2	General Description	1
3	EXERCISE COURSE	1
3.1	Introduction to the exercise	1
3.2	Keys to the exercise	4
3.2.1	Task 1 Setting up AbuseHelper	4
3.2.2	Task 2 Working with AbuseHelper	11
4	Summary of the exercise	19
5	REFERENCES	19

1 Introduction

Goal

Setting up and working with AbuseHelper.

Target audience

Technical and management CERT staff

Course Duration

4 hours

Frequency

Once per team

Structure of this document

	Task	Duration
	Introduction to the exercise	30 min
	Task 1: Setting up AbuseHelper	60 min
	Task 2: Working with AbuseHelper	120 min
	Summary of the exercise	30 min

2 General Description

During this exercise, the AbuseHelper application will be set up and used to consolidate diverse incident relevant information feeds. The participants will learn how to install, maintain and work with the tool.

3 EXERCISE COURSE

The course of this exercise is as follows. All discussions should be moderated by the trainer.

3.1 Introduction to the exercise

The introduction should consist of the following points:

- definition of proactive incident detection;
- introduction and presentation of information feeds;
- explanation of technology used in AbuseHelper (XMPP).

Definition of proactive incident detection

Proactive detection of incidents is the process of discovery of malicious activity in a CERT's constituency through internal monitoring tools or external services that publish information about detected incidents, **before the affected constituents become aware of the problem**. It can be viewed as a form of early warning service from the constituents' perspective. Effective proactive detection of network security incidents is one of the cornerstones of an efficient CERT service portfolio capability. It can greatly enhance a CERT's operations, improve its situational awareness and enable it to handle incidents more efficiently, thus strengthening the CERT's incident handling capability, which is one of the core services of national/governmental CERTs.¹

Introduction and presentation of information feeds

Most external services offer incident data in the form of IP addresses, URLs, domains or malware associated with a particular malicious activity, such as a bot, C&C server, malicious URL or scanning. Sometimes more sensitive data are offered, such as stolen user credentials or credit card data. Some services may also offer alerts in more abstract forms depending on detection models used internally in the service.²

Agents for these feeds are included with the AbuseHelper distribution:

- DShield http://www.dshield.org/feeds_doc.html
- Shadowserver <http://www.shadowserver.org/wiki/>
- AbuseCH <https://www.abuse.ch/>
- ARF <https://tools.ietf.org/html/rfc5965>
- Logfiles (Syslog)
- Arbor Networks <http://atlas.arbor.net/>
- CleanMX <http://www.clean-mx.de/>
- Dragon research Group <http://www.dragonresearchgroup.org/>
- IP lists (IPv4, IPv6)
- Malc0de <http://malc0de.com/database/>
- Malware Blacklist <http://www.malwareblacklist.com/>
- Malware Domainlist <http://www.malwaredomains.com/>
- Open BL <http://www.openbl.org/>
- PhishTank <https://www.phishtank.com/>
- Project Honeypot <https://www.projecthoneypot.org/>
- RSS <https://en.wikipedia.org/wiki/RSS>
- RTIR <http://bestpractical.com/rtir/>
- Spamhaus <http://www.spamhaus.org/>
- Whois <https://en.wikipedia.org/wiki/Whois>
- Zone-H <https://www.zone-h.org/>

¹ ENISA, Baseline Capabilities of National / Governmental CERTs, Part 2:

<http://www.enisa.europa.eu/act/cert/support/files/baseline-capabilities-of-national-governmental-certs-policy-recommendations>

² Proactive detection of network security incidents

<https://www.enisa.europa.eu/activities/cert/support/proactive-detection>

Service	Timeliness	Accuracy of results	Ease of use	Coverage	Resources required
DNS-BH Malware Domain Blocklist	Fair	Good	Excellent	Excellent	Excellent
MalwareURL	Good	Good	Excellent	Excellent	Excellent
DSHIELD	Excellent	Fair	Good	Excellent	Excellent
Google Safe Browsing Alerts	Good	Fair	Good	Excellent	Good
HoneySpider Network (as a service)	Excellent	Fair	Good	Fair	Excellent
AusCERT	Good	Good	Good	Good	Excellent
Cert.br data feed	Good	Good	Fair	Good	Good
FIRE	Good	Good	Fair	Good	Good
Team Cymru – TC Console	Excellent	Good	Good	Excellent	Excellent
EXPOSURE	Good	Good	Excellent	Good	Excellent
AmaDa	Excellent	Good	Excellent	Fair	Excellent
Malware Domain List	Excellent	Good	Excellent	Good	Excellent
Zeus/SpyEye Tracker	Good	Excellent	Excellent	Fair/Good	Excellent
The Spamhaus Project Datafeed	Excellent	Good	Good	Excellent	Good
Shadowserver Foundation	Good	Good	Excellent	Good/Excellent	Excellent
SGNET	Good	Excellent	Good	Fair	Good
ARAKIS	Good	Good	Excellent	Good	Excellent
Malc0de database	Excellent	Good	Excellent	N/A	Excellent
ParetoLogic URL Clearing House	Excellent	Good	Good	N/A	Good
SpamCop	Excellent	Good	Good	Excellent	Good
Arbor ATLAS	Good	Good	Excellent	Excellent	Excellent
CBL (Composite Blocking List)	Excellent	Excellent	Fair/Good	Excellent	Good
Cert.br Spampots	Excellent	N/A	Good	Fair	Fair
Team Cymru's CAP	Good	Excellent	Excellent	Excellent	Good
Project Honeypot	Good	Good	Excellent	Excellent	Good/Excellent
Malware Threat Center	Good	Fair	Excellent	Fair	Good
Smart Network Data Services	Good	Good	Excellent	Excellent	Good
Malware Patrol	Excellent	N/A	Excellent	N/A	Excellent
Zone-H	Excellent	Excellent	Good	Good	Fair-Excellent
Cisco IronPort SenderBase	Excellent	Good/Excellent	Excellent	Excellent	Good

Figure 1: Evaluation of services/feeds found in ENISA study 'Proactive detection of network security incidents'

Explanation of technology used in AbuseHelper (XMPP)

The software was developed to provide a framework on which incident response teams can build their own handling systems and processes. It is developed under the terms of the MIT license³ to let teams take part in the progress of the system.

The development is led by the company Clarified Networks Oy,⁴ which specializes in software products for analysing and visualising incidents. They provide commercial support, training and consulting in regards to AbuseHelper and other products (see summary section below). It is also possible to file issues via the Bitbucket site.⁵

AbuseHelper is written in Python and developed relying on XMPP protocol (not mandatory) and agents. The base principle is to control an agent via a central chat room where all bots are listening. Agents are exchanging information in subrooms. AbuseHelper is then scalable and each agent follows a KISS (*Keep it Simple, Stupid*) approach. Each user is able to produce the perfect workflow for his business. The user just needs to take the agents he needs and connect them together.

³ <http://www.opensource.org/licenses/mit-license.php>

⁴ <https://www.clarifiednetworks.com/FrontPage>

⁵ <https://bitbucket.org/clarifiednetworks/abusehelper/issues?status=new&status=open>



Figure 2: AbuseHelper workflow structure (<https://www.cert.be/files/demolayout3.png>)⁶

In the references section and in the folder on the Virtual Image (references /usr/share/trainer/14_PID), additional material is provided, which will help you in addressing this part of the exercise.

3.2 Keys to the exercise

3.2.1 Task 1 Setting up AbuseHelper

In this task, the students will set up their own instance of AbuseHelper. They will learn which issues might appear, which components are important for running the software and how to find additional information.

The software provides the collected information through different means. You can connect via a XMPP client and view the data manually or filter and write it to almost any output format you would need (like a trouble ticket system). You can use AbuseHelper to send alert mails or let it write information to files.

⁶ <https://www.cert.be/pro/abusehelper>

The Virtual Image has been prepared with all necessary tools, libraries and sources to install and use AbuseHelper. Please be aware that an Internet connection is mandatory to make use of the live feeds.

These are the steps that are necessary to install and run AbuseHelper on the Virtual Image.

1. Ejabber Daemon⁷

```
sudo /etc/init.d/ejabberd start # Start the Jabber service
sudo ejabberdctl register abusehel localhost exercise # register
a user for the bots (username host password)
sudo ejabberdctl register trainer localhost exercise # register
a user for the trainer (username host password)
sudo ejabberdctl register trainee localhost exercise # register
a user for the students (username host password)
sudo vi /etc/ejabberd/ejabberd.cfg # open the ejabberd
configuration file and edit the following lines
max_user_sessions 100 # maximum sessions for a single user
s2s_default_policy deny # deny server to server communication
%% {shaper, c2s_shaper}, # search for and comment out the
default shaper configuration
```



```
%%% =====
%%% LISTENING PORTS
%%
%% listen: Which ports will ejabberd listen, which service handles it
%% and what options to start it with.
%%
{listen,
 [
  {5222, ejabberd_c2s, [
    {access, c2s},
    %%{shaper, c2s_shaper},
    {max_stanza_size, 65536},
    %%zlib,
    starttls, {certfile, "/etc/ejabberd/ejabberd.pem"}
  ]},
```

Figure 3: ejabberd.cfg shaper configuration

```
{mod_muc, [
  %%{host, "conference.@HOST@"},
  {access, muc},
  {access_create, muc},
  {access_persistent, muc},
  {access_admin, muc_admin},
  {max_users_admin_threshold,
  20}, # add this entry
```

⁷ Ejabberd Installation and Operation Guide <http://www.ejabberd.im/files/doc/guide.html>

```
this entry          {max_user_conferences, 1000}, # add
                    {max_users, 500}
                ]},
```

```
{mod_muc, [
    %%{host, "conference.@HOST@"},
    {access, muc},
    {access_create, muc},
    {access_persistent, muc},
    {access_admin, muc_admin},
    {max_users_admin_threshold, 20},
    {max_user_conferences, 1000},
    {max_users, 500}
]},
```

Figure 4: ejabberd.cfg shaper configuration

```
sudo /etc/init.d/ejabberd restart # Restart ejabberd server
```

2. AbuseHelper

```
sudo useradd -m abusehel # add a system user for AbuseHelper
sudo mkdir -p /var/lib/ah2 # create the working directory
sudo chown root:abusehel /var/lib/ah2 # ownership of the working
directory
sudo chmod 0750 /var/lib/ah2 # directory access rights set to
read, write
cd /usr/share/trainer/14_PID/adds/abusehelper/ # change your
current directory (trainee for the students)
sudo python setup.py install # run the AbuseHelper setup script
cd /usr/local/lib/python2.7/dist-packages/abusehelper # change
directory
sudo python contrib/configgen/configgen.py /var/lib/ah2/production
# start the configuration script
```

Enter the following information:

```
XMPP username: abusehel@localhost # as defined during user
registration
```

```
XMPP password: exercise # you will be asked to enter this
twice
```

```
XMPP lobby channel: abusehelper # this is the initial channel
to connect to when starting the Jabber client
```

```
Configure mailer? Yes # let AbuseHelper send alert mails
```

```
SMTP host: localhost # use the local MTA for delivery
```

```
SMTP port: 25 # use the standard SMTP port
```

```
SMTP auth user: no auth # no authentication necessary
```

```
Mail sender: abusehelper@localhost # mail sender address
```

```
sudo chown -R root:abusehel /var/lib/ah2/production # access rights have to be corrected after the configuration script
```

```
sudo chmod 0750 /var/lib/ah2/production # see above
```

```
sudo chmod g+w /var/lib/ah2/production/archive # see above
```

```
sudo chown abusehel /var/lib/ah2/production/log # this directory has been added and must be owned by the abusehel system user for logging
```

```
sudo chown abusehel /var/lib/ah2/production/state # see above
```

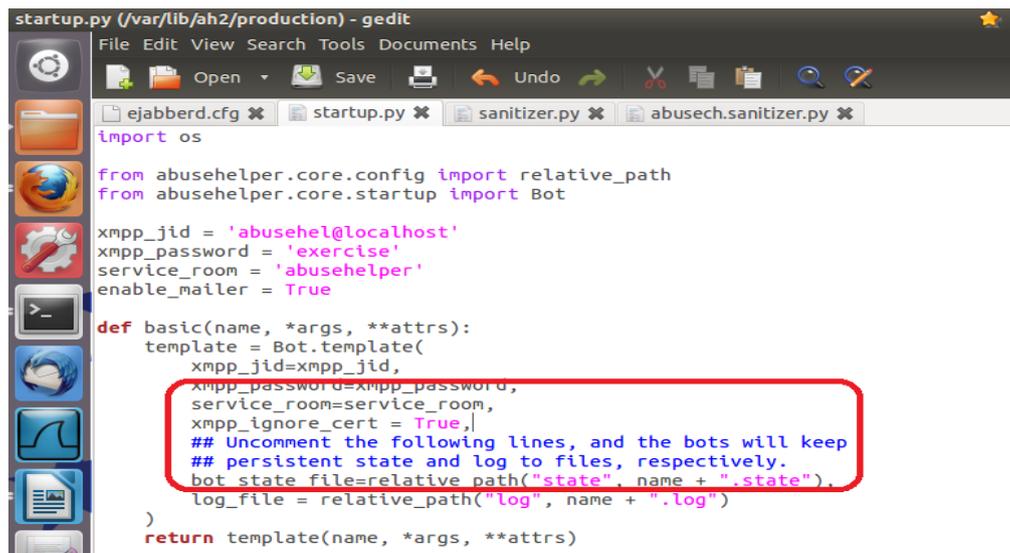
```
sudo vi /var/lib/ah2/production/startup.py # open the startup script and check the entries made by means of the configen script
```

Insert this line after 'service_room=service_room,' in the 'def basic' section:

```
xmpp_ignore_cert = True, # this deactivates checking ssl certificates
```

Comment out the following line in the 'def configs' section:

```
# yield basic("roomgraph")
```



```
startup.py (/var/lib/ah2/production) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
ejabberd.cfg startup.py sanitizer.py abusech.sanitizer.py
import os
from abusehelper.core.config import relative_path
from abusehelper.core.startup import Bot
xmpp_jid = 'abusehel@localhost'
xmpp_password = 'exercise'
service_room = 'abusehelper'
enable_mailer = True
def basic(name, *args, **attrs):
    template = Bot.template(
        xmpp_jid=xmpp_jid,
        xmpp_password=xmpp_password,
        service_room=service_room,
        xmpp_ignore_cert = True,
        ## Uncomment the following lines, and the bots will keep
        ## persistent state and log to files, respectively.
        bot_state file=relative_path("state", name + ".state").
        log_file = relative_path("log", name + ".log")
    )
    return template(name, *args, **attrs)
```

Figure 5: Startup.py

Configure the mail recipient in the runtime.py file:

```
sudo vi /var/lib/ah2/production/runtime.py
```

Change the recipient from someone@example.com to trainer@localhost (or trainee@localhost)

```
def configs():
    # Source definitions

    yield source("dshield",
                asns=[680,24940])

    yield source("abusech")
    yield source("arborssh")
    yield source("sshlog")
    yield source("cymru-rss")

    # Customer definitions

    yield customer("everything-to-mail-at-8-o-clock",
                  rules.ANYTHING(),
                  mail(to=["trainer@localhost"], times=["10:30"]))

    yield customer("asn3-or-netblock",
                  rules.OR(
                      rules.MATCH("asn", "3"),
                      rules.NETBLOCK("127.0.0.1", 16)))

    yield customer("fi-urls",
                  rules.MATCH("url", re.compile(r"^http(s)?://[\w\.-]+\.\fi(\w|$)", re.U | re.I)))

    yield customer("ENISA",
                  rules.MATCH("tag", "ENISA"),
                  mail(to=["trainer@localhost"], times=["10:35"]))
```

Figure 6: Runtime.py

3. Start AbuseHelper

```
sudo su - abusehel -s /bin/bash # change to the abusehel system user
```

```
botnet start /var/lib/ah2/production # start the bots defined in the startup.py script
```

```
botnet status /var/lib/ah2/production # ask for the status, at least one instance should be running
```

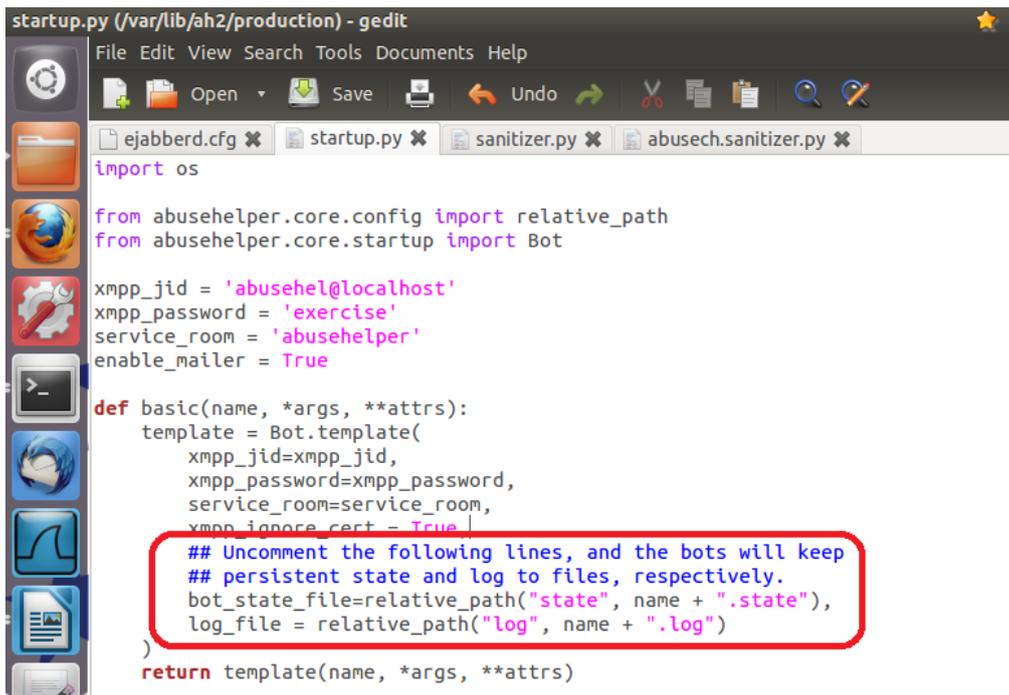
```
botnet stop /var/lib/ah2/production # stop the abusehelper bots
```

Logs can be found in these directories:

```
/var/lib/ah2/production/log/
```

```
/var/log/ejabberd/
```

To enable logging functionality for every bot (logs can be found from /var/lib/ah2/production/log) uncomment the lines outlined below in the picture.



```

startup.py (/var/lib/ah2/production) - gedit
File Edit View Search Tools Documents Help
[ejabberd.cfg] [startup.py] [sanitizer.py] [abusech.sanitizer.py]
import os

from abusehelper.core.config import relative_path
from abusehelper.core.startup import Bot

xmpp_jid = 'abusehel@localhost'
xmpp_password = 'exercise'
service_room = 'abusehelper'
enable_mailer = True

def basic(name, *args, **attrs):
    template = Bot.template(
        xmpp_jid=xmpp_jid,
        xmpp_password=xmpp_password,
        service_room=service_room,
        xmpp_ignore_cert = True
    )
    ## Uncomment the following lines, and the bots will keep
    ## persistent state and log to files, respectively.
    bot_state_file=relative_path("state", name + ".state"),
    log_file = relative_path("log", name + ".log")
    )
    return template(name, *args, **attrs)

```

Figure 7: Startup.py state and log configuration

4. Start Jabber clients

There are several Jabber clients installed; you should at least try the following ones:

- Psi+
Graphical client; you will have to trust the certificate presented by the Jabber service manually.

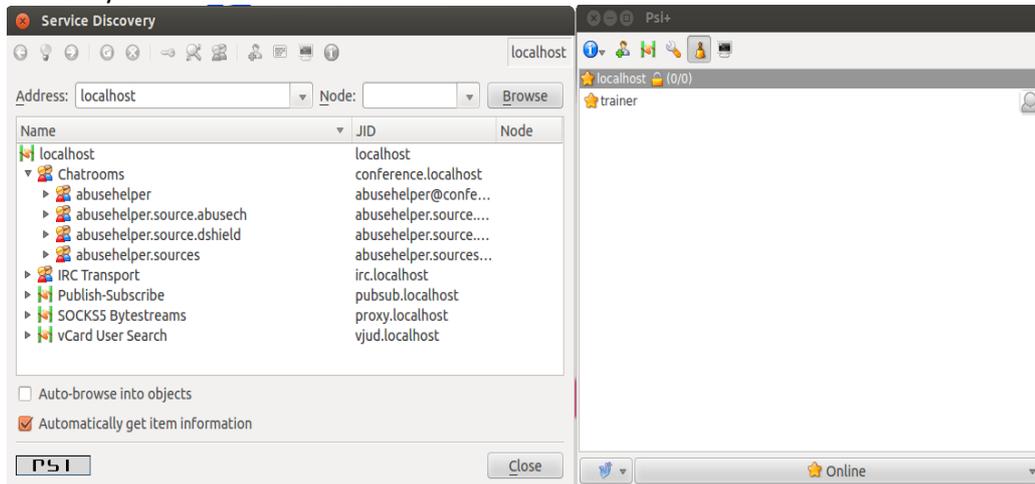


Figure 8: Psi+ initial configuration

This screenshot shows the service discovery feature:

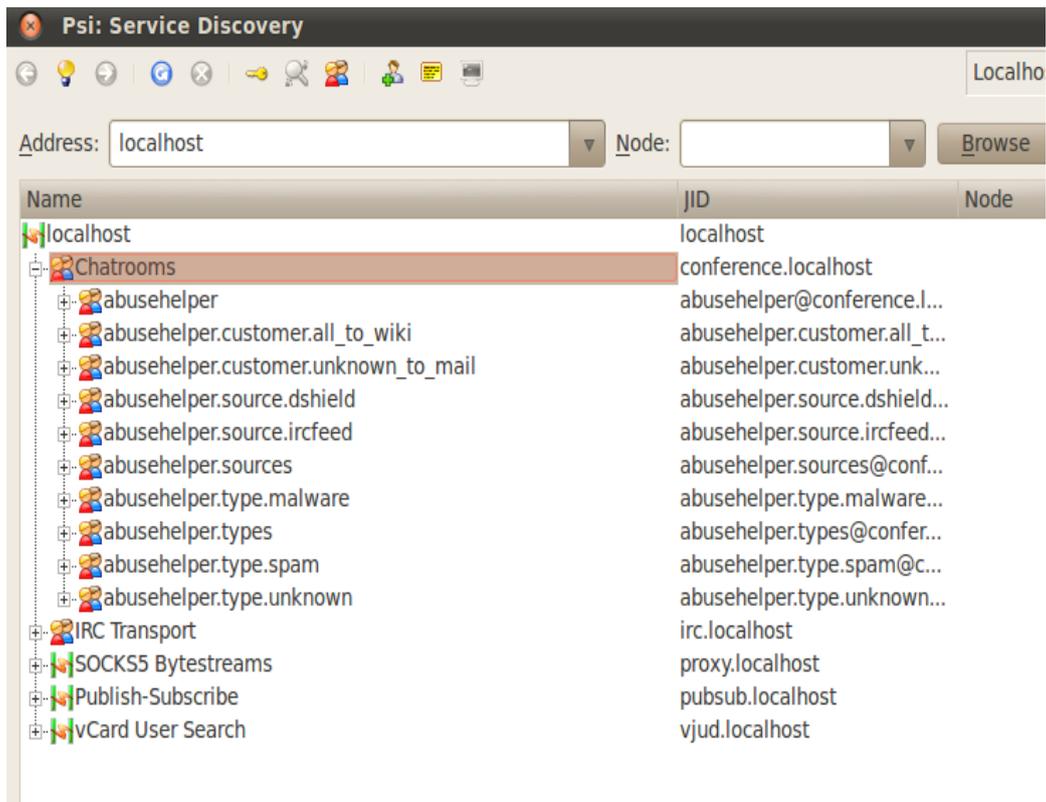


Figure 9: Psi+ service discovery

- Roomreader

Command line client, comes with AbuseHelper

```
roomreader --xmpp-ignore-cert trainer@localhost abusehelper
```

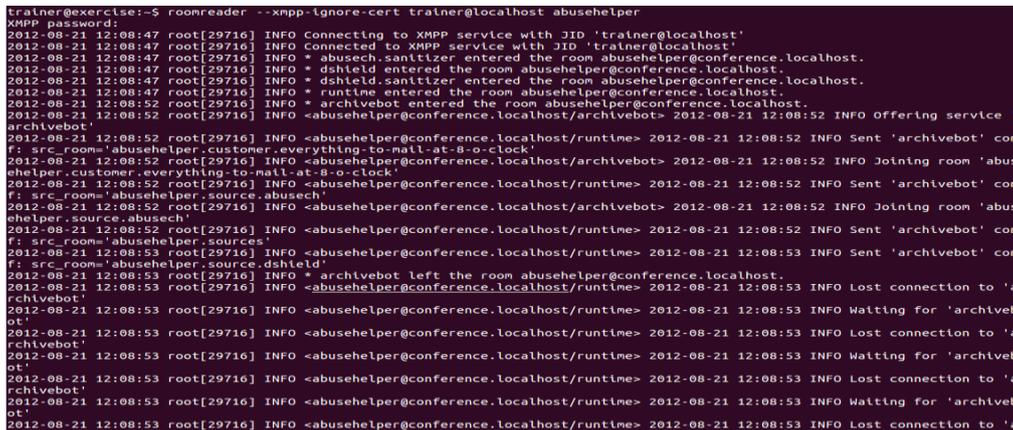


Figure 10: Roomreader

Use these credentials for the clients (used in roomreader application):

XMPP username: trainer@localhost # trainee@localhost for students

XMPP password: exercise

XMPP Room: abusehelper

3.2.2 Task 2 Working with AbuseHelper

This part will contain several subtasks.

1. Making yourself familiar with AbuseHelper

The first step will be to watch the different subrooms and identify the information flow. The table at the end of this section helps to structure and evaluate the learning process. Figures 11 and 12 show screenshots from different data feeds.

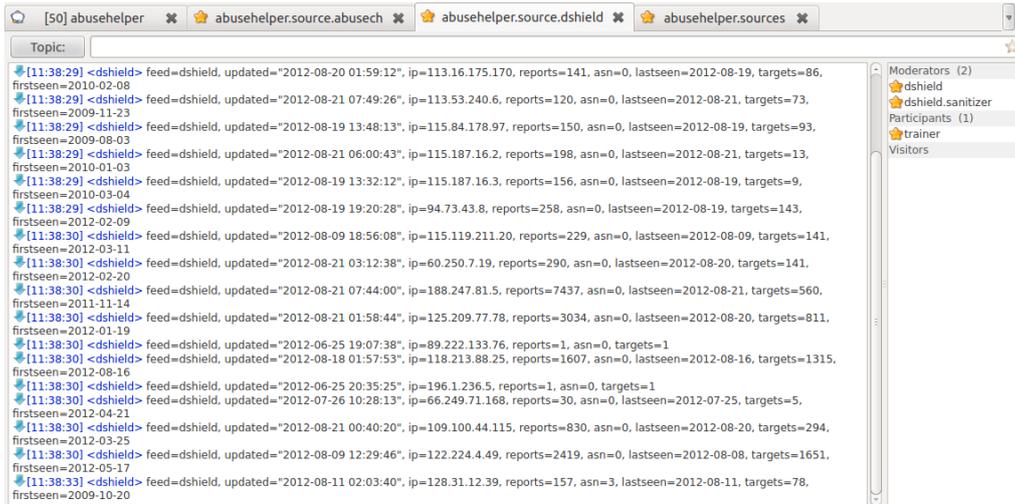


Figure 11: Dshield bot data

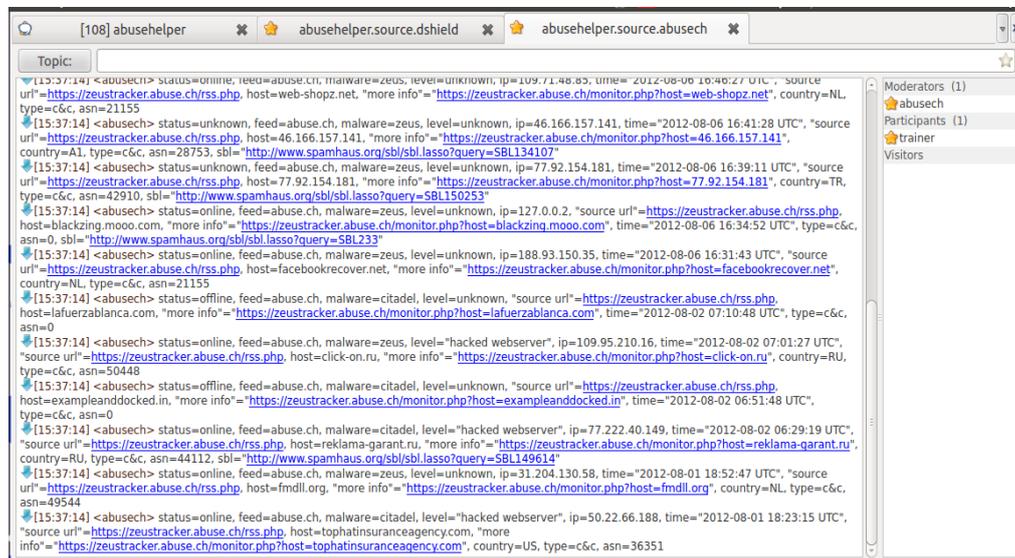


Figure 12: AbuseCH bot data

2. Carry on and include additional data feeds

In this task the students should identify and describe the data feeds in the `/usr/share/lib/python2.7/dist-packages/abusehelper/contrib` section and document which to include. Afterwards they should configure the bots in the `startup.py` and `runtime.py` files (error messages will be logged to the bot files in the

/var/lib/ah2/production/log/ folder):

```
# Launch a nice source bot from the contrib. Remember to explicitly
# define the bot module name, as this is not a core bot!

yield basic("abusech", "abusehelper.contrib.abusech.abusechbot")
yield basic("arborssh", "abusehelper.contrib.arbor.ssh")
yield basic("sshlog", "abusehelper.contrib.sshlogbot.sshlogbot", path="/var/log/auth.log")
yield basic("cymru-rss", "abusehelper.contrib.rssbot.rssbot", feeds="http://www.team-cymru.org/News/secnews.rss")
```

Figure 13: Startup.py bot configuration

```
# Source definitions

yield source("dshield",
    asns=[680,24940])

yield source("abusech")
yield source("arborssh")
yield source("sshlog")
yield source("cymru-rss")
```

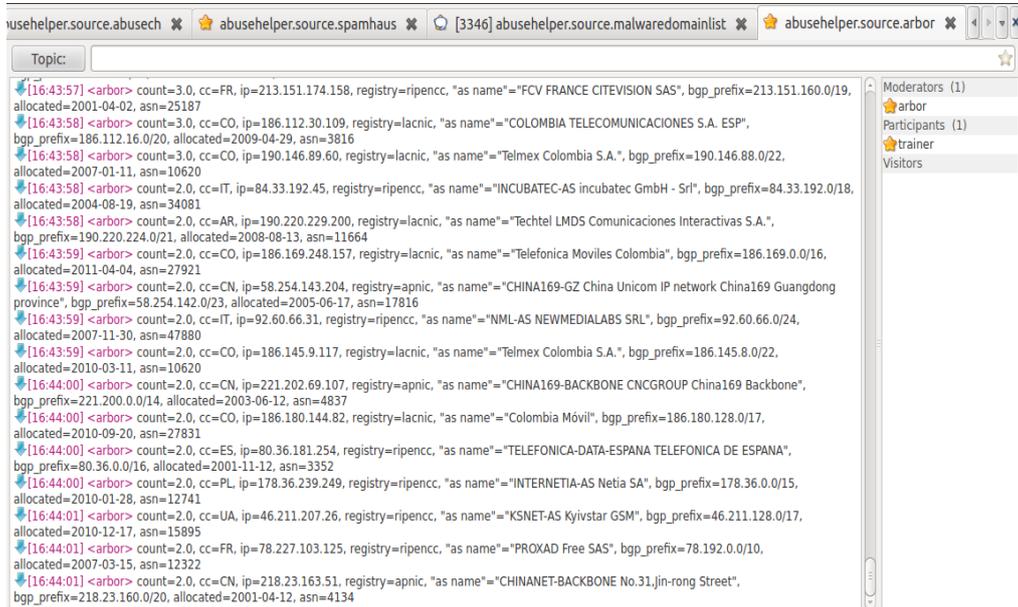
Figure 14: Runtime.py bot configuration

The user account used in order to run the AbuseHelper application is abusehel.

Restart AbuseHelper with commands:

```
botnet stop /var/lib/ah2/production
```

```
botnet start /var/lib/ah2/production
```



The screenshot shows a web browser window with several tabs. The active tab is titled "[3346] abusehelper.source.malwaredomainlist". The main content area displays a list of bot data entries, each starting with a timestamp and a bot name in angle brackets. The entries include details such as count, cc, ip, registry, and as name. The right sidebar shows a list of moderators and participants, including "arbor", "trainer", and "Visitors".

Figure 15: Arbor bot data

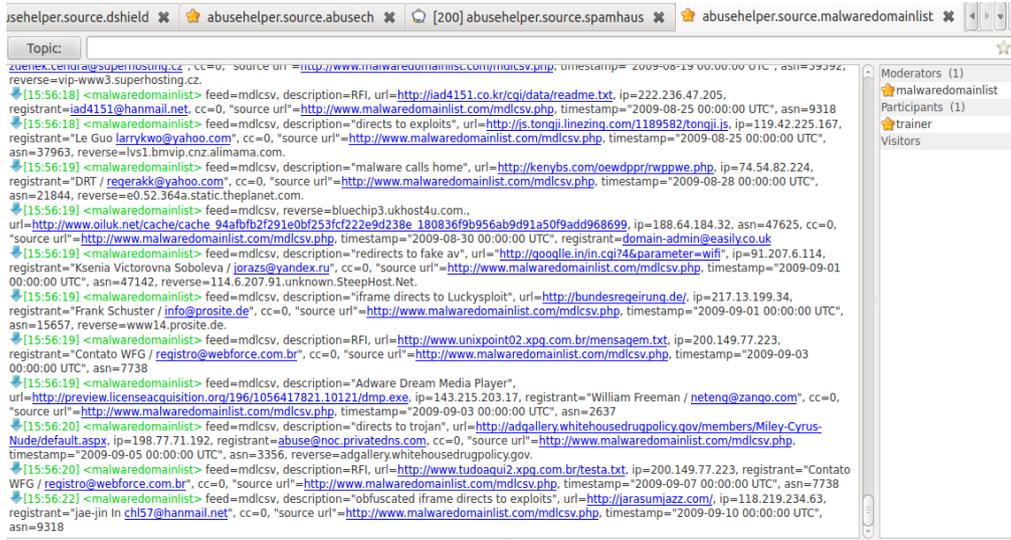


Figure 16: Malwaredomainlist data

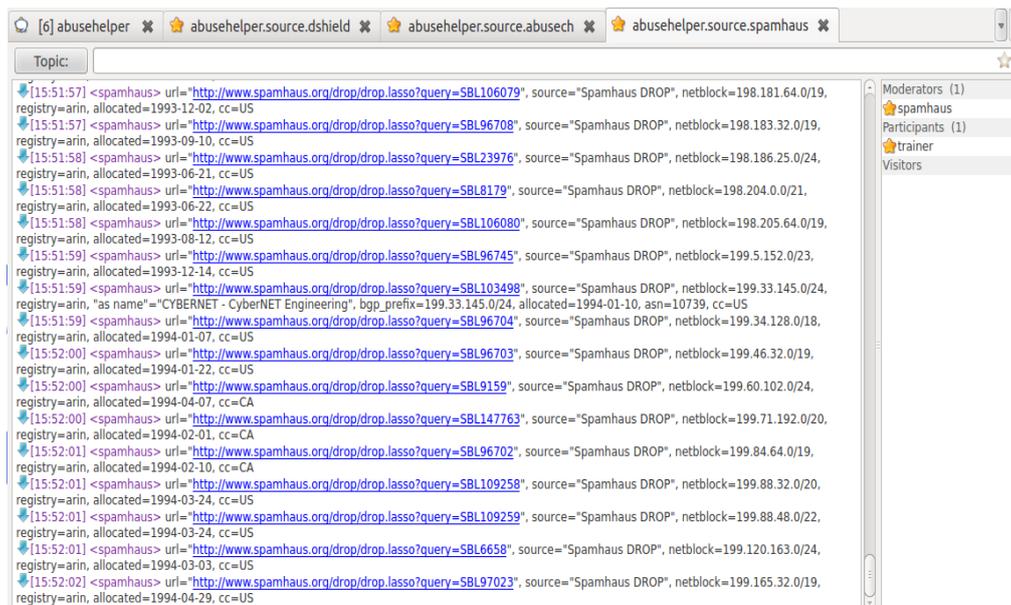


Figure 17: Spamhaus data

3. Filter information feeds

There are different ways to filter the incoming information to be more relevant to your organisations infrastructure. Start with dshield and open the runtime.py. You will find an entry regarding the Autonomous System Numbers (ASN) (see Figure 18). Change the ASN to your organisation's network(s). Edit runtime.py (/var/lib/ah2/production/runtime.py) to filter ASN numbers. In the following screenshot (Figure 18) we configured the ASN 680 and 24940 to fetch exclusively data for these networks. A list linking ASN to organisations can be found [here](#).

```
def configs():
    # Source definitions

    yield source("dshield",
                asns=[680, 24940])

    yield source("abusech")
    yield source("arbor")
    yield source("spamhaus")
```

Figure 18: Runtime ASN config in runtime.py

The functionality of this filter mechanism is implemented in the dshield bot itself.

Sanitizers take the raw data provided by the bots, clean it according to the configuration and deliver it into the abusehelper.sources room. Examples for sanitizer scripts are available in `/var/lib/ah2/production/custom/`. These can be easily adapted for other bots. You can add fields in the sanitizer scripts:

```
from abusehelper.core import events
import sanitizer

class DShieldSanitizer(sanitizer.Sanitizer):
    def sanitize(self, event):
        new = events.Event()

        new.update("ip", event.values("ip", sanitizer.ip))
        if not new.contains("ip"):
            self.log.error("No valid IP for event %r", event)
            return

        new.update("time", event.values("updated", sanitizer.time))
        if not new.contains("time"):
            self.log.error("No valid time for event %r", event)
            return

        new.update("asn", event.values("asn"))

        new.add("tag", "ENISA")

        yield new

if __name__ == "__main__":
    DShieldSanitizer.from_command_line().execute()
```

Figure 19: Dshield sanitizer with tag added

These fields/tags can be used in rules. Let the students create a sanitiser script for one of the bots from contrib and modify it to add the ENISA tag to the output.

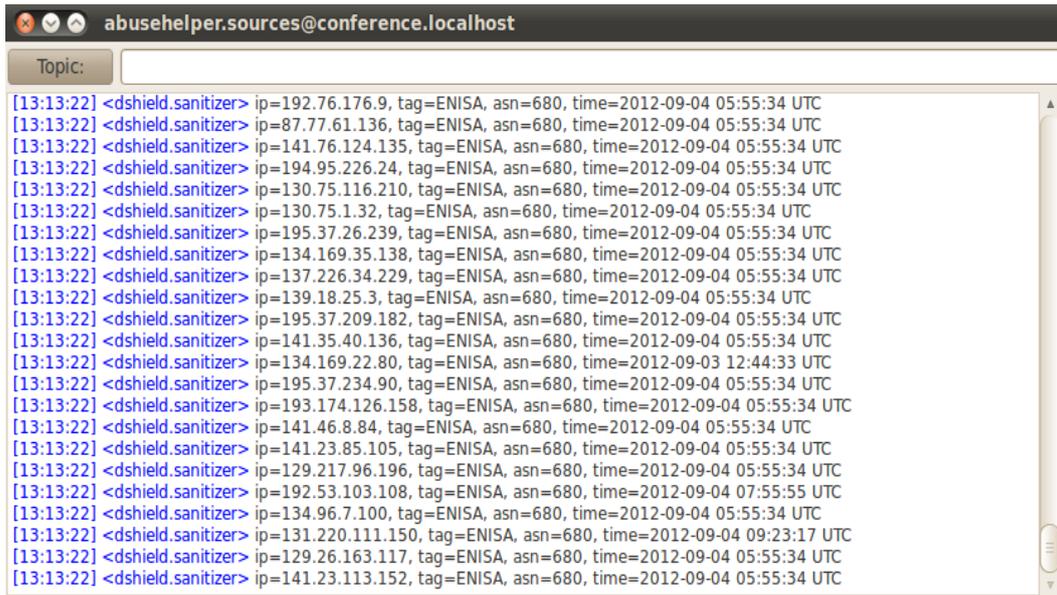


Figure 20: Dshield output with ENISA tag

You can write rules to filter output. First, tweak the `def _mail` section in the `runtime.py` to use `abusehelper.sources` as data input:

```

def _mail(room):
    mail_room = room_prefix + ".sources"
    yield Session("mailer",
        src_room=mail_room,
        to=to,
        cc=cc,
        times=times,
        template=template)
    return _mail

```

Figure 21: Runtime rules in runtime.py

Secondly, add a customer definition:

```

yield customer("ENISA",
    rules.MATCH("tag", "ENISA"),
    mail(to=["trainer@localhost"], times=["10:35"]))

```

Figure 22: Custom rule

In the next screenshot (Figure 23) an example report is shown (you might have to open the mail with the command line mail utility first):

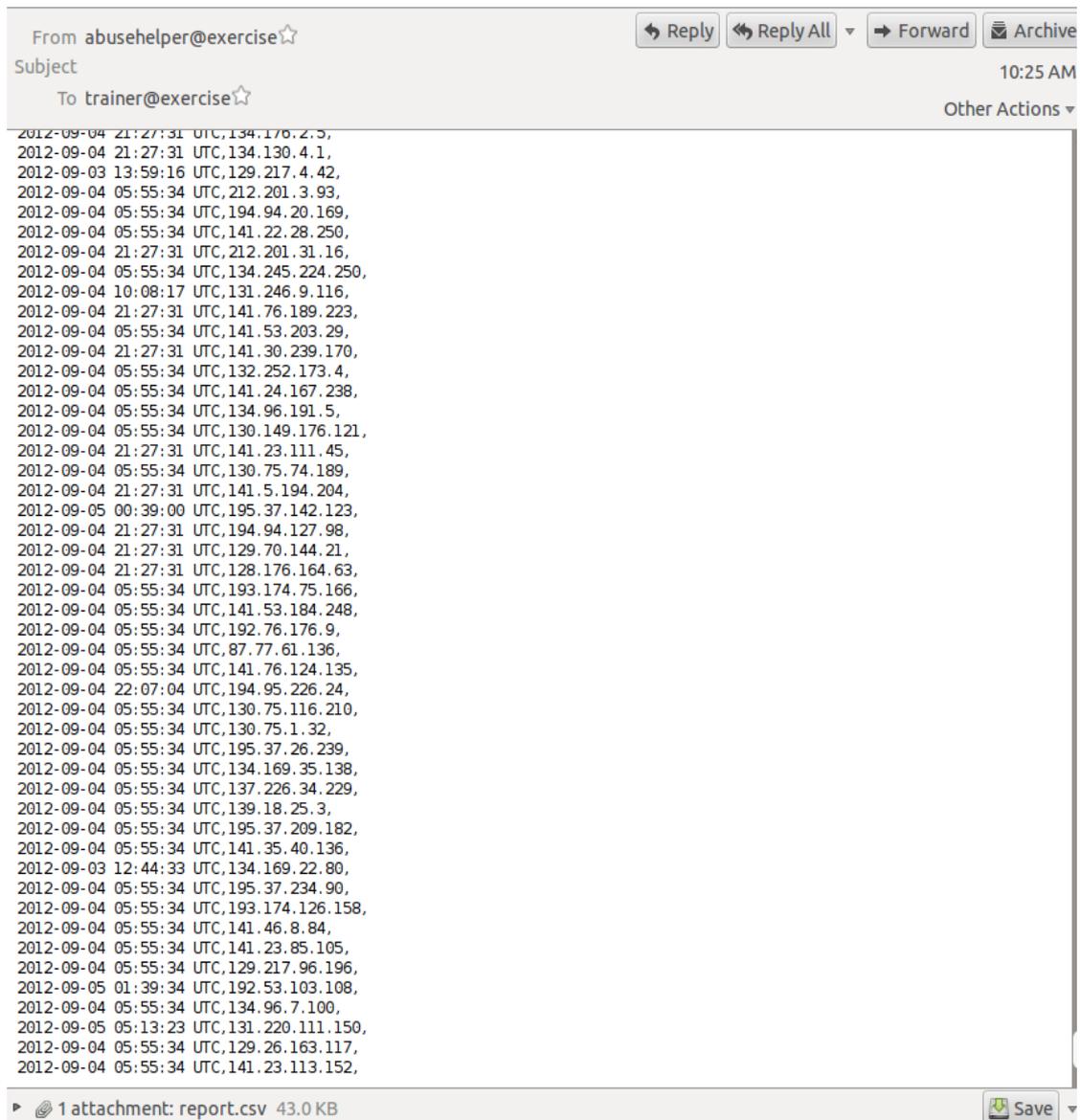


Figure 23: Screenshot of mail with report

Other examples for rules can be found in the BruCon workshop presentation (to be found in the /references folder).

The table below can be used in two ways by the students. First, it helps to structure the learning experience with this exercise by providing tasks that require the students to look at different aspects of the software. Second, it helps in the evaluation of the exercise. You can use the information entered into the table to measure the student's knowledge of AbuseHelper after the exercise. Of course, you are free to add more questions.

No.	Question	Answer
1	Which feeds are standard?	
2	Which information do these deliver?	
3	Where are additional feed bots available?	
4	Integrate the Arbor SSH bot startup.py: yield basic('arborssh','abusehelper.contrib.arbor.ssh') runtime.py: yield source('arborssh')	
5	Integrate the sshlogbot startup.py: yield basic('sshlog', 'abusehelper.contrib.sshlogbot.sshlogbot', path='/var/log/auth.log') runtime.py: yield source('sshlog')	
6	Integrate the Team Cymru RSS feed startup.py: yield basic('cymru rss','abusehelper.contrib.rssbot.rssbot', feeds='http://www.team- cymru.org/News/secnews.rss') runtime.py: yield source('cymru-rss')	
7	Create sanitizer scripts for the included bots (copy from existing abusech.sanitizer.py)	
8	Modify one sanitizer to add tag=ENISA to the output and send corresponding report to localhost mailbox.	
9	Name three bots you find most useful for your work and give reasons for your decision	



4 Summary of the exercise

The summary should contain the following information.

- a) VSRoom:⁸ VSRoom extends the AbuseHelper capabilities with data visualization (in the browser) and workflows for critical service providers and governmental agencies; the source code is provided on [Bitbucket](#);
- b) summary of the table entries;
- c) discussion of issues that occurred during installation/use.
The most important part is a discussion regarding the usefulness in the working procedures of the participating team. The instructor should lead the team to a decision whether to deploy AbuseHelper in their own organization.

5 REFERENCES

1. Cert.be, Abusehelper, 2011
<http://www.abusehelper.be/>
2. Clarified Networks, AbuseHelper, 2012
<https://bitbucket.org/clarifiednetworks/abusehelper>
3. Clarified Networks, AbuseHelper, 2012
<https://www.clarifiednetworks.com/AbuseHelper>

Additional resources are located in the /References directory of this exercise.

⁸ Virtual Situation Room (Clarified VSRoom) - a Situation Awareness System for Critical Infrastructure
<https://www.clarifiednetworks.com/Clarified%20VSRoom>

**ENISA**

European Union Agency for Network and Information Security
Science and Technology Park of Crete (ITE)
Vassilika Vouton, 700 13, Heraklion, Greece

Athens Office

1 Vass. Sofias & Meg. Alexandrou
Marousi 151 24, Athens, Greece



PO Box 1309, 710 01 Heraklion, Greece
Tel: +30 28 14 40 9710
info@enisa.europa.eu
www.enisa.europa.eu