CTI Operational Procedures with Jupyter Notebooks and MISP

CTIS - October 19/20 - 2022

cudeso.be
We Secure You
https://www.cudeso.be
koen.vanimpe@cudeso.be

Koen Van Impe

- **Freelancer**
  - Incident response, threat intelligence, security monitoring
- **Open source contributions**
  - MISP modules, taxonomies, automation and integration with DFIR tools, ...
  - "*MISP tip-of-the-week*"
- **BelgoMISP**
  - Belgian MISP User Group
- **OSINT threat feed**
  - botvrij.eu

koen.vanimpe@cudeso.be

https://www.cudeso.be

https://www.vanimpe.eu

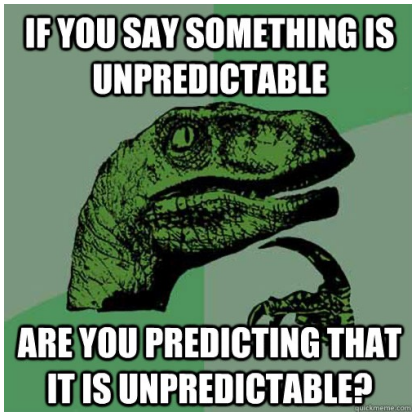
https://github.com/cudeso

@cudeso

CTI operational procedures

Operational procedures?

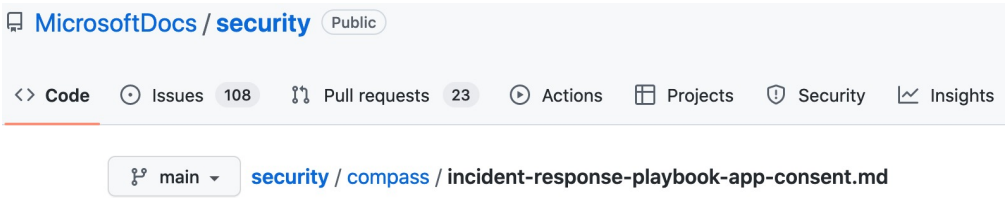
- Playbooks
- Standard Operating Procedures (SOP)
- Workflows



- Consistent approach
- Recipe for an investigation
- Repeatable
- Predictable
- Completeness checks
- **Documented** actions
- Leads up to automation

Typical formats of operating procedures, workflows or playbooks

- **Markdown**, stored in a wiki / GitLab / GitHub



- **Security Playbooks**: CACAO JSON, shared via MISP objects
  - Collaborative Automated Course of Action Operations (CACAO) security playbooks

OASIS OPEN

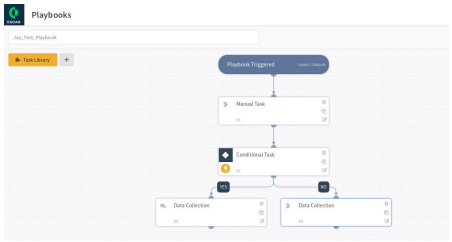
CACAO Security Playbooks Version 1.1

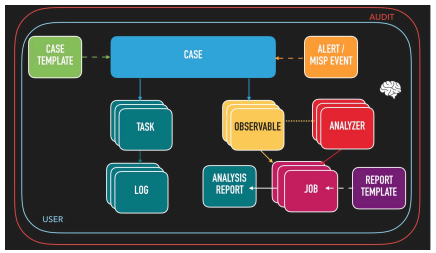
security-playbook

The security-playbook object provides meta-information and allows managing, storing, and sharing cybersecurity playbooks and orchestration workflows.

security-playbook is a MISP object available in JSON format at **this location** The JSON format can be freely reused in your application or automatically enabled in MISP.

- And others ...





Some basic examples of CTI operating procedures

Consumer

Observed IP address

1. Query matches in the cache of OSINT feeds
2. Query for matches in internal MISP
3. Document title, date and context (campaign, actor, sector)
4. Document advised action (PAP / CoA)
5. Start enrichment
6. Query ASN, rDNS and pDNS
7. Document ASN, rDNS and pDNS matches
8. Discover and document related IPs and domains

Producer

Encode object in TI platform

1. Check completeness of necessary attributes
2. Define MISP event where to add object
3. Check attributes for false positives (warninglist)
4. Check similarities for existing objects
5. Create object
6. Add the attributes
7. Add expected follow-up actions
8. Add relationships with other objects
9. Document object ID, event ID and outcome of creation

Improve this workflow

How to deal with these CTI operating procedures?

Uniform format

Shareable

Easy to use

Version tracking

Not just a list of queries to execute, must include documentation

How to deal with these CTI operating procedures?

Uniform format

Shareable

Easy to use

Version tracking

Not just a list of queries to execute, must include documentation

Jupyter notebooks in PyMISP

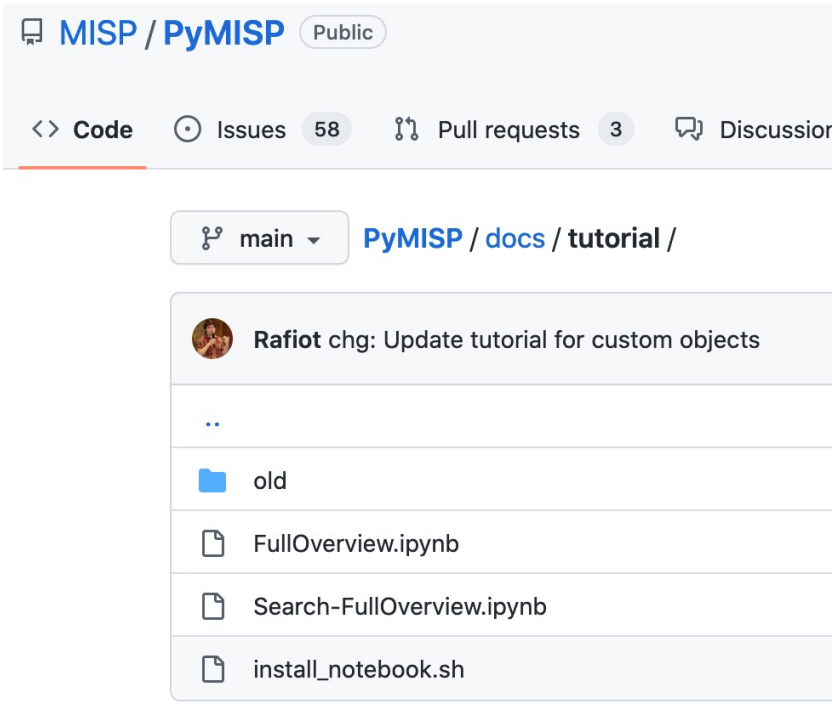
PyMISP

- Python library to access MISP
- Automation
- Adding and editing data
- Integration
- ...

Jupyter notebooks in PyMISP

PyMISP

- Python library to access MISP
- Automation
- Adding and editing data
- Integration
- ...



What are Jupyter notebooks?

Interactive environment

- Write and execute computer code
- Observe the results
- Document the code
- Text elements
- Images

Consumers

- Human readable
- Executable by machines

Kernel

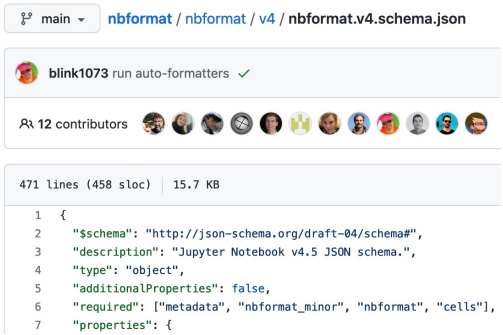
- Computational engine
- Executes the machine code

Distributed

- **Code and documentation are stored in the "execution environment"**
- **But documentation can be edited from anywhere**
- Web browser

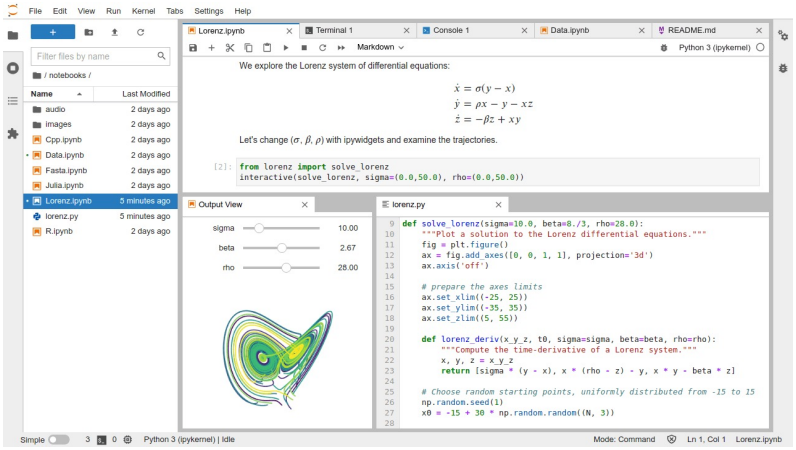
What are Jupyter notebooks?

- **Open source**
  - Used in data science
  - Other areas, such as troubleshooting, and CTI



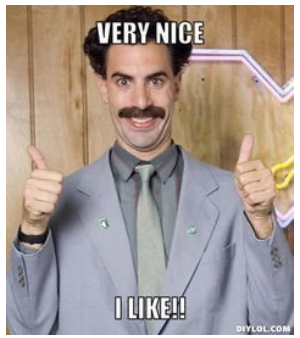
- Notebooks are stored in a **JSON** format (.ipynb)
  - Ideal for code repositories

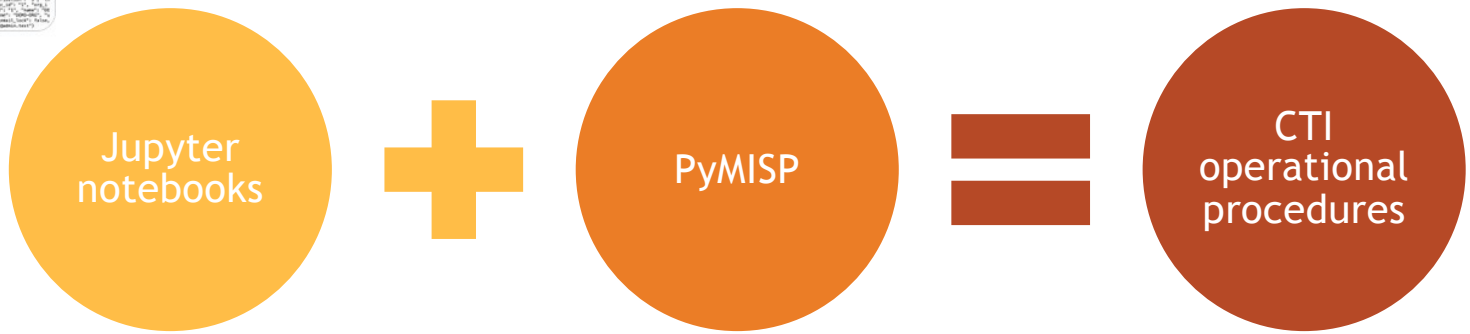
- Engines (kernel)
  - **Python**, community provided
  - Ruby, C++



- JupyterLab
  - “IDE”

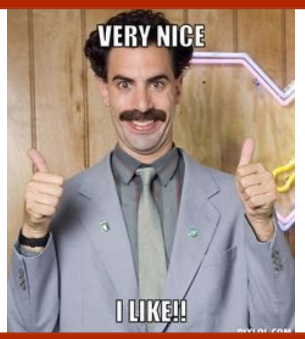
Jupyter notebooks, PyMISP and CTI operational procedures





Jupyter notebooks, PyMISP and CTI operational procedures

"Remote code execution" on a MISP server



Run the notebook on a stable, dedicated system (or user environment) with access to MISP

Jupyter notebooks + PyMISP = CTI operational procedures

Lesson 1: **Run notebooks from dedicated/well-maintained environments**

How to get started?

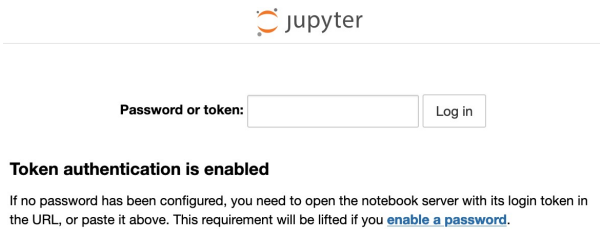

```
mkdir cti-operational-procedure
cd cti-operational-procedure
virtualenv venv
source venv/bin/activate
pip install notebook
pip install pymisp
jupyter notebook --no-browser --ip misp.demo.cudeso.be --port 8888
```

Access the Jupyter notebook server

```
(venv) koenv@misp-demo:~/cti-operational-procedure/notebooks$ jupyter notebook --no-browser --ip misp.demo.cudeso.be --port 8888
[I 14:46:11.877 NotebookApp] Serving notebooks from local directory: /home/koenv/cti-operational-procedure/notebooks
[I 14:46:11.877 NotebookApp] Jupyter Notebook 6.5.1 is running at:
[I 14:46:11.878 NotebookApp] http://misp.demo.cudeso.be:8888/?token=f8840d3b2db
[I 14:46:11.878 NotebookApp]  or http://127.0.0.1:8888/?token=f8840d3b2dbbc2170
[I 14:46:11.878 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

- When only running on localhost (or firewalled)
  - Login with SSH
  - Setup port forwarding

- Access the URL
  - **Token** gets set as a cookie in your browser
  - If you forget to include the token you have to enter it in a web form



A note on the Jupyter notebook server

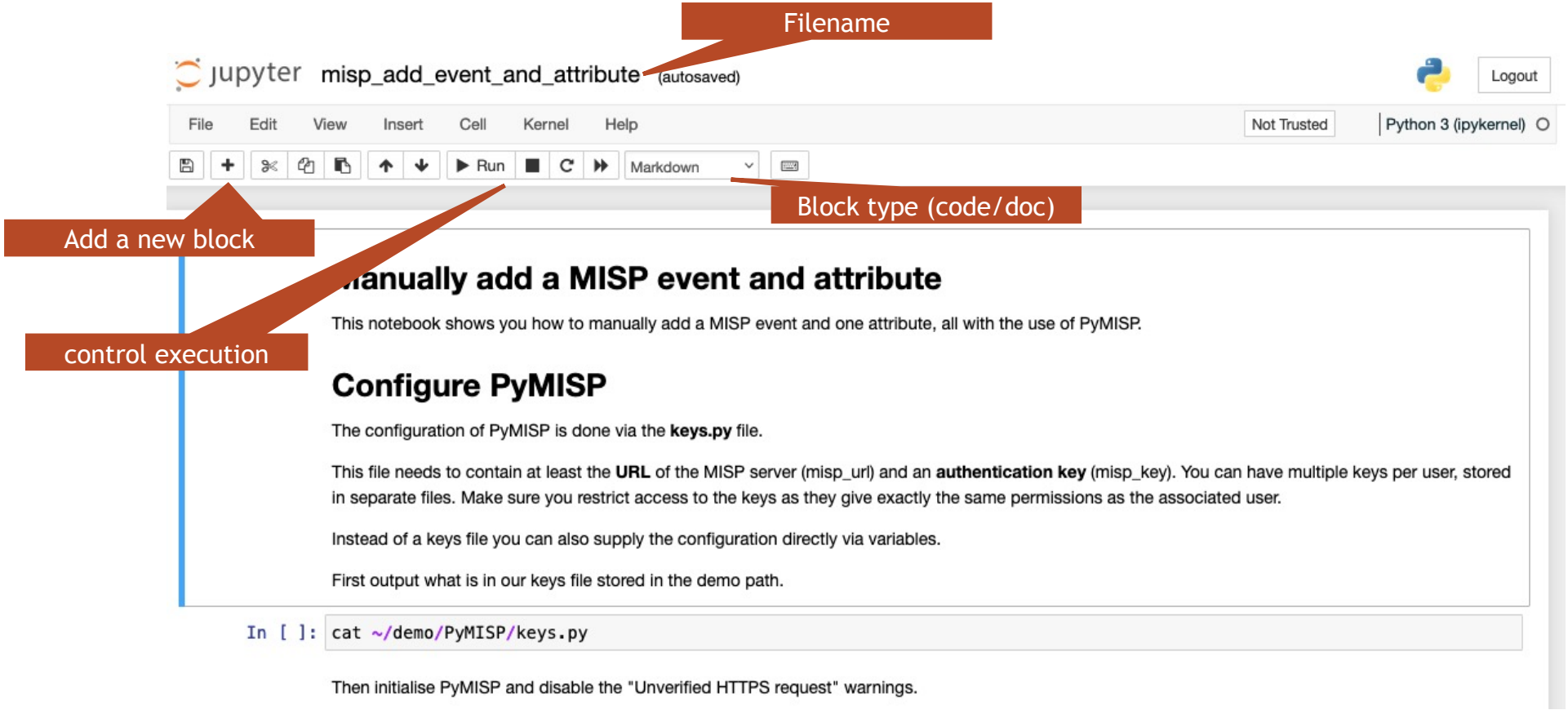
- Notebooks are served from the **directory** where the server is executed
  - Path
  - Environment conditions
    - MISP
    - Custom venv

```
git pull procedure-repo
git add .
git commit -m “Update to Procedure 1”
git push
```

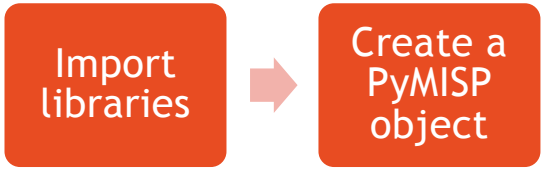
Lesson 2: **Pay attention to the location from where you start the notebook server**

- The notebook server is meant for **single-user use**
  - Multiple users accessing the interface will clutter the results
  - Need multiple users? Have a look at JupyterHub
    - Multi-user hub that spawns multiple instances of single-user Jupyter notebook server

Notebook interface



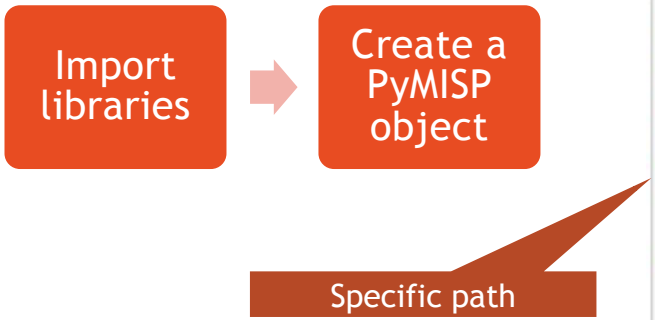
Basic skeleton script




```python
import urllib3
from pymisp import PyMISP, MISPEvent
from keys import misp_url, misp_key, misp_verifycert

if misp_verifycert is False:
    import urllib3
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

misp = PyMISP(misp_url, misp_key, misp_verifycert)

print("I will use the server {}".format(misp_url))
```

- Where is the API key? The MISP URL?
  - “keys.py”

Basic skeleton script




```python
import urllib3
from pymisp import PyMISP, MISPEvent
import sys
sys.path.insert(0, "/home/koenv/cti-operational-procedure/vault/")
from keys import misp_url, misp_key, misp_verifycert

if misp_verifycert is False:
    import urllib3
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

misp = PyMISP(misp_url, misp_key, misp_verifycert)

print("I will use the server {}".format(misp_url))
```

- Where is the API key? The MISP URL?
  - “keys.py”

Lesson 3: **Do not store credentials in a notebook!**

Example: Create a MISP event

Document threat behaviour in MISP

This procedures walks you through the steps of creating a new event in MISP.

Trigger

This procedure is triggered when a new threat behaviour is observed during incident response.

Configure PyMISP

In [1]:
```
import urllib3
from pymisp import PyMISP, MISPEvent
import sys
sys.path.insert(0, "/home/koenv/cti-operational-procedure/vault/")
from keys import misp_url, misp_key, misp_verifycert

if misp_verifycert is False:
    import urllib3
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

misp = PyMISP(misp_url, misp_key, misp_verifycert)

print("I will use the server {}".format(misp_url))
```

```
I will use the server https://misp.demo.cudeso.be/
```

Debug output.
If creation of pymisp object fails we will not get to this debug message

Lesson 4: Test the connection at the start of the procedure.

Example: Create a MISP event

Document threat behaviour in MISP

This procedures walks you through the steps of creating a new event in MISP.

Trigger

This procedure is triggered when a new threat behaviour is observed during incident response.

Configure PyMISP

In [1]:
```
import urllib3
from pymisp import PyMISP, MISPEvent
import sys
sys.path.insert(0, "/home/koenv/cti-operational-procedure/vault/")
from keys import misp_url, misp_key, misp_verifycert

if misp_verifycert is False:
    import urllib3
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

misp = PyMISP(misp_url, misp_key, misp_verifycert)

print("I will use the server {}".format(misp_url))
```

```
I will use the server https://misp.demo.cudeso.be/
```

Create a MISP event

Analyst: add basic event elements

Set the **event title**, **distribution**, **threat level** and **analysis**.

Distribution

- 0: Your organization only
- 1: This community-only
- 2: Connected communities
- 3: All communities

Threat level

- 1: Low (mass malware)
- 2: Medium (APT)
- 3: High (0-day)
- 4: Undefined

Analysis

- 0: Initial
- 1: Ongoing
- 2: Completed

One code block with **variables**
One code block with code to execute

In [2]:
```
new_event_title = "CTIS-2022 Threat alert"
new_event_distribution = 3
new_event_threat_level = 4
new_event_analysis = 1
```

In [3]:
```
event = MISPEvent()
event.info = new_event_title
event.distribution = new_event_distribution
event.threat_level_id = new_event_threat_level
event.analysis = new_event_analysis
```

Lesson 5: Avoid that analysts have to fiddle with ‘raw code.

Example: Create a MISP event

Create a MISP event

Analyst: add basic event elements

Set the **event title**, **distribution**, **threat level** and **analysis**.

Distribution

- 0: Your organization only
- 1: This community-only
- 2: Connected communities
- 3: All communities

Threat level

- 1: Low (mass malware)
- 2: Medium (APT)
- 3: High (0-day)
- 4: Undefined

Analysis

- 0: Initial
- 1: Ongoing
- 2: Completed

```
In [2]: new_event_title = "CTIS-2022 Threat alert"
        new_event_distribution = 3
        new_event_threat_level = 4
        new_event_analysis = 1
```

```
In [3]: event = MISPEvent()
        event.info = new_event_title
        event.distribution = new_event_distribution
        event.threat_level_id = new_event_threat_level
        event.analysis = new_event_analysis
```

Analyst: add a date

When did you discover / observed this threat?

```
In [4]: new_event_date = "2022-10-15"
```

```
In [5]: event.set_date(new_event_date)
```

Analyst: add the TLP level

By default we use TLP:AMBER. Refer to DOC for guidance on chosing the correct TLP level.

```
In [6]: event.add_tag("tlp:amber")
Out[6]: <MISPTag(name=tlp:amber)>
```

Create event in MISP

Send the request to the server.

```
In [7]: result = misp.add_event(event, pythonify=True)
        print("Created event ID {} for {}".format(result.id, result.uuid, new_event_title))
```

```
Created event ID 736 for d7da86fa-a549-4c4c-93c6-b2097433507d
```

Execution result.
Not just the “Python” success/failure, but the execution result of a step in the procedure.
Can also be used for reporting.

Lesson 6: **Print execution results of important steps.**

Example: Create a MISP event

Analyst: add a date

When did you discover / observed this threat?

In [4]:
```
new_event_date = "2022-10-15"
```

In [5]:
```
event.set_date(new_event_date)
```

Analyst: add the TLP level

By default we use TLP:AMBER. Refer to DOC for guidance on chosing the correct TLP level.

In [6]:
```
event.add_tag("tlp:amber")
```

Out[6]:
```
<MISPTag(name=tlp:amber)>
```

Create event in MISP

Send the request to the server.

In [7]:
```
result = misp.add_event(event, pythonify=True)
print("Created event ID {} for {}".format(result.id, result.uuid, new_event_title))
```

```
Created event ID 736 for d7da86fa-a549-4c4c-93c6-b2097433507d
```

Analyst: add attributes

Make sure you set the **type** and the **value**, and the **to_ids** flag. Set **to_ids** to True if the IP address needs to be blocked.

Add contextualisation to the attribute.

By default we use PAP:AMBER for the Permissible Action Procotol and set the expected Courses of Action to Deny.

In [8]:
```
new_attribute = {
    "type": "ip-dst",
    "value": "8.8.4.4",
    "to_ids": False,
    "tag": ["PAP:AMBER","course-of-action:active=\"deny\""],
    "comment": "Initial connectivity check"
}
```

In [17]:
```
from pymisp import MISPAttribute

attribute = MISPAttribute()
attribute.type = new_attribute["type"]
attribute.value = new_attribute["value"]
attribute.to_ids = new_attribute["to_ids"]
for t in new_attribute["tag"]:
    attribute.add_tag(t)
attribute.comment = new_attribute["comment"]
```

Add to event

And now add the attribute to the event

In [18]:
```
result_attr = misp.add_attribute(result.id, attribute, pythonify=True)
print("Added attribute {}".format(result_attr.uuid))
```

```
Added attribute 89c03549-0f76-40e5-85db-dd6a904a276a
```

Summary

Print a summary at the end of execution

In [19]:
```
print("The event {} ({}) was created to deal with the threat.".format(result.info,result.uuid))
print("The defined follow-up actions for {} are {}, in object {}".format(new_attribute["value"], new_attribute["tag"
```

```
The event CTIS-2022 Threat alert (d7da86fa-a549-4c4c-93c6-b2097433507d) was created to deal with the threat.
The defined follow-up actions for 8.8.4.4 are ['PAP:AMBER', 'course-of-action:active="deny"'], in object 89c03549-0
f76-40e5-85db-dd6a904a276a
```

Lesson 7: **Conclude procedures with a summary of what was done.**

Example: Create a MISP object

Create a MISP file object

Add a MISP file object to a specific threat event. As an analyst set the file attributes and the event ID.

Init MISP

In [1]:
```
import urllib3
from pymisp import PyMISP, MISPEvent, MISPObject
import sys
sys.path.insert(0, "/home/koenv/cti-operational-procedure/vault/")
from keys import *

if misp_verifycert is False:
    import urllib3
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

misp = PyMISP(misp_url, misp_key, misp_verifycert)

print("I will use the server {}".format(misp_url))
```

```
I will use the server https://misp.demo.cudeso.be/
```

Analyst: set file attributes

In [2]:
```
new_file = {
    'filename': "MyFileName.docx",
    'size-in-bytes': '2000',
    'md5': '6586f96163923187aaf4aaa233345162',
}
```

Analyst: set event ID

In [3]:
```
event_id = 674
```

Create the object

In [4]:
```
file_object = MISPObject('file')
file_object.add_attribute('filename', new_file.get('filename'))
file_object.add_attribute('md5', new_file.get('md5'))
file_object.add_attribute('size-in-bytes', new_file.get('size-in-bytes'))
res = misp.add_object(event_id, file_object)
print(res)
```

```
{'Object': {'id': '88447', 'name': 'file', 'meta-category': 'file', 'description': 'File object describing
ith meta-information', 'template_uuid': '688c46fb-5edb-40a3-8273-1af7923e2215', 'template_version': '24', '
': '674', 'uuid': 'a4c3d112-52bc-4411-8468-0a924d979996', 'timestamp': '1665862920', 'distribution': '5', '
group_id': '0', 'comment': '', 'deleted': False, 'first_seen': None, 'last_seen': None, 'Attribute': [{'id'
```

Example: Lookup MISP feed cache and PDNS results

Procedure to handle IP address

Init MISP

In [1]:
```
import urllib3
from pymisp import PyMISP, MISPEvent
import sys
sys.path.insert(0, "/home/koenv/cti-operational-procedure/vault/")
from keys import *

if misp_verifycert is False:
    import urllib3
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

misp = PyMISP(misp_url, misp_key, misp_verifycert)

import requests

misp_headers = {
            "Authorization": misp_key,
            "Content-Type": "application/json",
            "Accept": "application/json"
        }

import tabulate

print("I will use the server {}".format(misp_url))
```

```
I will use the server https://misp.demo.cudeso.be/
```

Analyst: set IP address

In [2]:
```
investigate_ip = "200.159.87.196"
```

Via 'requests' library, not via PyMISP

Lookup in the cache

In [3]:
```
misp_cache_url = "{}/feeds/searchCaches/value:{}".format(misp_url, investigate_ip)
cache_results = requests.get(misp_cache_url, headers=misp_headers, verify=misp_verifycert)
direct_urls = []
for el in cache_results.json():
    direct_urls.append(el["Feed"]["direct_urls"])
```

Fetch the cached events

In [4]:
```
cache_table = []
for el in direct_urls[0]:
    misp_preview_cache_url = el["url"]
    cached_event = requests.get(misp_preview_cache_url, headers=misp_headers, verify=misp_verifycert)

    cached_event_title = cached_event.json()["Event"]["info"]
    cached_event_data = cached_event.json()["Event"]["date"]
    cache_table.append([cached_event_title, cached_event_data])
```

In [5]:
```
print(tabulate.tabulate(cache_table, headers=["MISP Event title", "Date"]))
```

```
MISP Event title                                                                Date
------------------------------------------------------------------------------  ----------
Scraper: DeftTorero: tactics, techniques and procedures of intrusions revealed  2022-10-14
```

In [6]:
```
import pypdns
x = pypdns.PyPDNS(basic_auth=(pdns_user,pdns_password))
pdns_results = x.query(investigate_ip)
pdns_results_table = []
if len(pdns_results) > 0:
    for el in pdns_results:
        pdns_results_table.append([el["rdata"], el["rrname"], el["rrtype"]])

print(tabulate.tabulate(pdns_results_table, headers=["Data", "Name", "Type"]))
```

```
Data            Name                        Type
--------------  --------------------------  ------
200.159.87.196  c.ns.abys.com.br            A
200.159.87.196  a.ns.oscarcalcados.com.br   A
```

Example: Building integrations with a 3rd party

Task: integrate a new data provider in MISP

Learn 3rd party API functionality

A notebook to document your progress

Implement code snippets from provider

Link together and convert to MISP API queries

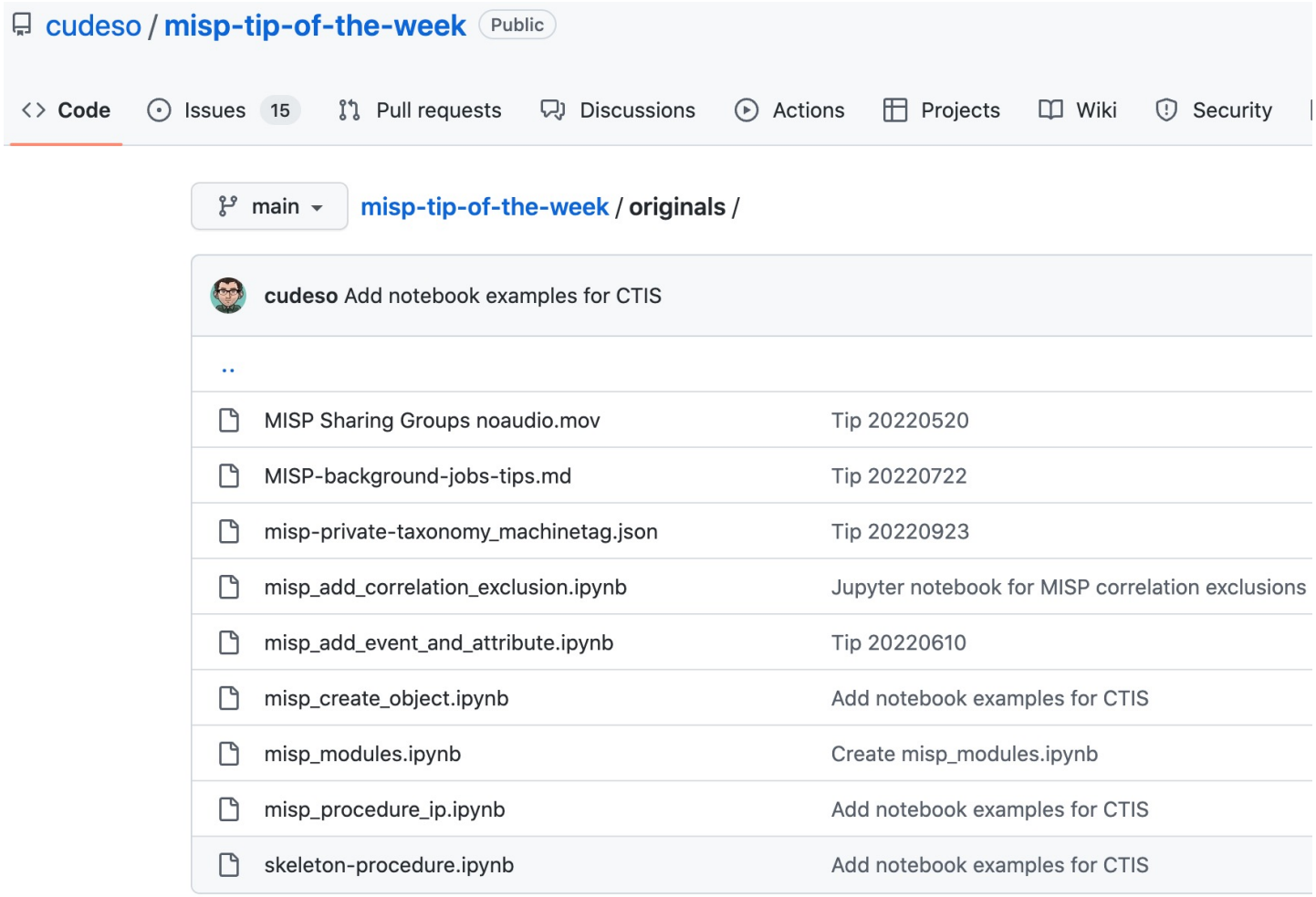
Code is stored together with findings

Demonstrate a PoC to colleagues

Show alternatives without multiple versions of a PoC

Your API usage complements documentation 3rd party

Some examples in ‘MISP tip of the week’



Resource monitoring

The Jupyter notebook executes code as a Python process

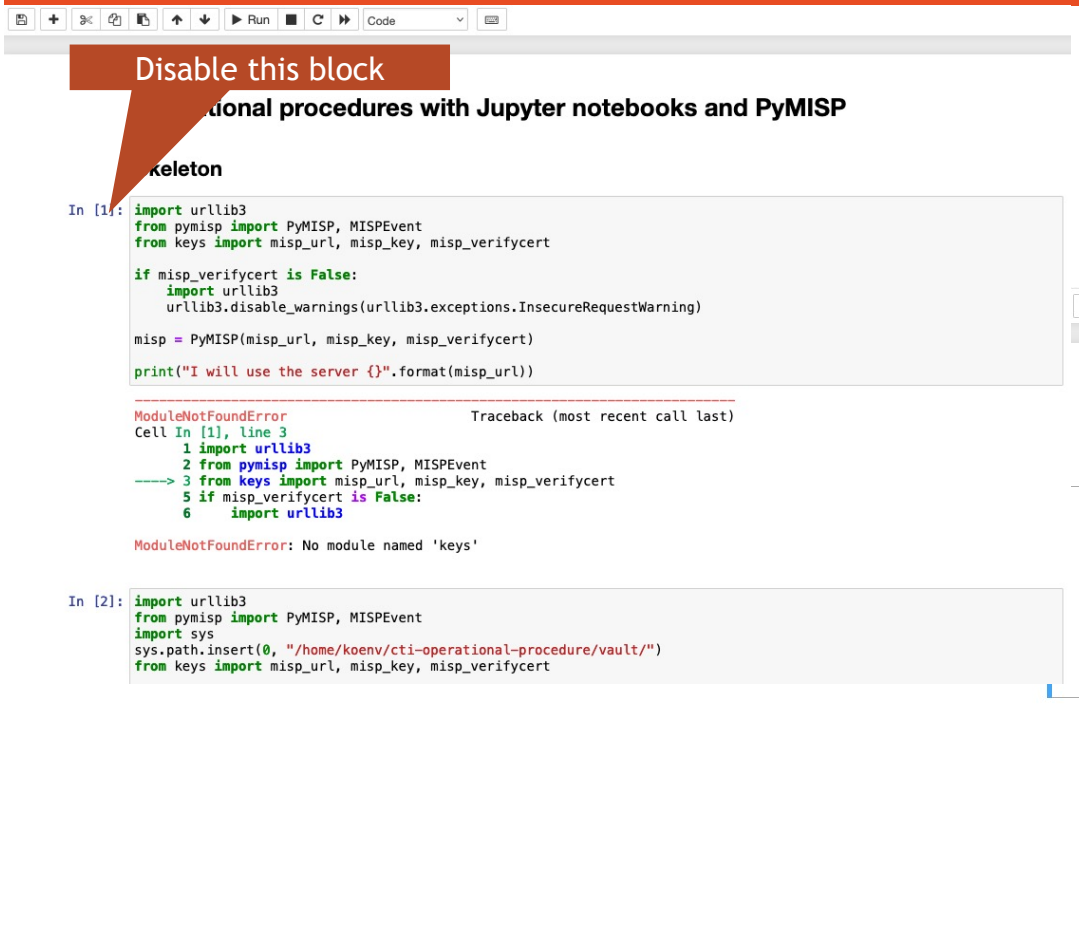
- Open a second console an monitor load for Python processes

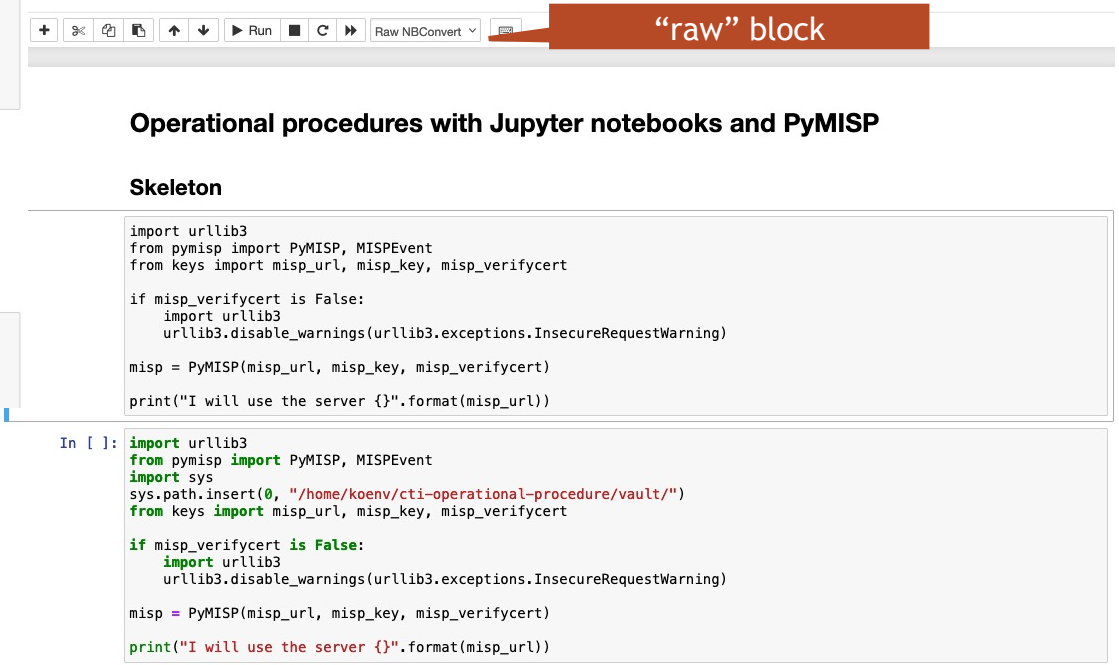
Large output can delay the response

- Whenever possible, limit the output

Lesson 8: **Monitor system resources when executing code.**

Multiple versions of a code cell





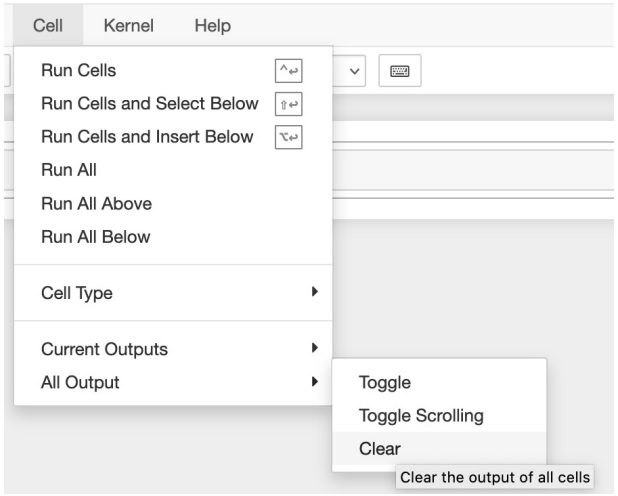
Lesson 9: Disable a code block with ESC R to test different versions of your code.

Output of code execution is stored in the notebook

Desired if you create a report

Not-desired if you create the base procedure

Not-desired if you share the script




```
(venv) koenv@misp-demo:~/cti-operational-procedure/notebooks$ cat skeleton-2.ipynb | jq '.[] | .[] | .outputs'
null
null
[
  {
    "name": "stdout",
    "output_type": "stream",
    "text": [
      "I will use the server https://misp.demo.cudeso.be/\n"
    ]
  }
]
```

Add a custom shortcut under Help>Edit Keyboard Shortcuts

toggle all cells output scrolled		add shortcut
clear all cells output	⇧O ✖	Shift-o
save notebook	S ✖	⌘S ✖ add shortcut
find and replace	F ✖	add shortcut
paste dialog	⌘V ✖	add shortcut

Lesson 10: **Clean the output of notebooks when sharing them.**

Workflow for notebooks

Notebooks stored in GitLab

Investigation machine

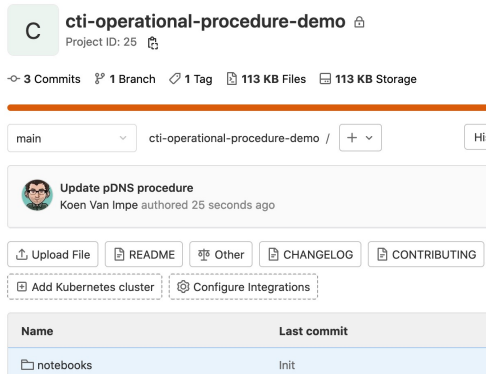
User with a 'stable' venv	Dev environment to push changes to notebooks

Shell script

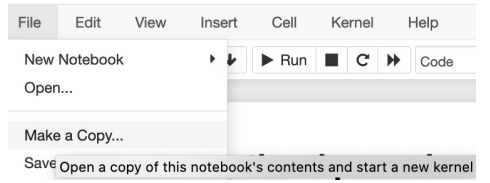
pulls latest notebooks	creates a case directory	starts the notebook server

In the notebook server

Create a copy of notebook	Avoids overwriting original notebook	Prepend with case name




```
git pull
cp -r notebooks $1
cd $1
jupyter notebook --no-browser --ip misp.demo.cudeso.be --port 8888
```



Moving forward

- Still discovering the possibilities of using Jupyter notebooks for operational procedures.

- Something to work on in the future ...

Lesson 10,5: It never stops ...

Moving forward

Shared repository of (PyMISP) notebooks

- Contribute to the existing repository?
- Assist communities that want to start writing their own procedures
- Help organisations who start with CTI

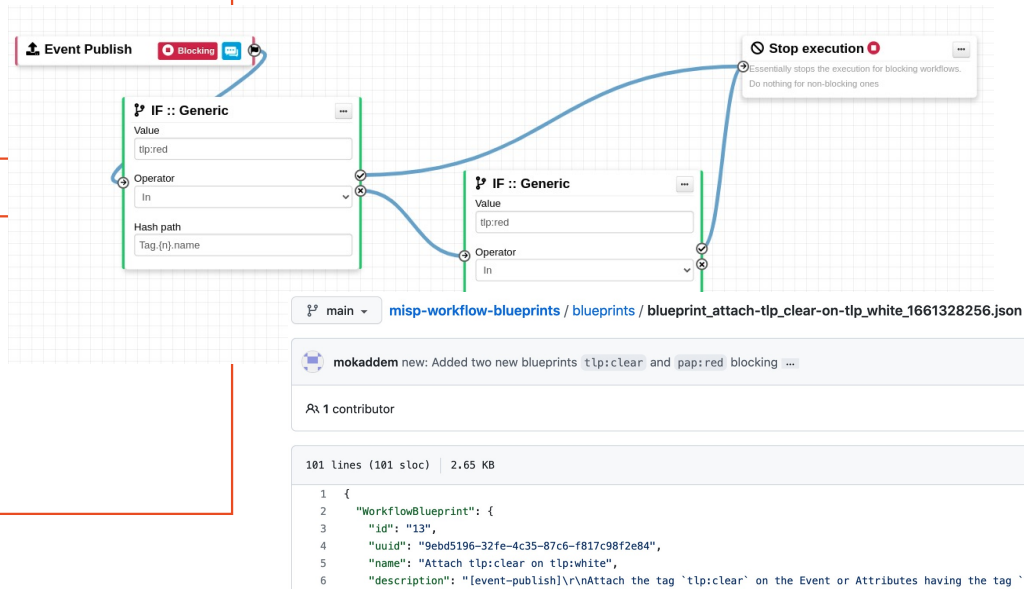
main PyMISP / docs / tutorial /

Develop good practices for documentation

- “Templates” on how to encode (threat) information
- Always add context and completeness
- Include expected actions
- ...

MISP workflows

- Integrate notebooks with the MISP workflows?
  - JSON
  - Trigger workflow actions from the notebook?
  - Process response from actions in the notebook?
  - Document execution path / workflow in a notebook



Questions?