Common Framework for Artifact Analysis Activities

Artifact analysis training material

December 2014
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.

Authors

This document was created by Lauri Palkmets, Cosmin Ciobanu, Yonas Leguesse, and Christos Sidiropoulos in consultation with DFN-CERT Services¹ (Germany), ComCERT² (Poland), and S-CURE³ (The Netherlands).

Contact

For contacting the authors please use cert-relations@enisa.europa.eu
For media enquiries about this paper, please use press@enisa.europa.eu

Acknowledgements

ENISA wants to thank all institutions and persons who contributed to this document.

¹ Klaus Möller, and Mirko Wollenberg
² Miroslaw Maj, Tomasz Chlebowski, Krystian Kochanowski, Dawid Osojca, Pawel Weżgowiec, and Adam Ziaja
³ Michael Potter, Alan Robinson, and Don Stikvoort
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), 2014

Reproduction is authorised provided the source is acknowledged.
Table of Contents

1 Introduction to the exercise  
1.1 Various types of data formats related to malware analysis  
1.2 Standardisation Efforts for Sharing Indicators of Compromise  
  1.2.1 OpenIOC  
  1.2.2 MACCSA (Multinational Alliance for Collaborative for Cyber Situational Awareness)  
  1.2.3 Common data formats (STIX, CybOX, IODEF)  

2 MANTIS  
2.1 Run MANTIS  
2.2 Import data to Mantis  
2.3 Find e-mail addresses  
2.4 Find hashes  
2.5 List files  
2.6 List e-mails  
2.7 Find IP addresses  

3 CRITs  
3.1 Run CRITs  
3.2 Upload binary sample to CRITs  
3.3 Upload E-mail files  
3.4 Upload E-mail with an attachment  

4 Python and common data formats  

5 Bibliography
### Main Objective

In this exercise the student will learn how to collect, store and correlate different types of information about samples and how to make use of this information with the assumption that having a structured and organised database is a good way to reaching synergy in the area of artifact analysis and incident investigation.

Students will learn what kind of data can be collected, what standards are relevant and how they can be used during analysis and investigations.

Students will learn how to design their “Threat Intelligence” environment and what kind of tools they can use.

### Targeted Audience

CERT staff involved in the process of incident handling, especially those responsible for detection of new threats related directly to the CERT customers.

### Total duration

4 hours

### Time Schedule

<table>
<thead>
<tr>
<th>Time Schedule</th>
<th>Description</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction to the exercise</strong></td>
<td></td>
<td>0.5 hour</td>
</tr>
<tr>
<td><strong>Task 1: MANTIS</strong></td>
<td></td>
<td>1.5 hour</td>
</tr>
<tr>
<td><strong>Task 2: CRITs</strong></td>
<td></td>
<td>1.0 hours</td>
</tr>
<tr>
<td><strong>Task 3: Python and common data formats</strong></td>
<td></td>
<td>1.0 hour</td>
</tr>
</tbody>
</table>

### Frequency

Every time a new member joins the team.
1 Introduction to the exercise

Threat and vulnerability information exchange has become one of the most burning issues within the security community. New vulnerabilities in popular software packages are discovered daily and new threats are identified. However, most of them are published in “human readable” formats – as vendors’ web notes, pdf reports, forum posts, and so on. It’s not easy to follow such a stream of unstructured information effectively enough to quickly implement file searches and network traffic patterns into own defence systems such as IDS/IPS or malware scanners.

We are also aware that some of the most notable vendors of network defence and endpoint protection systems focus only on vulnerabilities and threats that have been already addressed by software vendors and where patches are available. This is a sensible practice in terms of security products marketing but a complete failure in terms of information security practice. No 0-day vulnerabilities would be known in the systems ‘protected’ if security managers relied only on these products.

Security systems – IDS/IPS or endpoint protection – often give the possibility of creating own threat and vulnerability definitions to close or at least narrow the gap between threat/vulnerability detection and vendor’s response. Again, however, we encounter the problem of unstructured security information that slows down the implementation of countermeasures and requires much higher skill level from the implementers.

In this exercise we want to show some open formats that gain much attention⁴ and are promoted by US organisations and agencies⁵ to be used by vendors, especially whose products are deployed in critical infrastructure systems⁶.

---


⁵ See [http://csrc.nist.gov/publications/drafts/800-150/sp800_150_draft.pdf](http://csrc.nist.gov/publications/drafts/800-150/sp800_150_draft.pdf) for a draft version of new NIST publication on cyber threat information sharing

1.1 Various types of data formats related to malware analysis

Several practices have emerged in Europe and worldwide that aim at addressing effective information exchange and sharing data about cyber incidents. These efforts can be considered as possible approaches to secure information exchange.

Any piece of information that can be used to search for or identify potentially compromised systems is known as an indicator of compromise (IoC). These IoCs can include IP address/domain name, URL, file hash, email address, X-mailer, HTTP user agent, and file mutex. This information can be compiled into incident reports and enriched with analysis and remediation reports. Several standards exist for formatting information, but there is not a single leading one in place. However, the trend to share structured information rather than unstructured in plain emails can be observed. While, as mentioned, there is currently no single standard for data format that is generally accepted, it is crucial for an automated processing of received information. We provide an overview of existing

standards below, followed by a summary and discussion of known challenges related to automated IoC exchanges.

Multiple initiatives exist, or are currently in development, that aim to address the aforementioned barriers in a systematic way: CERTs still find it difficult to exchange information about (targeted) malware and attacks within a group of trusted partners or by bilateral agreement.

Despite the trend to exchange structured information, much of the information sharing nowadays still occurs through unstructured reports. Where, in order to process the data, it is necessary to manually copy & paste the information into text files that have to be parsed to be exported to (N)IDS and systems or used in log searches.

Some solutions to overcome these problems are being developed by CERTs, NATO, and private organizations, often with the participation of multiple stakeholders. In this document a few of them that enjoy a certain degree of support in the CERT community, which have reached a good level of development, and might address the barriers presented in this report are presented. Adopting these solutions more widely would help CERTs in forming and building larger sharing communities to exchange the benefits of previous detections and remediation efforts. This approach ultimately would lead to more confident and efficient incident response.

### 1.2 Standardisation Efforts for Sharing Indicators of Compromise

In this chapter we present a choice of security information sharing standardisation efforts. A more complete landscape of security information sharing methods – both structured and not can be found in the ENISA ‘Detect, SHARE, Protect’ document

#### 1.2.1 OpenIOC

OpenIOC is an extensible XML schema that enables to describe the technical characteristics of threats, an attacker’s methodology, or other evidence of compromise. Originally, it was designed to enable some commercial products to codify intelligence in order to rapidly search for potential security breaches. In response to requests from across the user community, the company (Mandiant) has standardised and open-sourced the OpenIOC schema to allow communication of threat information at machine speed (meaning automatically). Future versions of OpenIOC will include more flexible indicators and metadata extensions to the IoC (comments, confidentiality, criticality, etc.).

#### 1.2.2 MACCSA (Multinational Alliance for Collaborative for Cyber Situational Awareness)

MACCSA is a continuation of MNE7 (Multinational Experiment 7), which aims to create the conditions to enable the development, implementation, and operation of the Information Sharing Framework (ISF) for Collaborative Cyber Situational Awareness (CCSA).

Organisations targeted by MACCSA include international and multinational bodies such as the EU Military Staff, Europol, NATO, the U.S., countries from Europe and Asia/Pacific, and a number of private companies such as security vendors, operators, industrial companies, and consultancies. The ISF of MACCSA includes two main components: information sharing model and information sharing management. The information sharing model describes the means required for sharing information
 – proactive (push) and reactive (pull) – on alerts and warnings, best practices, security quality management, and for handling proactive artefacts.

Information sharing management focuses on ensuring the quality of the shared information. MACCSA proposes a mesh of hubs and nodes to coordinate information sharing. The model is based on existing federated secure collaboration capabilities in defence, intelligence, and industry, comprising independent entities bound together by information sharing agreements and further united by collaborative and community-centric governance authorities.

1.2.3 Common data formats (STIX, CybOX, IODEF)$^{11}$

TAXII, STIX, and CybOX (all free for public use) are community-driven technical specifications designed to enable automated information sharing for cyber security situational awareness, real-time network defence, and sophisticated threat analysis.

- TAXII™, the Trusted Automated eXchange of Indicator Information
- STIX™, the Structured Threat Information eXpression
- CybOX™, the Cyber Observable eXpression

---

11 Part of this introduction is taken from this original source https://www.us-cert.gov/Information-Sharing-Specifications-Cybersecurity
12 Picture from https://www.us-cert.gov/Information-Sharing-Specifications-Cybersecurity
1.2.3.1 Structured Threat Information Expression (STIX)

Structured Threat Information Expression - STIX\(^{13}\) is a relatively recent collaborative community-driven effort to define and develop a standardised language to represent structured cyber threat information. The STIX Language is intended to convey the full range of potential cyber threat information and strives to be fully expressive, flexible, extensible, automatable, and as human-readable as possible. Any interested party can participate in evolving STIX as part of its open and collaborative community.

![Figure 3. STIX v1.1 Architecture\(^{14}\)](image)

1.2.3.2 Cyber Observable Expression (CybOX)

The Cyber Observable Expression CybOX\(^{15}\) is a standardised schema for the specification, capture, characterisation, and communication of event properties that are observable in the operational domain. A wide variety of high-level cyber security use cases rely on such information, including event management/logging, malware characterisation, intrusion detection, incident response/management, and attack pattern characterisation. CybOX provides a common mechanism (structure and content) for addressing cyber observables across and among this full range of use cases, improving consistency, efficiency, interoperability, and overall situational awareness.

1.2.3.3 Incident Object Description Exchange Format (IODEF)

The Incident Object Description Exchange Format (IODEF) specification (RFC 5070\(^{16}\)) defines a data representation that provides a framework for sharing information commonly exchanged by CERTs

---

\(^{13}\) [http://stix.mitre.org](http://stix.mitre.org)

\(^{14}\) [https://stix.mitre.org/about/documents/STIX_v1.1_Introduction_Briefing.pdf](https://stix.mitre.org/about/documents/STIX_v1.1_Introduction_Briefing.pdf)

\(^{15}\) [http://cybox.mitre.org](http://cybox.mitre.org)

about computer security incidents. It provides an XML representation for conveying incident information across administrative domains between parties that have an operational responsibility for remediation or watch-and-warning over defined constituencies. The data model encodes information about hosts, networks, and the services running on these systems; attack methodology and associated forensic evidence; the impact of the activity; and limited approaches for documenting workflow.
2 MANTIS

The MANTIS (Model-based Analysis of Threat Intelligence Sources) framework consists of several Django\(^\text{17}\) apps that, in combination, support the management of cyber threat intelligence expressed in standards such as STIX, CybOX, OpenIOC, IODEF (RFC 5070), etc.

Unlike previous exercises, MANTIS is already installed in our virtual machine. Installation instructions can be found in appendices at the end of this document.

2.1 Run MANTIS

To start MANTIS type the following commands in a terminal:

```
enisa@enisa:~$ source /home/enisa/mantis/bin/activate
(mantis)enisa@enisa:~$ cd /home/enisa/django-mantis && bash quickstart.sh
```

![Figure 3. MANTIS first run](image)

\(^{17}\) https://www.djangoproject.com/
On the first time after installation the ‘quickstart.sh’ script will ask if you want to create an administrative user for Django. Type ‘yes’ and then enter user name (enisa in this case, it will be the default if run as enisa system user), email address (not essential in this installation, enisa@example.com for instance) and a password. We chose password ‘toor’. Answer ‘yes’ to the question about overwriting static files.

Login and password to your VM installation are login ‘enisa’ and password ‘toor’ as above.

2.2 Import data to Mantis

Now it is time to import some data to our database to search it through. During this exercise we will use some of the samples provided by CybOX Project at https://github.com/CybOXProject/schemas/tree/master/samples.
We will use the file ‘CybOX_Iran-Oil_Dynamic.xml’ from:

https://raw.githubusercontent.com/CybOXProject/schemas/master/samples/CybOX_Iran-Oil_Dynamic.xml

This file contains information about ‘Iran-Oil’ (among many other names used) attack campaign from March 2012 written in CybOX format.

This file is located at `/home/enisa/examples/` directory for your convenience.

To import the data to MANTIS, please write the following commands:

```
enisa@enisa:~/.django-mantis$ source ~/.mantis/bin/activate
(mantis)enisa@enisa:~/.django-mantis$ cd /home/enisa/django-mantis && python manage.py mantis_stix_import --settings=mantis.settings.local \
--trace --marking_json=quickstart_examples/markings/minimal_marking.json \
--marking_pfill=source "Iran-Oil" \
/home/enisa/examples/CybOX_Iran-Oil_Dynamic.xml
```

Login to this interface with username “enisa” and password “toor” (the credentials created in the previous step).

The MANTIS user interface is built around a drop-down menu at the top of the screen. All the following tasks begin with choosing one of the actions from this menu.
2.3 Find e-mail addresses

To find e-mail addresses in the MANTIS database we need to find e-mail messages first. Select ‘Fact Search (simple)’ from the drop down menu:

```
List, Filter & Search → Fact Search (simple)
```

Now type and select the following values in the form that showed up:

- **Value contains**: @
- **InfoObject Type**: cybox.mitre.org:EmailMessageObject
After following these instructions you will see two messages found:

In the ‘Fact-based filtering’ part of the window you will see the search results, while in the ‘Value’ column there will be e-mail addresses. Select one of these from the list by clicking on the ‘Info Object’ element.
2.4 Find hashes

Another useful feature of MANTIS is the ability to search for file hashes. Select the following from the drop down menu:

**List, Filter & Search → Fact Search (simple)**

Type ‘Hash_Value’ in the window:

**Fact term matches:** Hash_Value
Common Framework for Artifact Analysis Activities
Artifact analysis training material

December 2014

Figure 11. Looking for file hashes

After submitting the query you see a list of info objects that contain a file hash:

Figure 12. List of Info Objects containing hashes
Again the ‘Fact-based filtering’ part of the window presents the search results, hashes along with search conditions. The first column contains the filename for each hash. Select one of them by clicking on one of the hyperlinks in the ‘Info Object’ column.

![Figure 13. File details about “Iran’s Oil and Nuclear Situation.doc” file](image)

Now you can see the file details. The ‘Facts’ section contains information like a description, file name, hash and size. On the right-hand side of the window there is information about related objects.

### 2.5 List files

In order to get the list of files, select the following from the drop-down menu and select corresponding Info Object type in the window:

- **Saved Filters/Searches → Filter for STIX Packages**
- **InfoObject Type**: cybox.mitre.org:FileObject
Figure 14. Filtering for files

After submitting the query you will see a list of file object:

Figure 15. List of Info Objects

In the ‘List of Info Objects’ section you can find the list of all files, in the ‘Name’ column there are file names and the number of facts related to each of the files. Select the ‘test.mp4’ file from the list by clicking its ‘Identifier’ element.
Figure 15. Details about “test.mp4” file

Among the facts you can find the information that this file was downloaded by ‘Iran’s Oil and Nuclear Situation.doc’ from ‘http://208.115.230.76/test.mp4’.

Figure 16. File "test.mp4" details
The description of this file reads ‘This mp4 file causes memory corruption and code execution via heap-spraying code injection’ and you can infer from the information in the right-hand side box that this file was included in an exploit running the payload ‘us.exe’.

2.6 List e-mails

To list the e-mails stored in the database, select the following from the drop-down menu and select the e-mail message object type:

Saved Filters/Searches → Filter for STIX Packages

**InfoObject Type:** cybox.mitre.org:EmailMessageObject

![Filter Parameters](image)

Figure 17. Filter Parameters for e-mail messages

After submitting the query you will see list of e-mails shown in the ‘List of Info Objects’ section:
Figure 18. List of Info Objects

Select the first (and only) one from the list by clicking the element in the ‘Identifier’ column.

Figure 19. E-mail “Iran’s Oil and Nuclear Situation” details

On the left-hand side of the window you will see standard e-mail details like e-mail addresses, subject, attachments etc.
2.7 Find IP addresses

To find IP addresses in the database you need to select item from drop down menu in the right hand corner of the page:

List, Filter & Search → Fact Search (simple)

Then you need to type the following value in the form on the right side of the window:

**Fact term matches:** Properties/Address_Value

![Filter Parameters](image)

Figure 20. Filter Parameters

After following these instructions you will get IP addresses along with more information (type of dependency between the property and its value – ‘Equals’ in this example):
Fact-based filtering

Select one of the address values from the list by clicking the ‘Info Object’ element in the first row:

Info Object: AddressObject (3 facts)

The right-hand side of the window presents objects related to this address. This IP address is related to the domain ‘ftp.documents.myPicture.info’. Click on the hyperlink inside the red rectangle to see more facts about this domain.
As you can see the ‘ftp.documents.myPicture.info’ domain name is related to the file ‘us.exe’. Click on the hyperlink under that file name.
You can see all the details about the ‘us.exe’ file stored in the database. Along with file description you can find related object that possibly contained this file — other files (‘test.mp4’ for instance), domain names and e-mail document. Now, click the ‘Observable’ link:
Figure 25. ‘us.exe’ observables

From the description part we get the information that file “us.exe” is apparently a piece of malicious code creating the ‘Iran-Oil.exe’ Trojan file, and from the related objects we deduce the domain analysed previously served as its C&C server address.
3 CRITs

CRITs\(^{18}\) is a web-based tool which combines an analytic engine with a cyber-threat database that not only serves as a repository for attack data and malware, but it also provides analysts with a powerful platform for conducting malware analyses, correlating malware, and for targeting data. These analyses and correlations can also be saved and exploited within CRITs. CRITs employs a simple but very useful hierarchy to structure cyber threat information. This structure gives analyst the power to 'pivot' on metadata to discover previously unknown related content.

CRITs software has been already installed in the VM for your convenience, you can find installation instructions in the appendices at the end of this document.

3.1 Run CRITs

To run CRITs type the following commands in the terminal:

```
su crits
cd /data/crits/ contrib/mongo/UMA && sudo ./mongod_start.sh
cd /data/crits/ && /usr/bin/python manage.py runserver 127.0.0.1:8080
```

![Figure 26. CRITs server running](image)

VM password to “crits” user is “toor”.

CRITs web interface is available at [http://127.0.0.1:8080](http://127.0.0.1:8080). Log into this interface using username: ‘enis\a’ and password: ‘Enisa11’.

\(^{18}\) [http://crits.github.io/](http://crits.github.io/)
3.2 Upload binary sample to CRITs

As an example we will use the “putty.exe” binary from Putty\(^\text{19}\) (which is not malware).

This file is located ‘/home/enisa/examples/’ directory.

To upload this sample select from the menu on to the left of the screen:

Samples → Add Sample → View Sample

Choose the ‘raw’ file format just below file selection button.

![Figure 27. CRITs New Sample](image)

After submitting new sample you can see the file details:

![Figure 28. CRITs File Details](image)

From the top menu select:

Tools → Strings

\(^\text{19}\) [http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe](http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe)
This is the equivalent to the standard Unix ‘strings’ tool. Tools like ‘strings’ are commonly used in first and fast parts of binary files analyses.

### 3.3 Upload E-mail files

To upload an e-mail file, for example in EML format (you can also use Outlook, YAML, Raw) select from the menu:

**Emails → New Email (EML) → View email**

A sample EML file has been prepared on this VM: ‘/home/enisa/examples/test.eml’
After uploading you can see e-mail details:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>541ab503f6e52ef3018408c</td>
</tr>
<tr>
<td>From</td>
<td>Anonymous User <a href="mailto:user@alpha.example.com">user@alpha.example.com</a></td>
</tr>
<tr>
<td>To</td>
<td><a href="mailto:office@beta.example.com">office@beta.example.com</a>,</td>
</tr>
<tr>
<td>CC</td>
<td>Click pencil to edit...</td>
</tr>
<tr>
<td>Date</td>
<td>Tue, 26 Aug 2014 12:07:14 +0200</td>
</tr>
<tr>
<td>Subject</td>
<td>test abc</td>
</tr>
<tr>
<td>Message ID</td>
<td><a href="mailto:53FCC52.5020595@example.com">53FCC52.5020595@example.com</a></td>
</tr>
<tr>
<td>Reply To</td>
<td>None</td>
</tr>
<tr>
<td>File Details</td>
<td></td>
</tr>
</tbody>
</table>

Figure 32. CRITs e-mail details

Now you can view the imported EML file also in CybOX format after select “CybOX View” from the top menu.

Figure 32. CRITs e-mail details in CybOX format

You can now easily use the contents from the ‘CybOX View’ in any other tool supporting the CybOX format.

3.4 Upload E-mail with an attachment

To upload an e-mail with an attachment in the EML format use the same menu item as for a plain e-mail in previous task:

Emails → New Email (EML) → View email

Again, a sample EML file has been prepared on this VM: “/home/enisa/examples/test123.eml”.

After uploading it you will see e-mail details:
Figure 33. CRITs e-mail with attachment

Click the filename in the relationships box of this e-mail to get details about this attachment:

Figure 34. CRITs attachment details

This attachment is a RAR archive, to unpack it you can use a tool built into CRITs – just select ‘Unrar’ tab.
Our sample archive is protected with password, and the password is ‘infected’ (the archive is clean however).

After unpacking the RAR archive you will see relationships like the file origin (extracted from mail message).
After selecting the binary you will see also relationship to the archive it was extracted from.

4  Python and common data formats

The formats we are discussing here are based on the XML (eXtensible Markup Language), today’s standard in defining structured documents. Because of this simple fact you can easily play with these formats using standard XML libraries available for virtually all modern programming languages. As an example we will use one of the most popular - Python.

All these examples are stored on the VM in ‘/home/enisa/examples/’ directory.

```python
#!/usr/bin/env python
#
# -*- coding: utf-8 -*-

from lxml import etree  # http://lxml.de/xpathxslt.html#the-xpath-method

namespaces = {
    'xsi': 'http://www.w3.org/2001/XMLSchema-instance',
    'stix': 'http://stix.mitre.org/stix-1',
    'stixVocabs': 'http://stix.mitre.org/default_vocularies-1',
    'stixCommon': 'http://stix.mitre.org/common-1',
    'cybox': 'http://cybox.mitre.org/cybox-2',
    'cyboxCommon': 'http://cybox.mitre.org/common-2',
    'cyboxVocabs': 'http://cybox.mitre.org/default_vocularies-2',
    'indicator': 'http://stix.mitre.org/Indicator-2',
    'ttp': 'http://stix.mitre.org/ttp-1',
    'marking': 'http://data-marking.mitre.org/Marking-1',
    'simpleMarking': 'http://data-marking.mitre.org/extensions/MarkingStructure#Simple-1',
    'openiocTM': 'http://stix.mitre.org/extensions/TestMechanism#OpenIOC2010-1',
}```
This sample script opens “Appendix_G_IOCs_Full.xml” file from the STIX examples http://stix.mitre.org/downloads/APT1-STIX.zip.

After parsing it with namespaces and XPATH it prints all hashes from this file on the screen in a loop.
You can also query services like VirusTotal with these hashes. You will need VirusTotal API key. To request this key login to the VirusTotal service and select ‘My API key’ from the menu as shown below:

You can query services like VirusTotal with these hashes. You will need VirusTotal API key. To request this key login to the VirusTotal service and select ‘My API key’ from the menu as shown below:

Sample python script sending hashes to the VT:

```python
#!/usr/bin/env python
#
# -*- coding: utf-8 -*-

from lxml import etree # http://lxml.de/xpathxslt.html#the-xpath-method
```
import simplejson
import urllib
import urllib2
import time

namespaces = {
    'xsi': 'http://www.w3.org/2001/XMLSchema-instance',
    'stix': 'http://stix.mitre.org/stix-1',
    'stixVocabs': 'http://stix.mitre.org/default_vocabularies-1',
    'stixCommon': 'http://stix.mitre.org/common-1',
    'cybox': 'http://cybox.mitre.org/cybox-2',
    'cyboxCommon': 'http://cybox.mitre.org/common-2',
    'cyboxVocabs': 'http://cybox.mitre.org/default_vocabularies-2',
    'indicator': 'http://stix.mitre.org/Indicator-2',
    'ttp': 'http://stix.mitre.org/TTP-1',
    'marking': 'http://data-marking.mitre.org/Marking-1',
    'simpleMarking': 'http://data-marking.mitre.org/extensions/MarkingStructure#Simple-1',
    'openiocTM': 'http://stix.mitre.org/extensions/TestMechanism#OpenIOC2010-1',
    'mandiant': 'http://www.mandiant.com',
    'FileObj': 'http://cybox.mitre.org/objects#FileObject-2',
    'WinServiceObj': 'http://cybox.mitre.org/objects#WinServiceObject-2',
    'WinProcessObj': 'http://cybox.mitre.org/objects#WinProcessObject-2',
    'WinExecutableFileObj': 'http://cybox.mitre.org/objects#WinExecutableFileObject-2',
    'WinRegistryKeyObj': 'http://cybox.mitre.org/objects#WinRegistryKeyObject-2',
    'WinHandleObj': 'http://cybox.mitre.org/objects#WinHandleObject-2',
    'ProcessObj': 'http://cybox.mitre.org/objects#ProcessObject-2',
    'WinDriverObj': 'http://cybox.mitre.org/objects#WinDriverObject-2'
}

url = "https://www.virustotal.com/vtapi/v2/file/report"

f = 'Appendix_G_IOCs_Full.xml' # http://stix.mitre.org/downloads/APT1-STIX.zip

doc = etree.parse(f)

    print r.text
    parameters = {"resource": r.text, "apikey": "XXXXXXXXXX"} # VirusTotal API Key
    data = urllib.urlencode(parameters)
Common Framework for Artifact Analysis Activities
Artifact analysis training material

December 2014

```python
req = urllib2.Request(url, data)
response = urllib2.urlopen(req)
json = response.read()
print json

time.sleep(15)  # VirusTotal API request rate - 4 requests/minute
```

After running the above script you will get output like (without colour distinctions):

```python
$ python cybox_xpath-virustotal.py
b305b543da332a2cf6e1ce55ed2ea79
{"response_code": 0, "resource": "b305b543da332a2cf6e1ce55ed2ea79", "verbose_msg": "The requested resource is not among the finished, queued or pending scans"}

23e371b816bab10cd9cf4046154022c
{"response_code": 0, "resource": "23e371b816bab10cd9cf4046154022c", "verbose_msg": "The requested resource is not among the finished, queued or pending scans"}

5e17055c51724b0b89ff06d36d02f5208a
{"response_code": 0, "resource": "5e17055c51724b0b89ff06d36d02f5208a", "verbose_msg": "The requested resource is not among the finished, queued or pending scans"}

e62dad2856c099a066713883bc12788
{"response_code": 0, "resource": "e62dad2856c099a066713883bc12788", "verbose_msg": "The requested resource is not among the finished, queued or pending scans"}

05552a77620933dd80f1e176736f8fe7
{"response_code": 0, "resource": "05552a77620933dd80f1e176736f8fe7", "verbose_msg": "The requested resource is not among the finished, queued or pending scans"}

079028d315d039da0ffec2728bc9e6f

{"scans": {"Bkav": {"detected": true, "version": "1.3.0.4595", "result": "W32.WoletixC.Trojan", "update": "20140603"},
"MicroWorld-EScan": {"detected": true, "version": "12.0.250.0", "result": "Backdoor.Agent.AAZI", "update": "20140604"},
"nProtect": {"detected": true, "version": "2014-06-04.01", "result": "Backdoor/W32.Agent.14336.AG", "update": "20140604"},
"CMC": {"detected": true, "version": "1.1.0.977", "result": "Trojan-Downloader.Win32.Agent010", "update": "20140604"},
"CAT-Quick Heal": {"detected": true, "version": "14.00", "result": "Backdoor. Likepust.b3", "update": "20140604"},
"McAfee": {"detected": true, "version": "6.0.4.564", "result": "BackDoor-FAL0R079028D31500", "update": "20140604"},
"Malwarebytes": {"detected": false, "version": "1.75.0001", "result": null, "update": "20140604"},
"SUPERAntiSpyware": {"detected": false, "version": "5.6.0.1032", "result": null, "update": "20140604"},
"TheHacker": {"detected": true, "version": "6.8.0.5.463", "result": "Trojan/Downloader.Agent.tmyh", "update": "20140602"},
"K7GW": {"detected": true, "version": "9.178.12292", "result": "Backdoor ( 04c525311 )", "update": "20140603"},
"K7AntiVirus": {"detected": true, "version": "9.178.12292", "result": "Backdoor ( 04c525311 )", "update": "20140603"},
"Anitum": {"detected": true, "version": "5.5.1.3", "result": "Trojan.DL.Agent.o3844v8", "update": "20140602"},
"F-Proot": {"detected": true, "version": "4.7.1.166", "result": "W32/Trojan-Dl-SysWrt-based!Max", "update": "20140604"},
"Symantec": {"detected": true, "version": "20131.5.6.11", "result": "Backdoor.Trojan", "update": "20140604"},
"Norman": {"detected": true, "version": "7.04.04", "result": "Agent.AOILSS", "update": "20140604"},
"TotalDefense": {"detected": false, "version": "37.0.10977", "result": null, "update": "20140603"},
"TrendMicro-HouseCall": {"detected": true, "version": "9.700-1001", "result": "BKDR_LIKSPUT.SMR", "update": "20140604"},
"Avast": {"detected": true, "version": "8.0.1489.320", "result": "Win32:Malware-gen", "update": "20140604"},
"ClamAV": {"detected": false, "version": "0.98.3", "result": null, "update": "20140603"},
"Kaspersky": {"detected": true, "version": "12.0.0.1225", "result": "Trojan-Downloader.Win32.Agent.xumu", "update": "20140604"},
"BitDefender": {"detected": true, "version": "7.2", "result": "Backdoor.Agent.AAZI", "update": "20140604"},
"NANO-Antivirus": {"detected": true, "version": "0.28.0.60100", "result": "Trojan.Win32.Agent.cpgsv", "update": "20140604"},
"AegisLab": {"detected": false, "version": "1.5", "result": null, "update": "20140604"},
"ByteHero": {"detected": false, "version": "1.0.0.1", "result": null, "update": "20140604"},
"Tencent": {"detected": false, "version": "1.0.0.1", "result": null, "update": "20140604"},
"Ad-Aware": {"detected": true, "version": "12.0.163.0", "result": "Backdoor.Agent.AAZI", "update": "20140604"},
"Sophos": {"detected": true, "version": "4.9.0", "result": "Troj/Agent-UCB", "update": "20140604"},
"Comodo": {"detected": true, "version": "18430", "result": null}}
```
"UnclassifiedMalware", "update": "20140604"}, "F-Secure": {"detected": true, "version": "11.0.19100.45", "result": "Backdoor.Agent.AAZI", "update": "20140604"}, "DrWeb": {"detected": true, "version": "7.00.9.04080", "result": "Trojan.DownLoad2.44669", "update": "20140604"}, "VIPRE": {"detected": true, "version": "29924", "result": "Trojan.Win32.GenericIBT", "update": "20140604"}, "AntiVir": {"detected": true, "version": "7.11.152.224", "result": "TR/Spy.Gen", "update": "20140604"}, "TrendMicro": {"detected": true, "version": "9.740-1012", "result": "TROJ_GEN.FOCC2001413", "update": "20140604"}, "McAfee-GW-Edition": {"detected": true, "version": "2013", "result": "BackDoor-FALR10790283315D0", "update": "20140603"}, "Emsisoft": {"detected": true, "version": "3.0.0.599", "result": "Backdoor.Agent.AAZI (B)", "update": "20140604"}, "Antiy-AVL": {"detected": true, "version": "0.1.0.1", "result": "Trojan[Downloader]/Win32.Agent", "update": "20140603"}, "Kingsoft": {"detected": true, "version": "2013.04.09.267", "result": "Win32.TrojDownloader.Agent.(kcloud)", "update": "20140604"}, "Microsoft": {"detected": true, "version": "1.10600", "result": "Backdoor.Win32.Likseput.B", "update": "20140604"}, "VIRobot": {"detected": true, "version": "2011.4.7.4223", "result": "Trojan.Win32.A.Downloader.14336.AV", "update": "20140604"}, "AhnLab-V3": {"detected": true, "version": "2014.06.04.00", "result": "Downloader/Win32.Agent", "update": "20140603"}, "GData": {"detected": true, "version": "24", "result": "Backdoor.Agent.AAZI", "update": "20140604"}, "Commtouch": {"detected": true, "version": "5.4.1.7", "result": "W32/Trojan-Dir-SysWrt-based\Max", "update": "20140604"}, "ESET-NOD32": {"detected": true, "version": "9891", "result": "a variant of Win32.Agent.PNC", "update": "20140604"}, "VBA32": {"detected": true, "version": "3.12.26.0", "result": "TrojanDownloader.Agent", "update": "20140604"}, "Baidu-International": {"detected": true, "version": "3.5.1.41473", "result": "Trojan.Win32.Downloader.AYY", "update": "20140604"}, "Rising": {"detected": false, "version": "25.0.0.11", "result": "null", "update": "20140603"}, "Ikarus": {"detected": true, "version": "7.3.1.6.1.0", "result": "Backdoor.Win32.Likseput", "update": "20140604"}, "Fortinet": {"detected": true, "version": "4", "result": "W32/Agent.OIGIt", "update": "20140604"}, "AVG": {"detected": true, "version": "14.0.0.3955", "result": "Downloader/Agent2.AVNR", "update": "20140604"}, "Panda": {"detected": true, "version": "10.0.3.5", "result": "Generic Backdoor", "update": "20140604"}, "Qihoo-360": {"detected": true, "version": "1.0.0.1015", "result": "HEUR/Malware.QVM07.Gen", "update": "20140604"}, "scan_id": "4123011354d8259e919fbdf605be1973a79100074595dca9d0cd1955667b8e93-1401874699", "sha1": "565a1b0b23f7c8e89030bc13b51e80df264a13", "resource": "079028d315d039da0ffec27282b2c9ef6", "response_code": 1, "scan_date": "2014-06-04 09.38:19", "permalink": "https://www.virustotal.com/file/4123011354d8259e919fbdf605be1973a79100074595dca9d0cd1955667b8e93/analysis/1401874699", "verbose_msg": "Scan finished, scan information embedded in this object", "total": 51, "positives": 43, "sha256": "4123011354d8259e919fbdf605be1973a79100074595dca9d0cd1955667b8e93", "md5": "079028d315d039da0ffec27282b2c9ef6"}

Figure 40. VirusTotal queries with Python
The same, easy way you can query other malware databases such as Malware Hash Registry (MHR) from Team Cymru and a sample script for doing that is shown below:

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from lxml import etree # http://lxml.de/xpathsl.html#the-xpath-method
import hashlib
from cymru.mhr.dns import DNSClient as mhr

namespaces = {
    'xsi': 'http://www.w3.org/2001/XMLSchema-instance',
    'stix': 'http://stix.mitre.org/stix-1',
    'stixVocabs': 'http://stix.mitre.org/default_vocabularies-1',
    'stixCommon': 'http://stix.mitre.org/common-1',
    'cybox': 'http://cybox.mitre.org/cybox-2',
    'cyboxCommon': 'http://cybox.mitre.org/common-2',
    'cyboxVocabs': 'http://cybox.mitre.org/default_vocabularies-2',
    'indicator': 'http://stix.mitre.org/Indicator-2',
    'ttp': 'http://stix.mitre.org/TTP-1',
    'marking': 'http://data-marking.mitre.org/Marking-1',
    'simpleMarking': 'http://data-marking.mitre.org/extensions/MarkingStructure#Simple-1',
    'openiocTM': 'http://stix.mitre.org/extensions/TestMechanism#OpenIOC2010-1',
    'mandiant': 'http://www.mandiant.com',
    'FileObj': 'http://cybox.mitre.org/objects#FileObject-2',
    'WinServiceObj': 'http://cybox.mitre.org/objects#WinServiceObject-2',
    'WinProcessObj': 'http://cybox.mitre.org/objects#WinProcessObject-2',
    'WinExecutableFileObj': 'http://cybox.mitre.org/objects#WinExecutableFileObject-2',
    'WinRegistryKeyObj': 'http://cybox.mitre.org/objects#WinRegistryKeyObject-2',
    'WinHandleObj': 'http://cybox.mitre.org/objects#WinHandleObject-2',
    'ProcessObj': 'http://cybox.mitre.org/objects#ProcessObject-2',
    'WinDriverObj': 'http://cybox.mitre.org/objects#WinDriverObject-2'
}

client=mhr()

f = 'Appendix_G_IOCs_Full.xml' # http://stix.mitre.org/downloads/APT1-STIX.zip
doc = etree.parse(f)

    print r.text
```
print client.lookup(r.text)

After running it you will get an output like (without colour distinctions):

```python
$ python cybox_xpath-mhr.py
[...]
0c5e9f564115bfcb6e66377a829de55f
<cymru.mhr.dns.mhr instance: ts:1361642853|detection:41%|_hash:0c5e9f564115bfcb6e66377a829de55f>
0f23d5b93c0681655d8a4258bde129
<cymru.mhr.dns.mhr instance: ts:None|detection:None%|_hash:0f23d5b93c0681655d8a4258bde129>
0ff20d023d6b54661d66fb3ce90af3c
<cymru.mhr.dns.mhr instance: ts:None|detection:None%|_hash:0ff20d023d6b54661d66fb3ce90af3c>
120c2e085992ff59a21ba401ec29efc9
<cymru.mhr.dns.mhr instance: ts:1367288162|detection:64%|_hash:120c2e085992ff59a21ba401ec29efc9>
150c4c1f589c4baa794160276a3d4aba
<cymru.mhr.dns.mhr instance: ts:None|detection:None%|_hash:150c4c1f589c4baa794160276a3d4aba>
1ce4605e771a04e375e0d1083f183e8e
<cymru.mhr.dns.mhr instance: ts:1255088157|detection:60%|_hash:1ce4605e771a04e375e0d1083f183e8e>
1ede2c69d50e0efbe23f758d902216e0
<cymru.mhr.dns.mhr instance: ts:None|detection:None%|_hash:1ede2c69d50e0efbe23f758d902216e0>
1f92ff8711716ca795fbd81c477e45f5
<cymru.mhr.dns.mhr instance: ts:1361643138|detection:55%|_hash:1f92ff8711716ca795fbd81c477e45f5>
1fb4ce2e56ced51ddf1edff8ed15c21b
<cymru.mhr.dns.mhr instance: ts:1386799871|detection:61%|_hash:1fb4ce2e56ced51ddf1edff8ed15c21b>
```
With contemporary programming languages and their libraries the artifact analysis laboratory can be extended with many useful capabilities. It also allows user to make easy and fast mass verifications of artifacts in large databases like VirusTotal or MHR using their API.

With XPATH one can read any value from STIX, CybOX etc XML formats, so that creating many useful utilities like ‘format aware grep’ – a pattern matching utility becomes possible.
Annex A: Installation instructions

A.1 Mantis installation

The installation instructions below have been tested on an out-of-the-box installation of Ubuntu Linux 14.04 LTS.

**Attention**: If you are setting up a virtual machine, make sure to give it at least 3GB of memory if you want to import really large XML structures such as MITRE’s STIX conversion of the Mandiant APT-1 report ([http://stix.mitre.org/downloads/APT1-STIX.zip](http://stix.mitre.org/downloads/APT1-STIX.zip)) – importing large files currently takes a lot of memory – there seems to be a memory leak which we still have to track down.

Make sure that you have the required dependencies on OS level for building the XML-related packages. For example, on an Ubuntu system, execute the following commands:

```bash
$ sudo apt-get update && sudo apt-get install libxml2 libxml2-dev python-dev libxslt1-dev libz-dev
```

Also, while you are at it, install git, if you do not have it already:

```bash
$ sudo apt-get install git
```

If you are behind a proxy, you can configure a proxy for apt-get by putting a file `proxy` into `/etc/apt/apt.conf.d` that has the following contents:

```bash
Acquire::http::proxy "<proxy_url>";
Acquire::ftp::proxy "<proxy_url>";
Acquire::https::proxy "<proxy_url>";
```

It is recommended to use a virtual python environment.

Make sure that virtualenv and pip are installed:

```bash
$ sudo apt-get install python-virtualenv python-pip
```

Create a virtual environment:

```bash
$ virtualenv /home/enisa/mantis
$ source /home/enisa/mantis/bin/activate
```

---

Now the virtual environment is activated – you should see a changed prompt that is prefixed with (mantis).

Unfortunately, the process of getting libxml2-python installed using pip varies from OS to OS, because there is no proper library package available. For Ubuntu 14.04, do the following:

Download and unpack the libxml2 sources:

```
(mantis)$ wget http://xmlsoft.org/sources/libxml2-2.9.1.tar.gz
(mantis)$ tar -zxvf libxml2-2.9.1.tar.gz
```

Install via pip:

```
(mantis)$ pip install libxml2-2.9.1/python
```

Go to a location where you want to have the Django Mantis files and check out the git repository:

```
(mantis)$ git clone https://github.com/siemens/django-mantis.git
```

If you are behind a proxy, you can configure a proxy for git via the following:

```
(mantis)$ git config --global http.proxy <proxy_url>
```

Change into the django-mantis directory and do:

```
(mantis)$ cd django-mantis/
(mantis)$ sed -i 's/Django>=1.6/Django==1.6.2/g' requirements/base.txt
(mantis)$ pip install -r requirements/local.txt
(mantis)$ pip install "django-simple-menu>=1.0.6"
```
Last thing to do is to move the database location from /tmp (default) to our home directory:

```
(mantis)$ mkdir /home/enisa/django-mantis/db
(mantis)$ sed -i 's/\(tmp\)/\(django\)/' mantis/settings/local.py
```

You are now all set for running MANTIS on top of an SQLite database.

More details about installation (like running MANTIS on top of Postgresql) you can find on http://django-mantis.readthedocs.org/en/latest/installation.html

### A.2 CRITs installation

At the beginning you need to install dependencies, depending on the system:

- [https://github.com/crits/crits_dependencies](https://github.com/crits/crits_dependencies) – 64-bit dependencies

For Install dependencies on Ubuntu 14.04 32-bit type command:

```
wget https://raw.githubusercontent.com/adamziaja/crits_dependencies/master/install_dependencies_ubuntu_32bit.sh && chmod +x install_dependencies_ubuntu_32bit.sh && ./install_dependencies_ubuntu_32bit.sh
```

#### A.2.1 Setting up your single server instance of MongoDB

Create the database directory:

```
sudo mkdir -p /data/db
```

In the ‘contrib’ directory that came with CRITs, you will find a mongo directory with two directories in it: one for Ubuntu, and one for RHEL. They contain start scripts for your mongo processes. These scripts properly configure reclaim_mode on your server and start the mongod process. cd to the directory for your OS and run the mongod_start.sh script:

```
sudo ./mongod_start.sh
```

Verify this is working by connecting to it with the following command:

```
Mongo
```

This should bring up the mongo shell on localhost.

#### A.2.2 Installing CRITs using the Django runserver

The Django runserver is our recommended web server for development or test instances of CRITs. It is quick, light, and provides a way for developers and administrators to look at the web server requests/responses in real time. It is also useful for debugging and viewing print statements.

Installing the codebase:

---

21 [https://github.com/crits/crits](https://github.com/crits/crits)

22 See [https://github.com/crits/crits](https://github.com/crits/crits) for current CRITs materials
If you are a developer cloning a git repository, we generally recommend you clone to ~/git/crits. If you are using a release tarball, un-tar the tarball in a place of choice.

Edit the database file for your environment:

In the crits/config directory that came with the CRITs codebase, copy database_example.py to database.py:

```
cp database_example.py database.py
```

Edit database.py using the comments to configure your MongoDB connection information and your SECRET_KEY. If you are unsure what S3 is or if you are using it, leave FILE_DB alone.

Create the default collections in MongoDB:

NOTE: at this point you should have MongoDB running!

Run the create_default_collections management command to setup your database:

```
python manage.py create_default_collections
```

Add your first user:

Take a look at the options for the user management command:

```
python manage.py users --h
```

Use that command to setup your first admin user for CRITs. Be sure to use -A to set them as an admin. Make note of the temporary password provided in the output!

Set your allowed hosts:

Django needs to know the host(s) or domain name(s) that you will be serving your CRITs instance from for security purposes. To set this, run the following command:

```
python manage.py setconfig allowed_hosts "foo"
```

Where "foo" is the host/domain name, or a comma separated list of names that will be serving CRITs.

### A.2.3 CRITs cronjobs

The main cronjob we recommend is for the script which executes common mapreduce jobs. These jobs do things like collect database statistics, generate Campaign information, and other useful bits of information. If you would like the Counts and stats updated on your Dashboard, you will need to add this.

We also support sending batch email notifications to users of your system. The email provided a non-detailed overview of how many changes have happened to items they are subscribed to. This cronjob also updates the notifications users will see in the interface.

As a user who has access to the codebase and to execute python code, edit their crontab:

```
crontab --e
```

Add the following entries, making adjustments for the folder path and the frequency you want them to run:

```
0 * * * * cd /data/crits/ && /usr/bin/python manage.py mapreduces
0 * * * * cd /data/crits/ && /usr/bin/python manage.py generate_notifications
```
5 Bibliography

1. STIX and TAXII: On the road to becoming the de facto standard

2. NIST Special Publication 800-150 (Draft) : Guide to Cyber Threat 5 Information Sharing (Draft)

3. The International Security Community Should Embrace the STIX and TAXII Standards

4. Detect, SHARE, Protect - Solutions for Improving Threat Data Exchange among CERTs

5. US-CERT: Information Sharing Specifications for Cybersecurity

6. ENISA: Detect, SHARE, Protect : Solutions for Improving Threat Data Exchange among CERTs

7. OpenIOC: An Open Framework for Sharing Threat Intelligence: Overview


12. Network Working Group, RFC 5070 The Incident Object Description Exchange Format
Common Framework for Artifact Analysis Activities

December 2014

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
Maroussi 151 24, Athens, Greece