Microsoft Security

Jupyter & MSTICPy in security operations and threat hunting

The MSTICPy Team

Ian Hellen

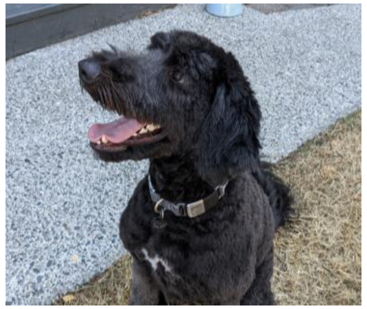
Principal Software Engineer

Pete Bryan

Senior Security Researcher

Ashwin Patil

Senior Security Researcher

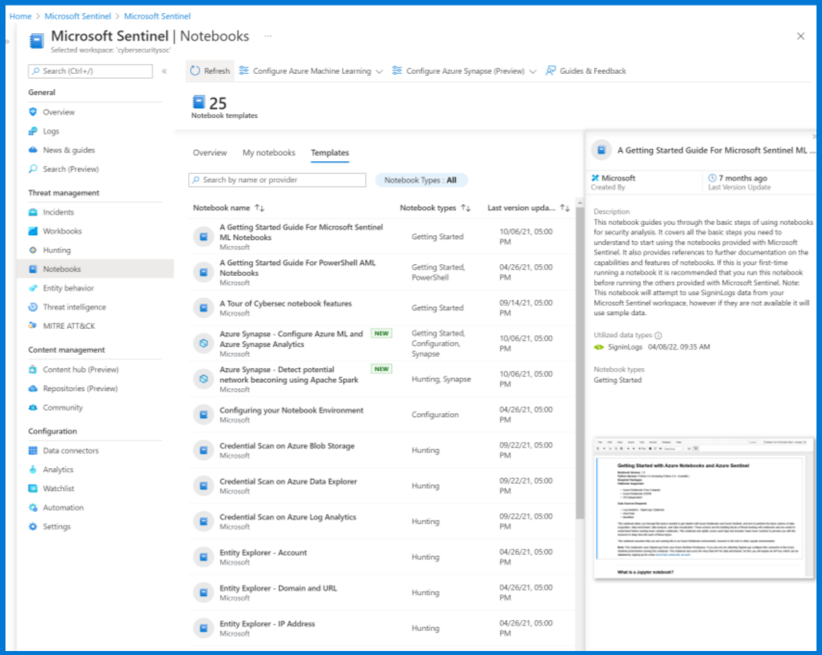


Our Community

Many & varied

History of MSTICPy

- Notebooks in Sentinel
- Need to share code with customers
- Need to maintain it
- Plenty of input and growth
  - 150k+ downloads



Notebook name	Notebook types	Last version upda...
A Getting Started Guide For Microsoft Sentinel ML Notebooks Microsoft	Getting Started	10/06/21, 05:00 PM
A Getting Started Guide For PowerShell AML Notebooks Microsoft	Getting Started, PowerShell	04/26/21, 05:00 PM
A Tour of Cybersec notebook features Microsoft	Getting Started	09/14/21, 05:00 PM
Azure Synapse - Configure Azure ML and Azure Synapse Analytics NEW Microsoft	Getting Started, Configuration, Synapse	10/06/21, 05:00 PM
Azure Synapse - Detect potential network beaconing using Apache Spark NEW Microsoft	Hunting, Synapse	10/06/21, 05:00 PM
Configuring your Notebook Environment Microsoft	Configuration	04/26/21, 05:00 PM
Credential Scan on Azure Blob Storage Microsoft	Hunting	09/22/21, 05:00 PM
Credential Scan on Azure Data Explorer Microsoft	Hunting	09/22/21, 05:00 PM
Credential Scan on Azure Log Analytics Microsoft	Hunting	09/22/21, 05:00 PM
Entity Explorer - Account Microsoft	Hunting	04/26/21, 05:00 PM
Entity Explorer - Domain and URL Microsoft	Hunting	04/26/21, 05:00 PM
Entity Explorer - IP Address Microsoft	Hunting	04/26/21, 05:00 PM

A Getting Started Guide For Microsoft Sentinel ML ...

Microsoft
Created By

7 months ago
Last Version Update

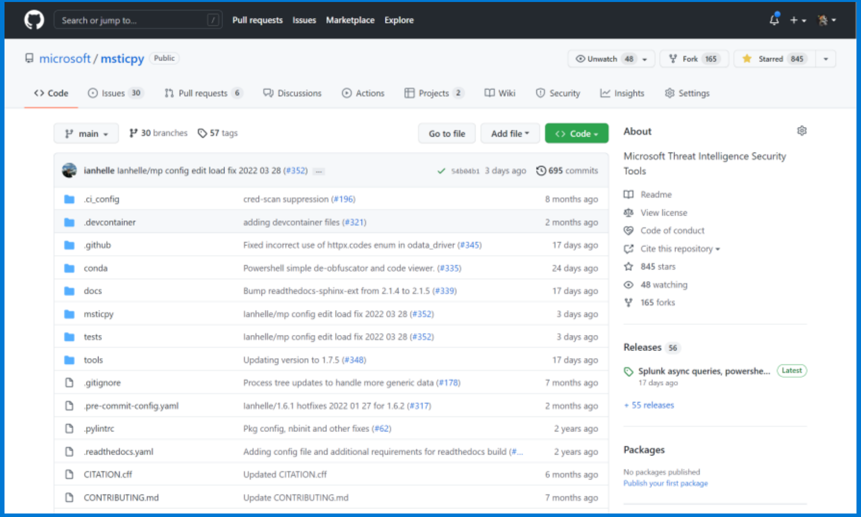
Description
This notebook guides you through the basic steps of using notebooks for security analysis. It covers all the basic steps you need to understand to start using the notebooks provided with Microsoft Sentinel. It also provides references to further documentation on the capabilities and features of notebooks. If this is your first-time running a notebook it is recommended that you run this notebook before running the others provided with Microsoft Sentinel. Note: This notebook will attempt to use SigninLogs data from your Microsoft Sentinel workspace, however if they are not available it will use sample data.

Utilized data types
SigninLogs 04/08/22, 09:35 AM

Notebook types
Getting Started

Getting Started with Azure Notebooks and Azure Sentinel

What is a Jupyter notebook?



ianhelle Ianhelle/mp config edit load fix 2022 03 28 (#352) ✓ 54b04b1 3 days ago 695 commits

.ci_config	cred-scan suppression (#196)	8 months ago
.devcontainer	adding devcontainer files (#321)	2 months ago
.github	Fixed incorrect use of httpx.codes enum in odata_driver (#345)	17 days ago
conda	Powershell simple de-obfuscator and code viewer. (#335)	24 days ago
docs	Bump readthedocs-sphinx-ext from 2.1.4 to 2.1.5 (#339)	17 days ago
msticpy	Ianhelle/mp config edit load fix 2022 03 28 (#352)	3 days ago
tests	Ianhelle/mp config edit load fix 2022 03 28 (#352)	3 days ago
tools	Updating version to 1.7.5 (#348)	17 days ago
.gitignore	Process tree updates to handle more generic data (#178)	7 months ago
.pre-commit-config.yaml	Ianhelle/1.6.1 hotfixes 2022 01 27 for 1.6.2 (#317)	2 months ago
.pylintrc	Pkg config, nbinit and other fixes (#62)	2 years ago
.readthedocs.yaml	Adding config file and additional requirements for readthedocs build (#...	2 years ago
CITATION.cff	Updated CITATION.cff	6 months ago
CONTRIBUTING.md	Update CONTRIBUTING.md	7 months ago

About
Microsoft Threat Intelligence Security Tools

Readme
View license
Code of conduct
Cite this repository
845 stars
48 watching
165 forks

Releases 56
Splunk async queries, powershe... Latest
17 days ago
+ 55 releases

Packages
No packages published
Publish your first package

What’s Included

Data Acquisition:

- Sentinel, Kusto, MDE, Graph
- Splunk
- More!

Data Enrichment

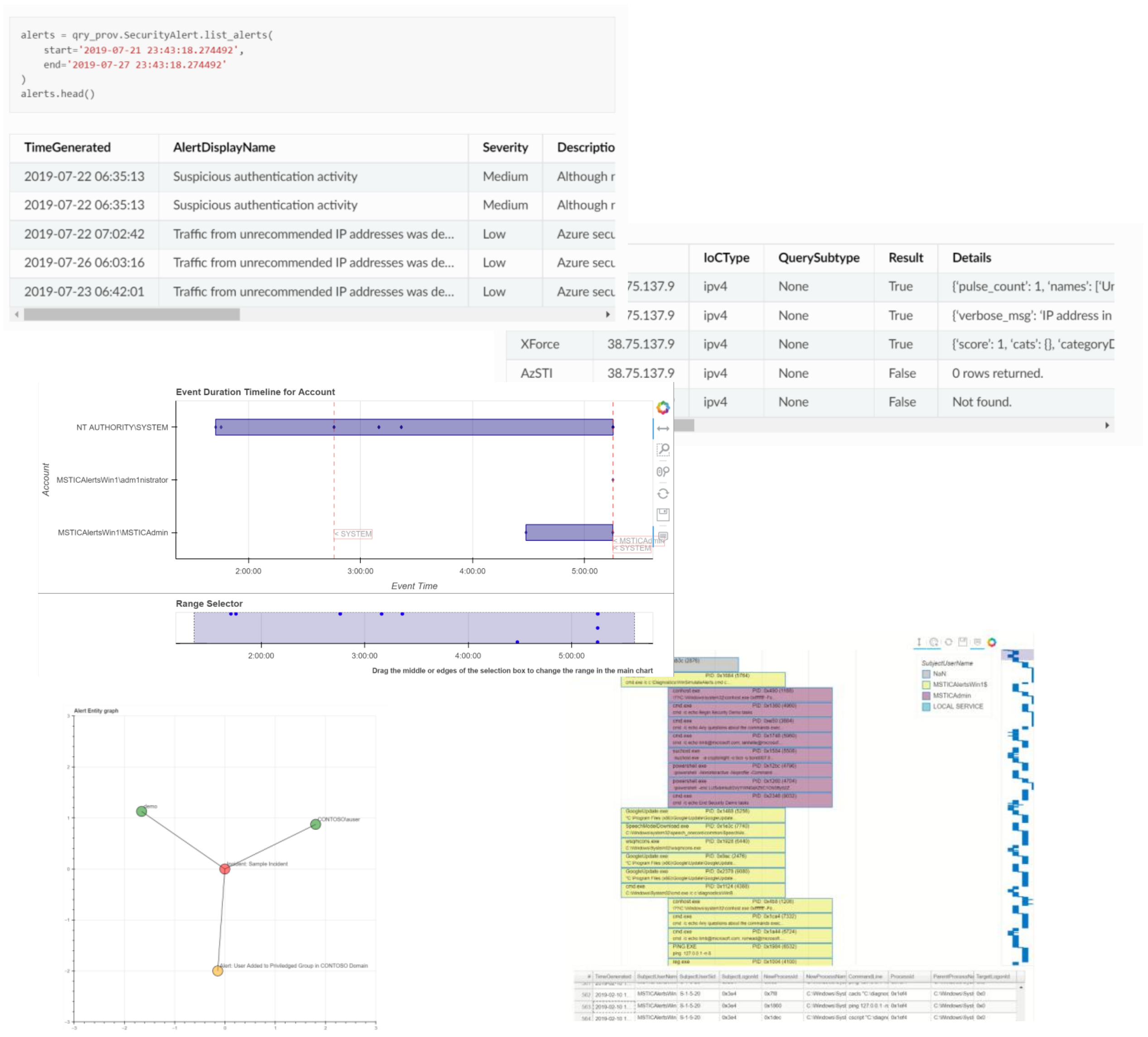
- Threat Intel lookups
- Context from Azure APIs
- WhoIs, GeoIP +

Analyzing Data

- Decode
- Extract
- ML

Displaying Data

- Timelines
- Process Trees
- Graphs



```
alerts = qry_prov.SecurityAlert.list_alerts(
    start='2019-07-21 23:43:18.274492',
    end='2019-07-27 23:43:18.274492'
)
alerts.head()
```

TimeGenerated	AlertDisplayName	Severity	Descriptio
2019-07-22 06:35:13	Suspicious authentication activity	Medium	Although r
2019-07-22 06:35:13	Suspicious authentication activity	Medium	Although r
2019-07-22 07:02:42	Traffic from unrecommended IP addresses was de...	Low	Azure secu
2019-07-26 06:03:16	Traffic from unrecommended IP addresses was de...	Low	Azure secu
2019-07-23 06:42:01	Traffic from unrecommended IP addresses was de...	Low	Azure secu

		IoCType	QuerySubtype	Result	Details
	75.137.9	ipv4	None	True	{'pulse_count': 1, 'names': ['Ur
	75.137.9	ipv4	None	True	{'verbose_msg': 'IP address in
XForce	38.75.137.9	ipv4	None	True	{'score': 1, 'cats': {}, 'categoryD
AzSTI	38.75.137.9	ipv4	None	False	0 rows returned.
		ipv4	None	False	Not found.

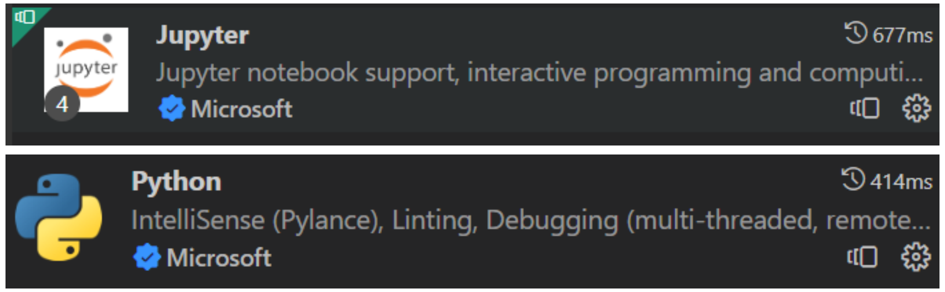
Event Duration Timeline for Account
NT AUTHORITY\SYSTEM
MSTICAlertsWin1\adm1nistrator
MSTICAlertsWin1\MSTICAdmin
Account
< SYSTEM
< MSTICAdmin
< SYSTEM
2:00:00
3:00:00
4:00:00
5:00:00
Event Time
Range Selector
2:00:00
3:00:00
4:00:00
5:00:00
Drag the middle or edges of the selection box to change the range in the main chart
Alert Entity graph
CONTOSO\user
Incident: Sample Incident
Alert: User Added to Privileged Group in CONTOSO Domain
SubjectUserName
NaN
MSTICAlertsWin1$
MSTICAdmin
LOCAL SERVICE

The Agenda

Section	Time
Intros & Setup	20 min
Intro to MSTICPy & Notebooks	30 min
MSTICPy Config	15 min
Break	15 min
Acquiring data with MSTICPy	20 min
Enrichment with MSTICPy	20 min
Jupyter notebooks advanced	15 min
Break	15 min
Data Analysis with MSTICpy	30 min
Data Visualization with MSTICPy	30 min
Putting it into operation	30 min

Setup

- VSCode Installed
  - Juypter and Python Extensions Installed
- Anaconda Installed
- Azure CLI Installed
- KeyVault Created
- Got API keys for:
  - Alienvault OTX
  - IBM Xforce
  - VirusTotal
  - GeoIPLite
- Check you have access – https://aka.ms/sentineldemo


Jupyter 677ms
Jupyter notebook support, interactive programming and computi...
Microsoft

Python 414ms
IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote...
Microsoft

Setup

- Open AnacondaPrompt
- Create new environment
  - \`conda create --name msticpy_training python=3.8\`
- Activate Environment
  - \`conda activate msticpy_training\`
- Clone the GitHub repo
  - \`git clone https://github.com/microsoft/msticpy-training\`
- Installed required packages
  - \`pip install -r msticpy-training\requirements.txt\`
- Navigate to our Workshop
  - \`cd msticpy-training\workshops\oct2022\`
- Run VSCode from here

Questions & Issues

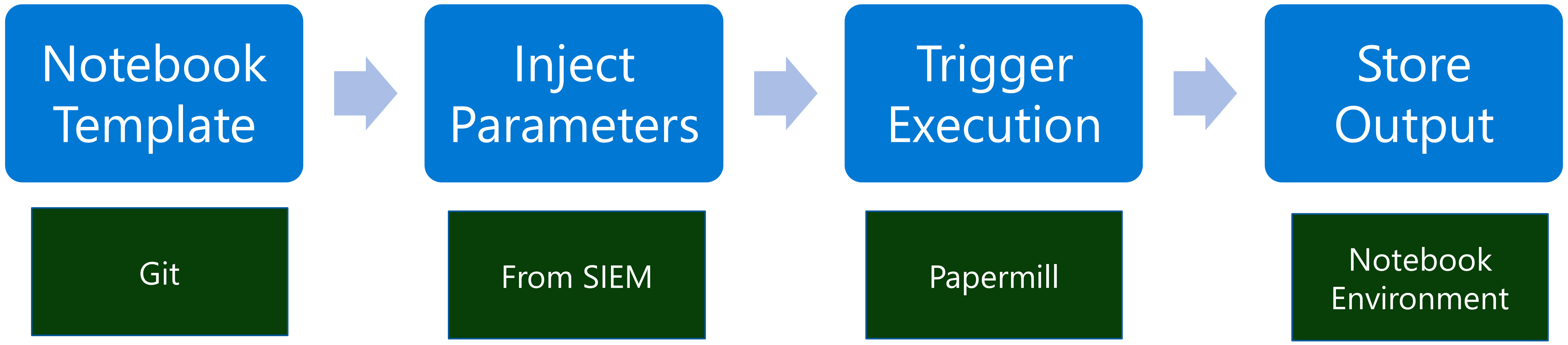
- Teams Channel - https://aka.ms/msticpy_training_teams
- Speak Up
- Breaks to Help Fix Issues

Go to VSCode and select the folder of the msticpy-training repository you just cloned. Open the IntroToMsticpy.ipynb file.

Putting it All Together

Operating Model

Automating notebooks execution allows the SOC to benefit from expert knowledge and process



Creating notebook templates

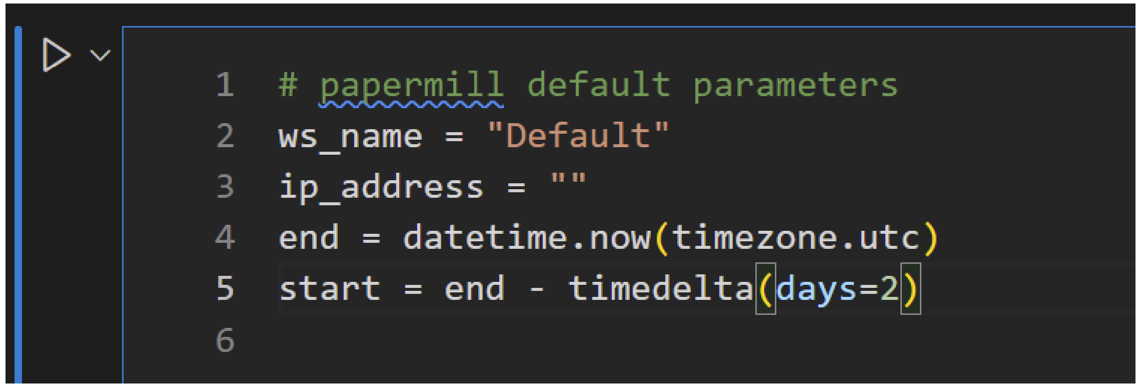
Version Control

Unattended Execution

Execution Options

- Papermill parameters
- Default execution path
- Resilient to errors
- Non-interactive

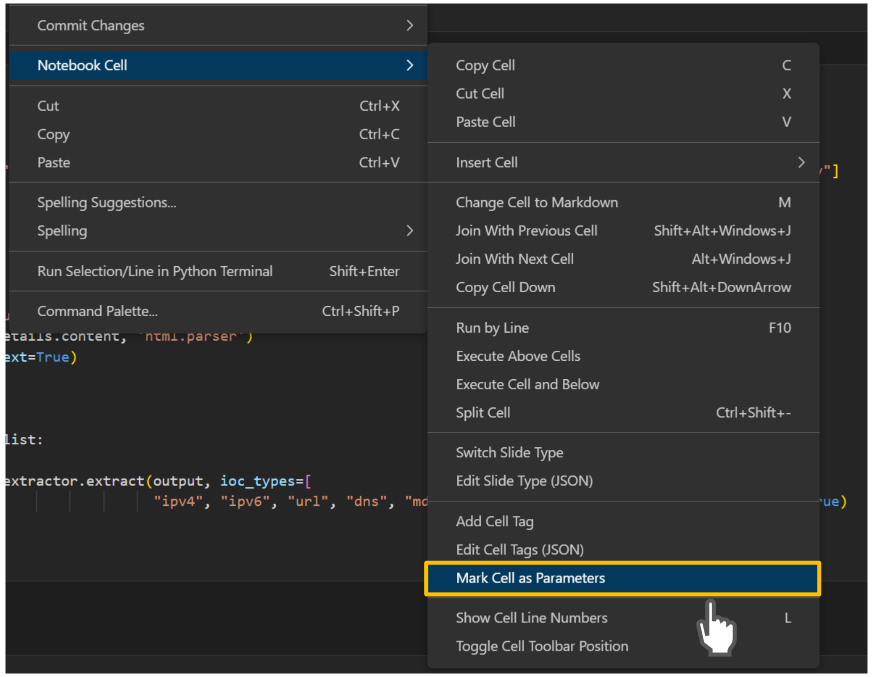
Adding Papermill Parameters



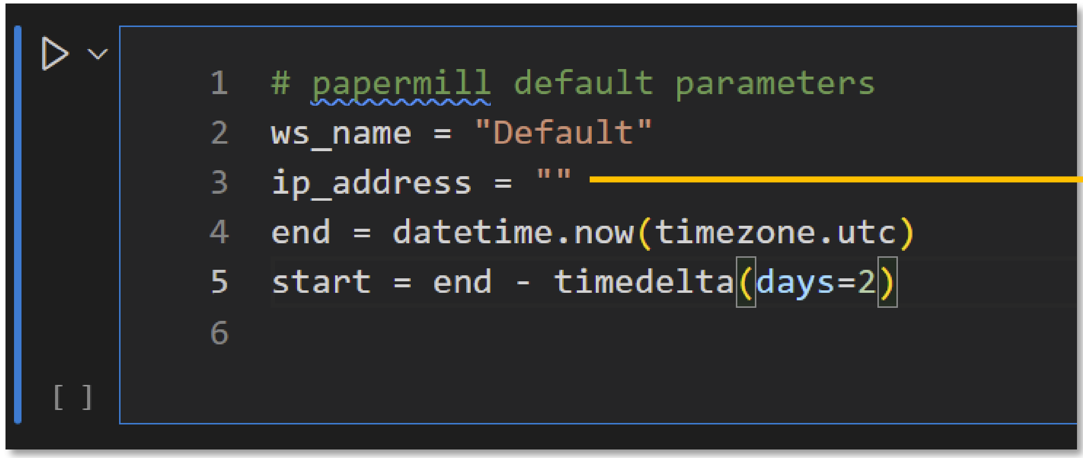
```python
# papermill default parameters
ws_name = "Default"
ip_address = ""
end = datetime.now(timezone.utc)
start = end - timedelta(days=2)

```

Create template cell for parameters.
Some or all values can have defaults.

Create “parameters” cell tag.

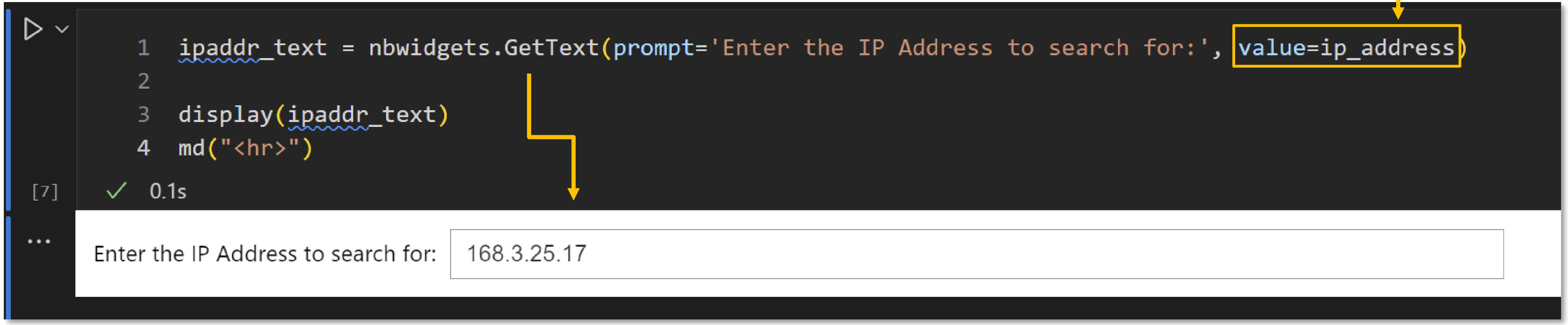


Allow interactive and automated use



```
# papermill default parameters
ws_name = "Default"
ip_address = ""
end = datetime.now(timezone.utc)
start = end - timedelta(days=2)

```

Notebook parameter cell



```
ipaddr_text = nbwidgets.GetText(prompt='Enter the IP Address to search for:', value=ip_address)

display(ipaddr_text)
md("<hr>")
```

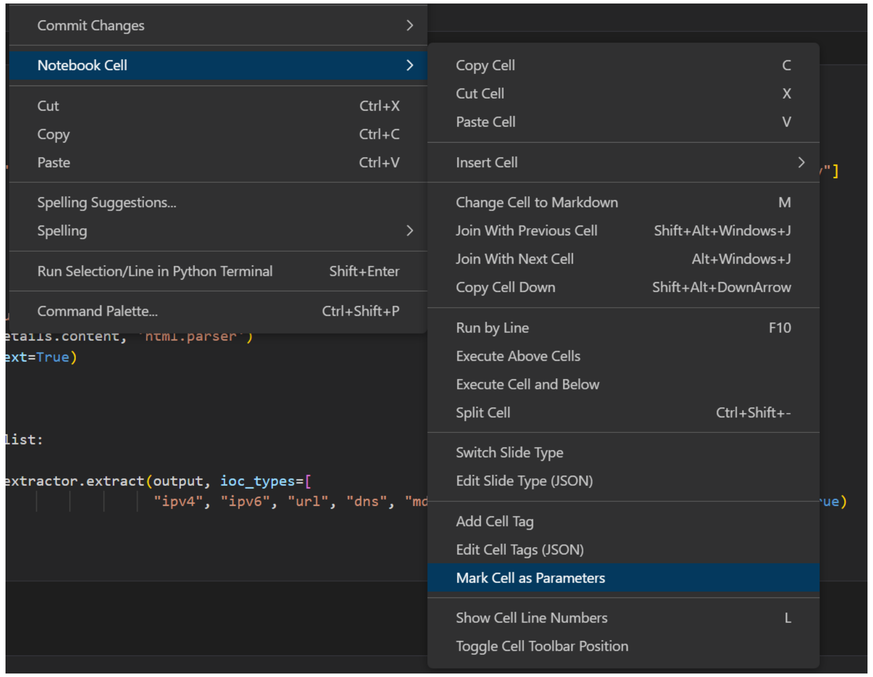
Enter the IP Address to search for: 168.3.25.17

Allow editing of parameters in non-blocking UI for interactive use

Exercise – Notebook Parameters

Open the `AutomatedNotebooks.ipynb` notebook

Right click on the cell to parameterize and select Notebook Cell > Mark Cell as Parameters



```
etails.content, 'html.parser')
ext=True)

list:

xtractor.extract(output, ioc_types=[
                    "ipv4", "ipv6", "url", "dns", "md
```

Injecting parameters

1. On the command line

```
$ papermill src/ip_addr.ipynb out/output.ipynb ⏎
    -p ip_address "128.1.2.3" ⏎
    -p start "2002-07-01 13:05" ⏎
    -p end "2002-07-02 13:05"  ⏎
```

2. In a yaml file

```
ip_address: 128.1.2.3
start: 2002-07-01 13:05
end: 2002-07-02 13:05
```

```
$ papermill src/ip_addr.ipynb out/output.ipynb ⏎
    -f params.yaml
```

3. From Python

```
return pm.execute_notebook(
    input_path=input_nb,
    output_path=output_nb,
    parameters=nb_params.papermill, # Python dict
    **nb_kwargs,
)
```

Exercise – Injecting Parameters

- Open up your Anaconda prompt.
- Attempt to inject parameters into your AutomatedNotebook.ipynb
  - `ip 115.43.212.159`
- Execute the notebook with these parameters and see what output we get

Triggering execution

Scheduled - daily health checks, watch lists

On demand - investigation/analysis tasks

Event triggered - incident/alert triage

You may need **all** of these

Triggering - implementation

Use a cloud service

- Databricks, Azure Synapse, Amazon Sagemaker, etc.
- Likely need to customize for event-triggering

Roll your own

- Cron/Windows job – schedule
- File drop – on demand
- Poller – event-triggered

Build a trigger API

- HTTP endpoint
- JSON parameters

Execution - authentication and secrets

Authentication can be tricky

Data store (queries)
Services (TI)

Use a cloud service identity

Store credentials in vault (e.g. Azure Key Vault)

Avoid passing secrets/credentials as Papermill params!!!

Storing and retrieving results

Azure blob

- Cheap!

Output format

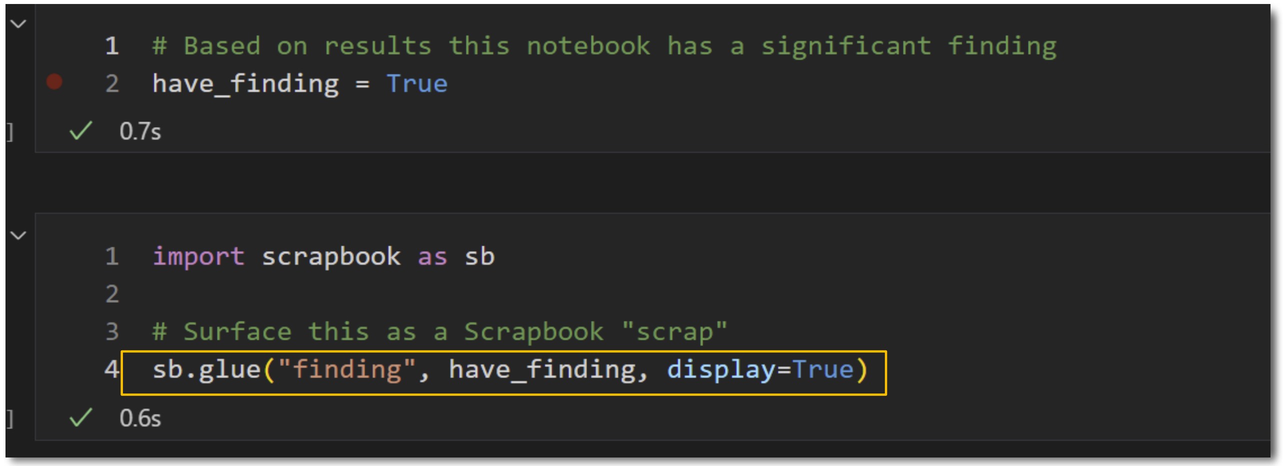
- Create output folder structure and naming scheme to organize your outputs

```
/output/2022/08/01/ip-context_124_34_13_59_{UUID}_{date}.ipynb
```

- *Papermill* can strip input code for easier reading
- Create html copies for notebooks with findings (*nbconvert*)

Storing output

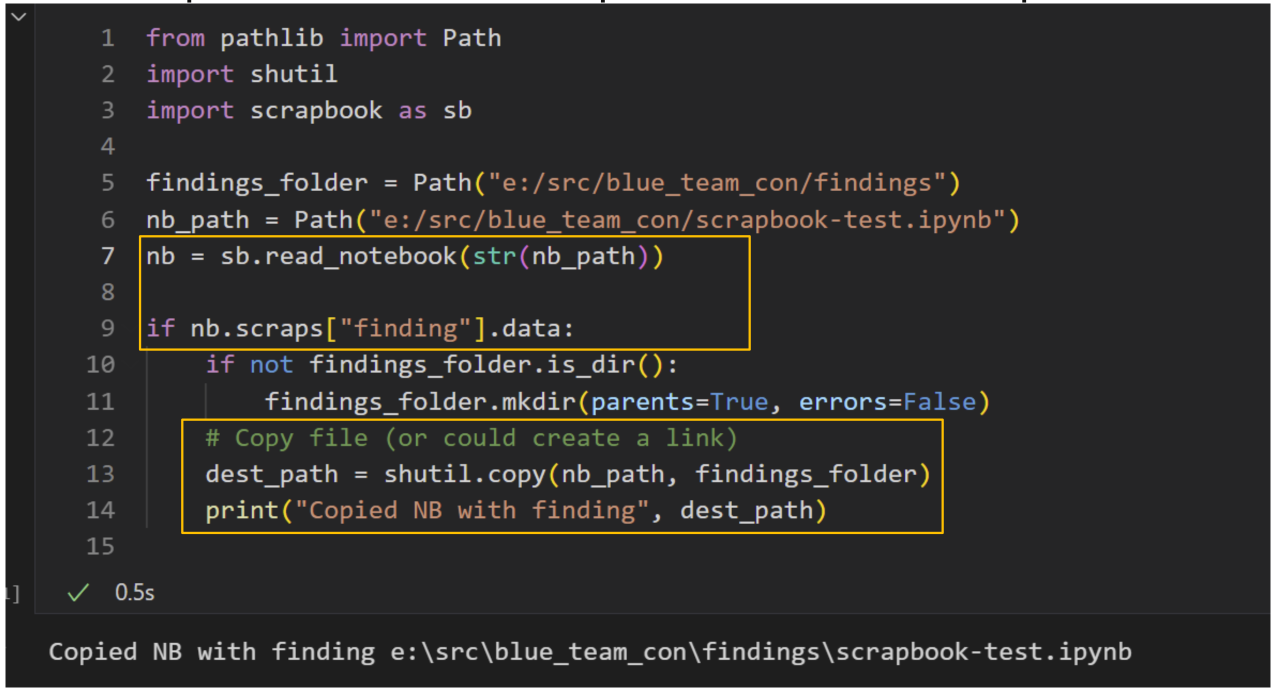
Identifying findings:
nteract Scrapbook



```
# Based on results this notebook has a significant finding
have_finding = True
```
✓ 0.7s

```
import scrapbook as sb

# Surface this as a Scrapbook "scrap"
sb.glue("finding", have_finding, display=True)
```
✓ 0.6s

Use scrapbook to check for presence of the scrap

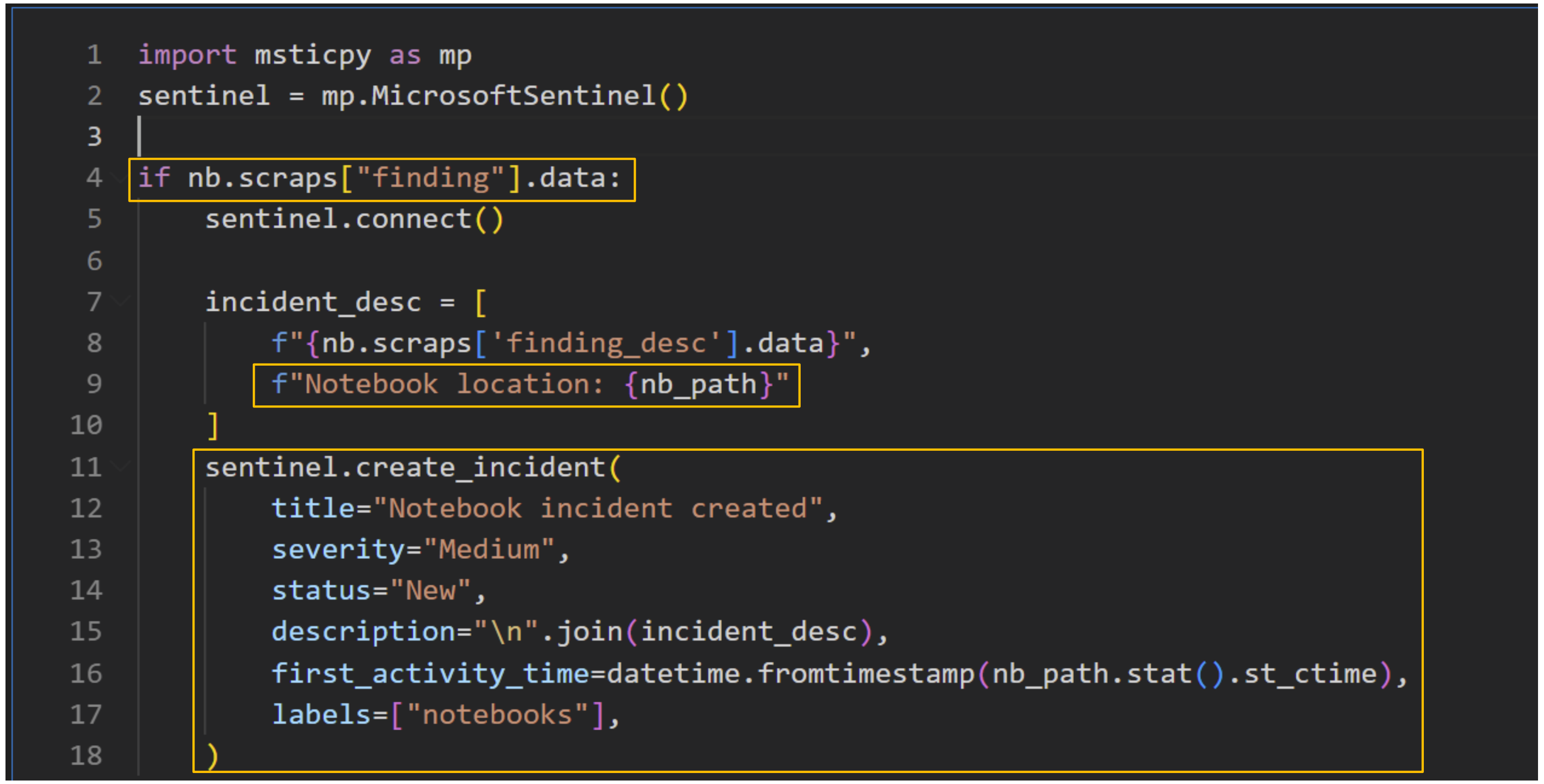


```
from pathlib import Path
import shutil
import scrapbook as sb

findings_folder = Path("e:/src/blue_team_con/findings")
nb_path = Path("e:/src/blue_team_con/scrapbook-test.ipynb")
nb = sb.read_notebook(str(nb_path))

if nb.scraps["finding"].data:
    if not findings_folder.is_dir():
        findings_folder.mkdir(parents=True, errors=False)
    # Copy file (or could create a link)
    dest_path = shutil.copy(nb_path, findings_folder)
    print("Copied NB with finding", dest_path)

```
✓ 0.5s

```
Copied NB with finding e:\src\blue_team_con\findings\scrapbook-test.ipynb
```

Storing output

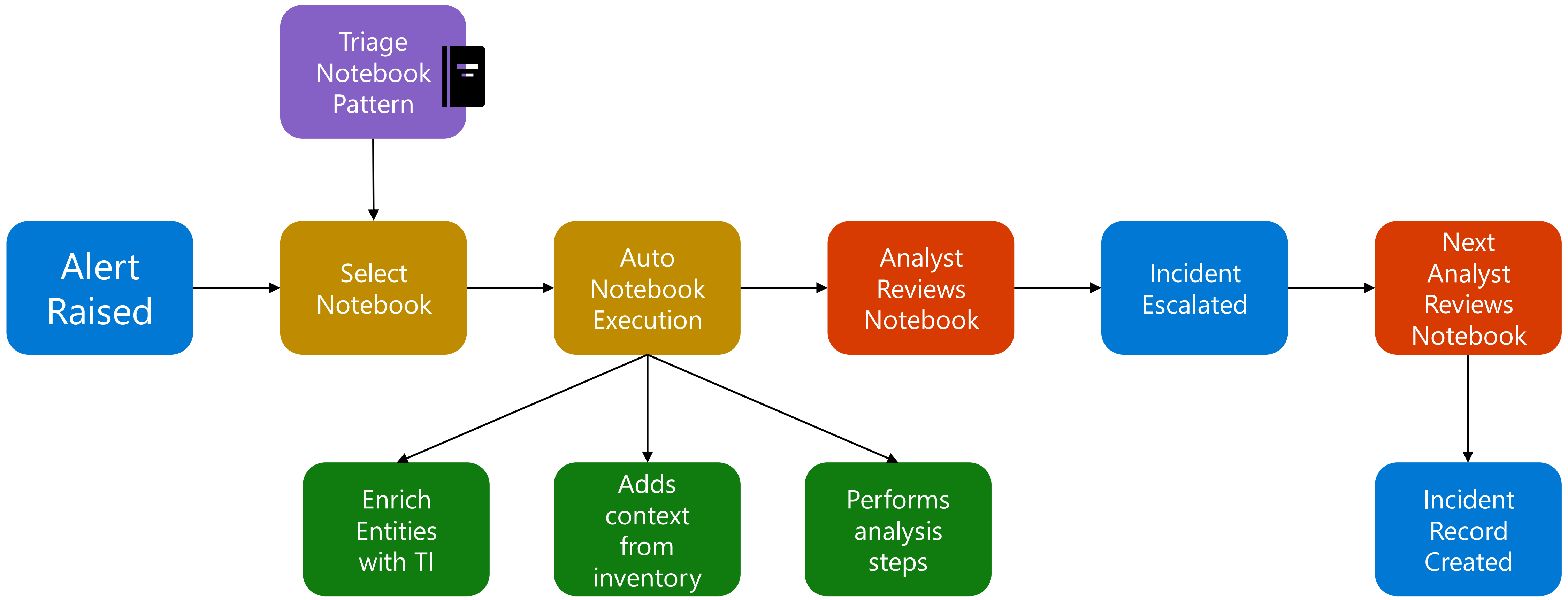
Identifying findings: create alert/incident



```python
import msticpy as mp
sentinel = mp.MicrosoftSentinel()

if nb.scraps["finding"].data:
    sentinel.connect()

    incident_desc = [
        f"{nb.scraps['finding_desc'].data}",
        f"Notebook location: {nb_path}"
    ]
    sentinel.create_incident(
        title="Notebook incident created",
        severity="Medium",
        status="New",
        description="\n".join(incident_desc),
        first_activity_time=datetime.fromtimestamp(nb_path.stat().st_ctime),
        labels=["notebooks"],
    )
```

Most incident management systems have equivalent mechanism

Notebooks for Alert Triage



Notebook automation examples

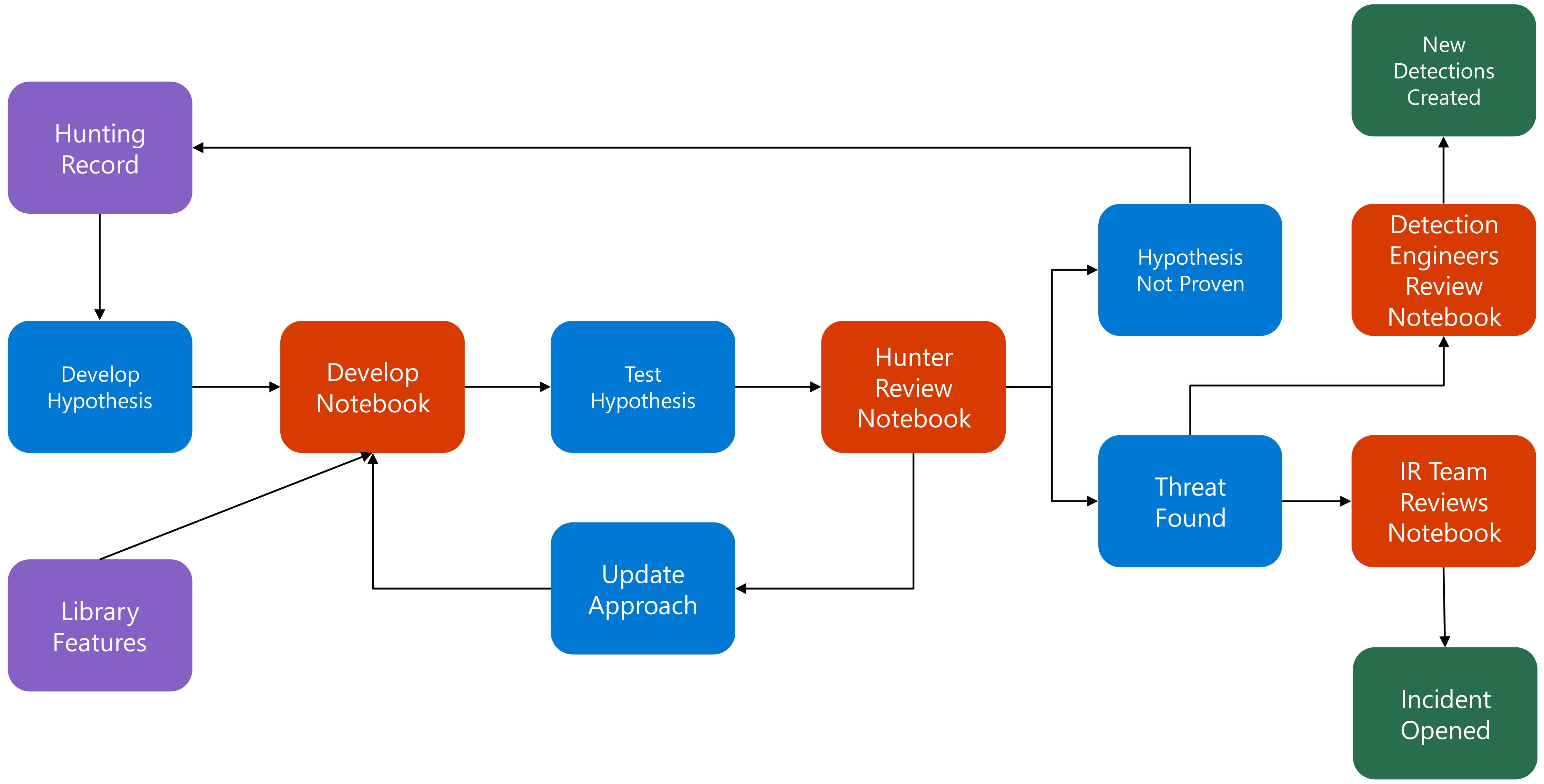
Big brother of the demo

- Software Defined Monitoring - Using Automated Notebooks and Azure Sentinel to Improve Sec Ops
- Create Azure VM to run notebooks triggered from incidents
- Should be adaptable to other cloud platforms

Our Demo

- Simple solution using Docker + Papermill
- Triggered by YAML parameters file
- Full source on GitHub (see refs)

Notebooks for Threat Hunting



Threat hunting requirements

Usually **ad hoc** but may contain some automated elements

Library support is crucial – make it easy to:

- Query and retrieve information
- Create visualizations
- Repeatable analysis and data extraction/transformation

Package common tasks in parameter-driven notebooks/notebooklets

Apply the same standards as automated notebooks:

- Version control processes (for library and building-block code)
- Output naming and storage

Final Exercise - Optional

- Create your own automated notebook
- Take what you have learnt today and create a notebook using MSTICpy that completes some task
- Parameterize the notebook
- Execute notebook with injected parameters
- Schedule execution for a future time

Find out more

- PyPi

https://pypi.org/project/msticpy/

- GitHub Code

https://github.com/microsoft/msticpy

- Issues

https://github.com/microsoft/msticpy/issues

- Plans

https://github.com/microsoft/msticpy/discussions

- ReadTheDocs

https://msticpy.readthedocs.io/en/latest/index.html

- msticpy@microsoft.com

Microsoft Security

Thank You