大型語言模型與資訊安全系統

Applying Large Language Models in Cybersecurity Systems

劉定睿

日期：2026-05-24

目錄

TASK 1: Introduction to Google Colab

Part 2 — Basic Python in Colab

Run the following code:

```
[1]
print("Hello, Google Colab!")
```

```
Hello, Google Colab!
```

```
[2]
a = 10
b = 5
print("Sum:", a + b)
```

```
Sum: 15
```

＋程式碼 ＋文字

Part 3 — Importing Libraries

Most data science tasks require Python libraries.

Example:

```
[3]
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

＋程式碼 ＋文字

Check the library version:

```
[4]
import pandas as pd
print(pd.__version__)
```

```
2.2.2
```

Part 4 — Uploading Data to Colab

Method 1: Upload Files from Local Computer

Run:

```
[5]
from google.colab import files
uploaded = files.upload()
```

```
選擇檔案 1.csv
1.csv(text/csv) – 63873 bytes, last modified: 2026/5/11 – 100% done
Saving 1.csv to 1.csv
```

After uploading, you can read the file.

Example (CSV file):

After uploading, you can read the file.

Example (CSV file):

```
[7]
import pandas as pd

data = pd.read_csv("1.csv")
data.head()
```

	@timestamp	_id	_index	_score	_type	agent.ephemeral_id	agent.hostname	agent.id	agent.name	agent.type	...	source.address	source.
0	Apr 4, 2025 @ 21:54:46.315	wF0VAZYBsR-CzrJnMV62	.ds-logs-auditd.log-default-2025.04.04-000001	–	_doc	9027fbed-4373-4ac6-a2bb-7ff36679e43c	appsrv06	7bd602d3-ca4f-4534-a9e5-85e7d219bcec	appsrv06	filebeat	...	192.168.211.69	192.168.211.
1	Apr 4, 2025 @ 21:54:42.883	dl0VAZYBsR-CzrJnJV7u	.ds-logs-auditd.log-default-2025.04.04-000001	–	_doc	9027fbed-4373-4ac6-a2bb-7ff36679e43c	appsrv06	7bd602d3-ca4f-4534-a9e5-85e7d219bcec	appsrv06	filebeat	...	192.168.211.69	192.168.211.
2	Apr 4, 2025 @ 21:54:40.119	G10VAZYBsR-CzrJnFI1N	.ds-logs-auditd.log-default-2025.04.04-000001	–	_doc	9027fbed-4373-4ac6-a2bb-7ff36679e43c	appsrv06	7bd602d3-ca4f-4534-a9e5-85e7d219bcec	appsrv06	filebeat	...	192.168.211.69	192.168.211.
3	Apr 4, 2025 @ 21:54:37.707	_V0VAZYBsR-CzrJnElxk	.ds-logs-auditd.log-default-2025.04.04-000001	–	_doc	9027fbed-4373-4ac6-a2bb-7ff36679e43c	appsrv06	7bd602d3-ca4f-4534-a9e5-85e7d219bcec	appsrv06	filebeat	...	192.168.211.69	192.168.211.
	Apr 4, 2025		.ds-logs-auditd.log-			9027fbed-4373-		7bd602d3-					

Part 5 — Loading Dataset from the Internet

You can load data directly from a URL.

Example:

```
[8]
0 秒
```

```
url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv"

import pandas as pd
data = pd.read_csv(url)

data.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

後續步驟： 使用 data 生成程式碼 New interactive sheet

Check dataset information:

```
[9]
0 秒
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

＋ 程式碼 ＋ 文字

Basic statistics:

```
[10]
0 秒
```

```
data.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Part 6 — Accessing Google Drive

Sometimes datasets are stored in Google Drive.

Mount Google Drive:

```
[11]
35 秒
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Example of reading data:

```
[12]
1 秒
```

```
data = pd.read_csv('/content/drive/MyDrive/1.csv')
data.head()
```

	@timestamp	_id	_index	_score	_type	agent.ephemeral_id	agent.hostname	agent.id	agent.name	agent.type	...	source.address	source.
0	Apr 4, 2025 @ 21:54:46.315	wF0VAZYBsR-CzrJnMV62	.ds-logs-auditd.log-default-2025.04.04-000001	–	_doc	9027fbed-4373-4ac6-a2bb-7ff36679e43c	appsrv06	7bd602d3-ca4f-4534-a9e5-85e7d219bcec	appsrv06	filebeat	...	192.168.211.69	192.168.211.
1	Apr 4, 2025 @ 21:54:42.883	dI0VAZYBsR-CzrJnJV7u	.ds-logs-auditd.log-default-2025.04.04-000001	–	_doc	9027fbed-4373-4ac6-a2bb-7ff36679e43c	appsrv06	7bd602d3-ca4f-4534-a9e5-85e7d219bcec	appsrv06	filebeat	...	192.168.211.69	192.168.211.
	Apr 4, 2025	G10VAZYBsR-	.ds-logs-auditd.log-			9027fbed-4373-		7bd602d3-ca4f-4534-					

Part 7 — Exporting / Saving Data

Save data to a CSV file:

```
[13] 0 秒
data.to_csv("processed_data.csv", index=False)
```

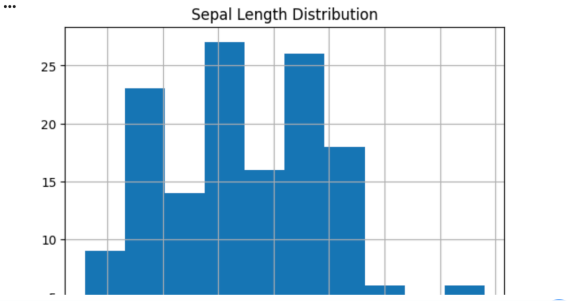
Download file to your computer:

```
[14] 0 秒
from google.colab import files
files.download("processed_data.csv")
```

Part 8 — Simple Data Visualization

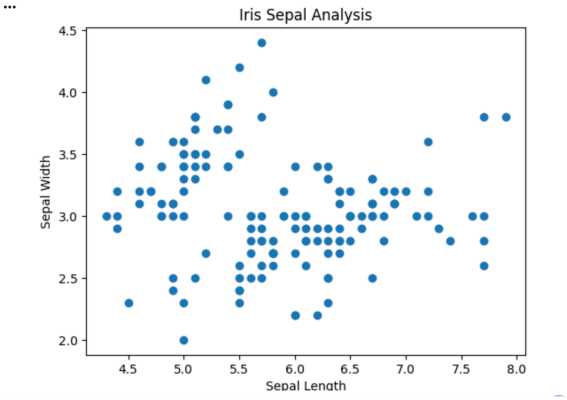
Example plot:

```
[42] 0 秒
import matplotlib.pyplot as plt
import pandas as pd

# 重新讀取鳶尾花資料以確保欄位存在
url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv"
iris_data = pd.read_csv(url)

iris_data['sepal_length'].hist()
plt.title("Sepal Length Distribution")
plt.show()
```



Scatter plot example:

```
[44] 0 秒
import matplotlib.pyplot as plt

# 使用先前載入的 iris_data 變數，而非被覆蓋的 data 變數
plt.scatter(iris_data['sepal_length'], iris_data['sepal_width'])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.title("Iris Sepal Analysis")
plt.show()
```



Part 9 — Installing Additional Libraries

Sometimes Colab does not have the required library.

Install with:

```
[18] 8 秒
!pip install seaborn
```

顯示隱藏的輸出內容

Then import it:

```
[19] 2 秒
import seaborn as sns
```

Part 10 — Exporting Data to PDF and JSON

In addition to CSV files, datasets or results can also be exported to other formats such as JSON and PDF. These formats are commonly used for data exchange and reporting.

10.1 Export Data to JSON

JSON (JavaScript Object Notation) is a lightweight format commonly used for APIs and web applications.

Example:

[20] 0 秒

```
data.to_json("dataset.json", orient="records")
```

Download the JSON file:

[21] 0 秒

```
from google.colab import files
files.download("dataset.json")
```

Load JSON again:

[22] 0 秒

```
import pandas as pd

data_json = pd.read_json("dataset.json")
data_json.head()
```

@timestamp	_id	_index	_score	_type	agent.ephemeral_id	agent.hostname	agent.id	agent.name	agent.type	...	source.address	source.

10.2 Export Results to PDF

To create a PDF report, we can use the reportlab library.

Install the library

[23] 6 秒

```
!pip install reportlab
```

顯示隱藏的輸出內容

Example: Generate a Simple PDF Report

[24] 0 秒

```
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer
from reportlab.lib.styles import getSampleStyleSheet

styles = getSampleStyleSheet()

content = []

content.append(Paragraph("Google Colab Data Report", styles['Title']))
content.append(Spacer(1,20))

content.append(Paragraph("Dataset Summary:", styles['Heading2']))
content.append(Paragraph(str(data.describe()), styles['BodyText']))

pdf = SimpleDocTemplate("data_report.pdf")
pdf.build(content)
```

Download the generated PDF:

[25] 0 秒

```
from google.colab import files
files.download("data_report.pdf")
```

Scenario 1 — Missing Library

Run the following code:

[45] 0 秒

```
import seaborn as sns

sns.load_dataset("iris")
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

Scenario 2 — Wrong File Path

Run:

Gemini

```
[49]
-import pandas as pd

-data = pd.read_csv("aaaaa.csv")
+# 修正檔名為實際存在於環境中的 1.csv
+data = pd.read_csv("1.csv")
+data.head()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipykernel_3386/814383188.py in <cell line: 0>()
      1 import pandas as pd
      2
----> 3 data = pd.read_csv("aaaaa.csv")

 4 frames
/usr/local/lib/python3.12/dist-packages/pandas/io/common.py in
get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text,
errors, storage_options)
    871         if ioargs.encoding and "b" not in ioargs.mode:
    872             # Encoding
--> 873             handle = open(
    874                 handle,
    875                 ioargs.mode,

FileNotFoundError: [Errno 2] No such file or directory: 'aaaaa.csv'
```

後續步驟： 說明錯誤

Gemini

取的安全性日誌（1.csv），而不是鳶尾花資料集。日誌資料中並沒有 `sepal_length` 這個欄位。我已將程式碼修正，改為引用正確的 `iris_data` 變數來繪製散佈圖。

D 請說明這個錯誤：

`FileNotFoundError: [Errno 2] No such file or directory: 'aaaaa.csv'`

這個錯誤 `FileNotFoundError` 表示程式找不到名為 `aaaaa.csv` 的檔案。這通常是因為檔名輸入錯誤或檔案尚未上傳。根據您的筆記本內容，您剛才上傳的檔案應該是 `1.csv`。我已經將程式碼修正為讀取正確的檔案名稱。

接受並執行 接受 取消

需要我幫忙建構什麼嗎？

Gemini 3 Flash

Gemini 可能會出錯，因此請查證回覆內容，並謹慎使用程式碼。 瞭解詳情

Scenario 3 — Syntax Error

Run:

Gemini

```
[50]
-for i in range(5)
+for i in range(5):
    print(i)
```

```
  File "/tmp/ipykernel_3386/3964378094.py", line 1
    for i in range(5)
                    ^
SyntaxError: expected ':'
```

後續步驟： 說明錯誤

Gemini

D 請說明這個錯誤：

`SyntaxError: expected ':'`

這個錯誤 `SyntaxError: expected ':'` 是因為在 Python 中，`for` 迴圈的語句結尾必須加上冒號（:）。這告訴 Python 下一行開始是迴圈的內容。我已經幫您在 `range(5)` 後面補上了冒號。

接受並執行 接受 取消

需要我幫忙建構什麼嗎？

Gemini 3 Flash

Gemini 可能會出錯，因此請查證回覆內容，並謹慎使用程式碼。 瞭解詳情

Part 12 — Practice: Checking Timeline

In this section, you will practice a **simple timeline checking**: from system log data.

Attack timeline reconstruction is a common task in **digital forensics and incident response**, where analysts examine logs to understand:

- When the attack started
- What actions the attacker performed
- Which systems were accessed
- The sequence of events during the attack

```
[29]
0 秒
        "Failed login attempt",
        "Failed login attempt",
        "User login success",
        "File download",
        "Privilege escalation"
    ],
    "user": [
        "admin",
        "unknown",
        "unknown",
        "admin",
        "admin",
        "admin"
    ]
}

logs = pd.DataFrame(log_data)
logs
```

	timestamp	event	user
0	2025-03-10 09:00:12	User login	admin
1	2025-03-10 09:05:33	Failed login attempt	unknown
2	2025-03-10 09:10:02	Failed login attempt	unknown
3	2025-03-10 09:12:45	User login success	admin
4	2025-03-10 09:20:11	File download	admin
5	2025-03-10 09:35:40	Privilege escalation	admin

後續步驟： 使用 logs 生成程式碼 New interactive sheet

Step 2 — Convert Timestamp Format

Convert the timestamp column into a proper datetime format.

[30] 0 秒

```
logs["timestamp"] = pd.to_datetime(logs["timestamp"])
logs
```

	timestamp	event	user
0	2025-03-10 09:00:12	User login	admin
1	2025-03-10 09:05:33	Failed login attempt	unknown
2	2025-03-10 09:10:02	Failed login attempt	unknown
3	2025-03-10 09:12:45	User login success	admin
4	2025-03-10 09:20:11	File download	admin
5	2025-03-10 09:35:40	Privilege escalation	admin

後續步驟： 使用 logs 生成程式碼 New interactive sheet

Step 3 — Sort Events by Time

Reconstruct the timeline by sorting the events.

[31] 0 秒

```
timeline = logs.sort_values(by="timestamp")
timeline
```

	timestamp	event	user
0	2025-03-10 09:00:12	User login	admin
1	2025-03-10 09:05:33	Failed login attempt	unknown
2	2025-03-10 09:10:02	Failed login attempt	unknown
3	2025-03-10 09:12:45	User login success	admin
4	2025-03-10 09:20:11	File download	admin
5	2025-03-10 09:35:40	Privilege escalation	admin

後續步驟： 使用 timeline 生成程式碼 New interactive sheet

Step 4 — Checking the failed login attempts, privilege escalation and unusual downloads

Suspicious activities might include.....

[32] 0 秒

```
ourfocus = logs[logs["event"].str.contains("Failed|Privilege", case=False)]
ourfocus
```

	timestamp	event	user
1	2025-03-10 09:05:33	Failed login attempt	unknown
2	2025-03-10 09:10:02	Failed login attempt	unknown
5	2025-03-10 09:35:40	Privilege escalation	admin

後續步驟： 使用 ourfocus 生成程式碼 New interactive sheet

Step 5 — Visualize the Attack Timeline

Display the ordered timeline.

[33] 0 秒

```
for index, row in timeline.iterrows():
    print(row["timestamp"], "-", row["event"], "-", row["user"])
```

```
2025-03-10 09:00:12 - User login - admin
2025-03-10 09:05:33 - Failed login attempt - unknown
2025-03-10 09:10:02 - Failed login attempt - unknown
2025-03-10 09:12:45 - User login success - admin
2025-03-10 09:20:11 - File download - admin
2025-03-10 09:35:40 - Privilege escalation - admin
```

TASK 2: Additional Practice

Download & Extract Dataset

[34]

```python
import requests
from zipfile import ZipFile
from io import BytesIO

# Dataset URL
url = "https://raw.githubusercontent.com/OTRF/Security-Datasets/master/datasets/atomic/windows/lateral_movement/host/empire_psexec_dcerpc_tcp_svcctl.zip"

# Download ZIP dataset
zipFileRequest = requests.get(url)

# Extract ZIP file
zipFile = ZipFile(BytesIO(zipFileRequest.content))

# Extract JSON dataset
datasetJSONPath = zipFile.extract(zipFile.namelist()[0])

print("Dataset extracted successfully:")
print(datasetJSONPath)
```

```
Dataset extracted successfully:
/content/empire_psexec_dcerpc_tcp_svcctl_2020-09-20121608.json
```

Read JSON Dataset

[35]

```python
import pandas as pd
from pandas.io import json

# Read JSON logs
df = json.read_json(
    path_or_buf=datasetJSONPath,
    lines=True
)

# Display dataset shape
print(df.shape)

# Preview first rows
df.head()
```

tID	ProviderGuid	ExecutionProcessID	Channel	host	AccountType	UserID	...	MessageNumber	ScriptBlockText	MessageTotal	ScriptBl
10	{5770385F–C22A–43E0–BF4C–06F5698FFBD9}	9848	Microsoft–Windows–Sysmon/Operational	wec.internal.cloudapp.net	User	S–1–5–18	...	NaN	NaN	NaN	
10	{5770385F–C22A–43E0–BF4C–06F5698FFBD9}	9848	Microsoft–Windows–Sysmon/Operational	wec.internal.cloudapp.net	User	S–1–5–18	...	NaN	NaN	NaN	
10	{5770385F–C22A–43E0–BF4C–06F5698FFBD9}	9848	Microsoft–Windows–Sysmon/Operational	wec.internal.cloudapp.net	User	S–1–5–18	...	NaN	NaN	NaN	
	{5770385F–										

Explore Available Channels

[36]

```python
df['Channel'].value_counts()
```

Channel	count
Microsoft–Windows–Sysmon/Operational	2941
Windows PowerShell	440
security	383
Microsoft–Windows–PowerShell/Operational	354
Security	216
System	10
Microsoft–Windows–WMI–Activity/Operational	3
Microsoft–Windows–TaskScheduler/Operational	1

dtype: int64

Explore Event IDs

[37]
```
df['EventID'].value_counts().head(20)
```

EventID	count
10	1493
12	718
800	424
7	357
4103	350
13	287
4658	134
5158	91
5156	83
4656	69
4690	67
4663	49
4703	37
9	26
11	25
600	14

Analytic I — Local Service Installation

Look for new services being created in the environment.

[38]
```
analytic_1 = (
    df[
        [
            '@timestamp',
            'Hostname',
            'SubjectUserName',
            'ServiceName',
            'ServiceType',
            'ServiceStartType',
            'ServiceAccount'
        ]
    ]
    [
        (df['Channel'].str.lower() == 'security')
        & (df['EventID'] == 4697)
    ]
)

analytic_1.head(10)
```

	@timestamp	Hostname	SubjectUserName	ServiceName	ServiceType	ServiceStartType	ServiceAccount
1212	2020-09-20T16:16:58.214Z	WORKSTATION6.theshire.local	pgustavo	Updater	0x10	3.0	LocalSystem

[39]
```
        [
            '@timestamp',
            'Hostname',
            'TargetUserName',
            'TargetLogonId',
            'IpAddress',
            'LogonType'
        ]
    ]
    [
        (df['Channel'].str.lower() == 'security')
        & (df['EventID'] == 4624)
        & (df['LogonType'] == 3)
        & (~df['TargetUserName'].str.endswith('$', na=False))
    ]
)

# Correlate service creation with network logon
remote_service_creation = pd.merge(
    serviceInstallDf,
    networkLogonDf,
    left_on='SubjectLogonId',
    right_on='TargetLogonId',
    how='inner'
)

remote_service_creation.head(10)
```

	@timestamp_x	Hostname_x	SubjectUserName	SubjectLogonId	ServiceName	ServiceType	@timestamp_y	Hostname_y	TargetUserNam
0	2020-09-20T16:16:58.214Z	WORKSTATION6.theshire.local	pgustavo	0x2990d17	Updater	0x10	2020-09-20T16:16:58.212Z	WORKSTATION6.theshire.local	pgustav

Export Results to CSV

[40] 0 秒

```
remote_service_creation.to_csv(
    "remote_service_creation_results.csv",
    index=False
)

print("Results exported successfully.")
```

```
Results exported successfully.
```

Download CSV File in Colab

[41] 0 秒

```
from google.colab import files

files.download("remote_service_creation_results.csv")
```

TASK 3: Timeline Reconstruction Tools in Digital Forensics

Tool Selection

- Timesketch

Installation or Access

- `!pip install timesketch`

Basic Analysis Exercise

[2] ✓ 11 秒

```
import requests
from io import BytesIO
# 安裝支援 7z 格式的函式庫
!pip install py7zr
import py7zr

# Dataset URL
url = "https://github.com/SecurityNik/CTF/raw/refs/heads/main/Silience_of_the_RAM-Tushar/Event%20logs.7z"

# Download 7z dataset
response = requests.get(url)

# Extract 7z file
with py7zr.SevenZipFile(BytesIO(response.content), mode='r') as archive:
    archive.extractall(path="/content")
    file_names = archive.getnames()

print("Dataset extracted successfully:")
print(file_names[0])
```

顯示隱藏的輸出內容

[4] ✓ 5 秒

```
import pandas as pd
!pip install evtx
from evtx import PyEvtxParser
import json

def convert_evtx_to_csv(evtx_path, csv_path):
    parser = PyEvtxParser(evtx_path)
    # Extract records as JSON and parse them into a list of dicts
    records = [json.loads(r["data"])["Event"] for r in parser.records_json()]

    # Flatten and save to CSV using Pandas
    df = pd.json_normalize(records)
    df.to_csv(csv_path, index=False)

# 修正路徑：將 "Event_log" 改為 "Event logs"
convert_evtx_to_csv("Event logs/sysmon-logs.evtx", "output.csv")
```

```
Requirement already satisfied: evtx in /usr/local/lib/python3.12/dist-packages (0.11.0)
```

在將 Sysmon 日誌轉換為 CSV 後，我們可以使用 Pandas 進行時間軸分析，找出可疑的系統活動。

[7] ✓ 0 秒

```
import pandas as pd

# 載入資料
df_sysmon = pd.read_csv("output.csv")

# 轉換時間格式以確保排序正確（假設欄位名稱為 System.TimeCreated_Xml 或類似名稱）
# 注意：具體欄位名稱取決於 evtx 轉換後的結構，通常包含 'System.TimeCreated' 或 'TimeCreated'
time_col = [col for col in df_sysmon.columns if 'TimeCreated' in col][0]
df_sysmon[time_col] = pd.to_datetime(df_sysmon[time_col])

# 依時間排序
df_sysmon = df_sysmon.sort_values(by=time_col)

print(f"日誌範圍：{df_sysmon[time_col].min()} 到 {df_sysmon[time_col].max()}")
print(f"總事件數：{len(df_sysmon)}")

# 顯示前 5 筆記錄
display(df_sysmon.head())
```

```
日誌範圍：2025-12-03 05:55:04.700447+00:00 到 2025-12-03 07:13:51.283230+00:00
總事件數：1023
```

	#attributes.xmlns	System.Provider.#attributes.Name	System.Provider.#attributes.Guid	System

日誌範圍：2025-12-03 05:55:04.700447+00:00 到 2025-12-03 07:13:51.283230+00:00
總事件數：1023

	#attributes.xmlns	System.Provider.#attributes.Name	System.Provider.#attributes.Guid	System
0	http://schemas.microsoft.com/win/2004/08/event...	Microsoft-Windows-Sysmon	5770385F-C22A-43E0-BF4C-06F5698FFBD9	
1	http://schemas.microsoft.com/win/2004/08/event...	Microsoft-Windows-Sysmon	5770385F-C22A-43E0-BF4C-06F5698FFBD9	
2	http://schemas.microsoft.com/win/2004/08/event...	Microsoft-Windows-Sysmon	5770385F-C22A-43E0-BF4C-06F5698FFBD9	
3	http://schemas.microsoft.com/win/2004/08/event...	Microsoft-Windows-Sysmon	5770385F-C22A-43E0-BF4C-06F5698FFBD9	
4	http://schemas.microsoft.com/win/2004/08/event...	Microsoft-Windows-Sysmon	5770385F-C22A-43E0-BF4C-06F5698FFBD9	

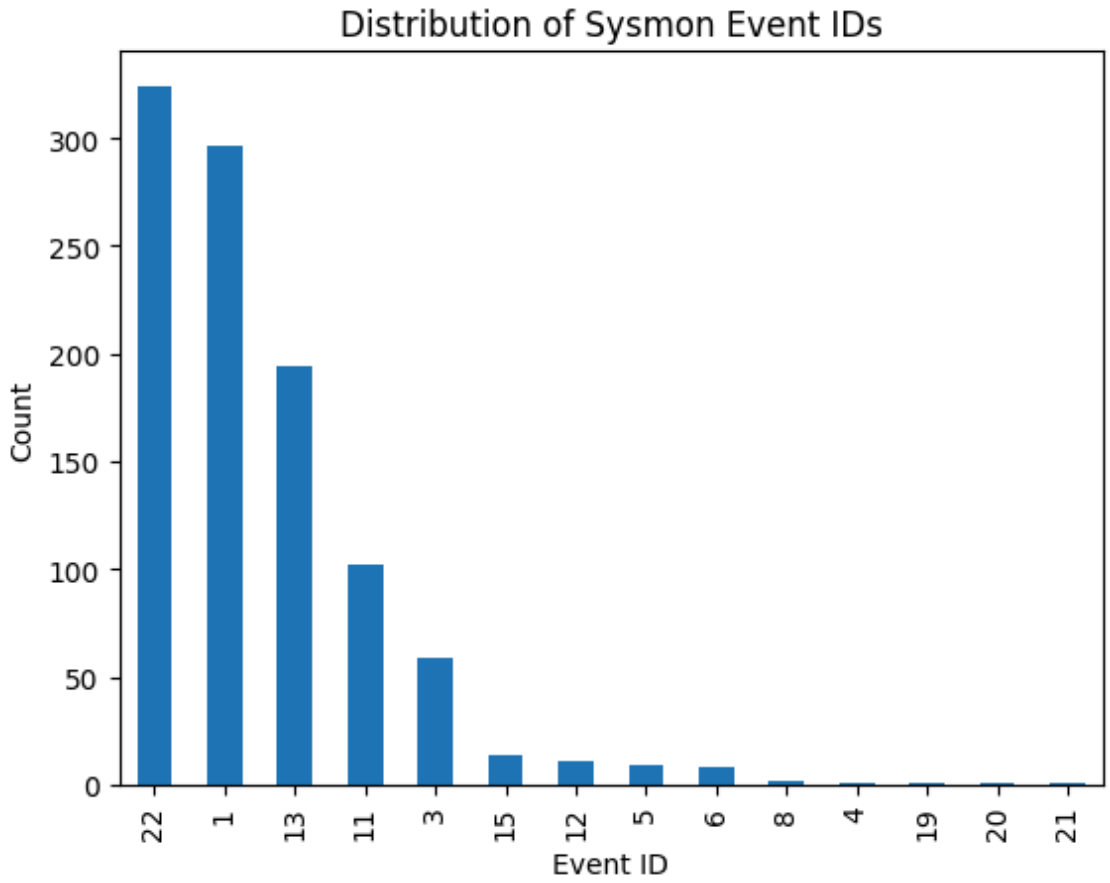
5 rows × 89 columns

統計 Event ID 分佈

Sysmon 的 Event ID 非常重要（例如 ID 1 是程序建立，ID 3 是網路連線）。

```
event_id_col = [col for col in df_sysmon.columns if 'EventID' in col][0]
event_counts = df_sysmon[event_id_col].value_counts()

print("Event ID 統計: ")
print(event_counts)

# 繪製簡單的統計圖
import matplotlib.pyplot as plt
event_counts.plot(kind='bar', title='Distribution of Sysmon Event IDs')
plt.xlabel('Event ID')
plt.ylabel('Count')
plt.show()
```

```
Event ID 統計：
System.EventID
22    324
1     296
13    194
11    102
3      59
15     14
12     11
5       9
6       8
8       2
4       1
19      1
20      1
21      1
Name: count, dtype: int64
```



深入分析：程序建立 (Event ID 1)

此事件紀錄了系統中執行的所有程序與指令。我們特別關注 `CommandLine` 欄位，查看是否有異常指令。

```
# 過濾出 Event ID 1 (Process Creation)
df_proc = df_sysmon[df_sysmon['System.EventID'] == 1].copy()

# 尋找包含 CommandLine 的欄位名稱 (通常在 EventData 中)
cmd_col = [col for col in df_proc.columns if 'CommandLine' in col][0]
image_col = [col for col in df_proc.columns if 'Image' in col and 'Source' not in col][0]

# 顯示前 10 筆程序建立紀錄，重點查看指令內容
print("程序建立紀錄 (前 10 筆): ")
display(df_proc[[time_col, image_col, cmd_col]].head(10))
```

```
... 程序建立紀錄 (前 10 筆)：
```

	System.TimeCreated.#attributes.SystemTime	EventData.Image	EventData.C
0	2025-12-03 05:55:04.700447+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	"C:\Program Files (x86)\Microsoft\Ed
1	2025-12-03 05:55:04.737985+00:00	C:\Windows\System32\svchost.exe	C:\Windows\system32\sv L
2	2025-12-03 05:55:05.356322+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	"C:\Program Files (x86)\Microsoft\Ed
7	2025-12-03 05:55:31.958187+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	"C:\Program Files (x86)\Microsoft\Ed
9	2025-12-03 05:55:50.055738+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	"C:\Program Files (x86)\Microsoft\Ed
11	2025-12-03 05:56:09.324939+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	"C:\Program Files (x86)\Microsoft\Ed
15	2025-12-03 05:56:09.533078+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	"C:\Program Files (x86)\Microsoft\Ed
16	2025-12-03 05:56:09.886602+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	"C:\Program Files (x86)\Microsoft\Ed
27	2025-12-03 05:56:16.361424+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	"C:\Program Files (x86)\Microsoft\Ed
36	2025-12-03 05:56:19.048760+00:00	C:\Users\Vic\AppData\Local\Microsoft\OneDrive\...	"C:\Users\Vic\AppData\Local\Microso

篩選可疑活動：PowerShell 與 Cmd

攻擊者經常使用這些工具來下載惡意軟體或進行橫向移動。

```
# 篩選包含 powershell 或 cmd 的指令
suspicious_cmds = df_proc[df_proc[cmd_col].str.contains('powershell|cmd.exe|base64|bypass', case=False, na=False)]

print(f"發現 {len(suspicious_cmds)} 筆可疑的指令執行：")
display(suspicious_cmds[[time_col, image_col, cmd_col]])
```

```
發現 27 筆可疑的指令執行：
```

	System.TimeCreated.#attributes.SystemTime	EventData.Image	Eve
55	2025-12-03 05:58:59.921400+00:00	C:\Windows\System32\cmd.exe	C:\Windc
56	2025-12-03 05:59:05.706086+00:00	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	
184	2025-12-03 06:37:09.018350+00:00	C:\Windows\System32\cmd.exe	cmd.exe /c "echo
194	2025-12-03 06:39:48.158238+00:00	C:\Windows\System32\cmd.exe	cmd.exe /c "echo
206	2025-12-03 06:44:15.564380+00:00	C:\Windows\System32\cmd.exe	cmd.exe /c "echo
355	2025-12-03 06:50:04.578841+00:00	C:\Windows\System32\cmd.exe	C:\Windows\system32\cmd.ex
364	2025-12-03 06:50:04.652176+00:00	C:\Windows\System32\cmd.exe	C:\Windows\syst
753	2025-12-03 06:51:13.654325+00:00	C:\Windows\System32\cmd.exe	c:\windc
789	2025-12-03 06:53:45.001095+00:00	C:\Windows\SysWOW64\cmd.exe	cmd.exe /c "echo
794	2025-12-03 06:53:45.358407+00:00	C:\Windows\SysWOW64\cmd.exe	C:\Windows\system32\cm
795	2025-12-03 06:53:45.435260+00:00	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\pow...	
802	2025-12-03 06:53:49.029247+00:00	C:\Windows\SysWOW64\cmd.exe	cmd.
804	2025-12-03 06:53:49.628678+00:00	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\pow...	powershell.e

分析：網路連線 (Event ID 3)

這可以幫助我們判斷受感染的主機是否在與外部 C2 (Command and Control) 伺服器通訊。

```
# 過濾出 Event ID 3 (Network Connection)
df_net = df_sysmon[df_sysmon['System.EventID'] == 3].copy()

# 找出目的 IP 與 目的 端口的欄位
dst_ip_col = [col for col in df_net.columns if 'DestinationIp' in col][0]
dst_port_col = [col for col in df_net.columns if 'DestinationPort' in col][0]

# 統計最常連線的目的地
print("最常連線的目的地 IP：")
print(df_net[dst_ip_col].value_counts().head(10))

display(df_net[[time_col, image_col, dst_ip_col, dst_port_col]].head(10))
```

```
最常連線的目的地 IP：
EventData.DestinationIp
10.0.0.84                    12
fe80:0:0:0:2069:2960:3dfc:d5bb    8
10.0.0.118                    7
ff02:0:0:0:0:0:0:c            4
239.255.255.250               4
173.236.138.116               3
0:0:0:0:0:0:0:1               2
72.136.197.42                 2
64.71.255.204                 2
ff02:0:0:0:0:0:1:3            2
Name: count, dtype: int64
```

	System.TimeCreated.#attributes.SystemTime	EventData.Image	EventData.DestinationIp	Event
24	2025-12-03 05:56:10.189503+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	10.0.0.118	
25	2025-12-03 05:56:10.189518+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	10.0.0.118	
41	2025-12-03 05:56:36.600554+00:00	C:\Users\Vic\Downloads\AdobeFlashUpdate.exe	10.0.0.118	
80	2025-12-03 06:05:08.106220+00:00	C:\Users\Vic\Downloads\SysinternalsSuite\proce...	23.10.154.237	
81	2025-12-03 06:05:08.607647+00:00	C:\Users\Vic\Downloads\SysinternalsSuite\proce...	72.136.197.42	
173	2025-12-03 06:32:36.450965+00:00	C:\Windows\System32\OpenSSH\sshd.exe	10.0.0.84	
217	2025-12-03 06:50:01.677974+00:00	C:\Windows\system32\svchost.exe	224.0.0.251	
218	2025-12-03 06:50:01.678262+00:00	C:\Windows\system32\svchost.exe	10.0.0.84	
219	2025-12-03 06:50:01.678515+00:00	C:\Windows\system32\svchost.exe	ff02:0:0:0:0:0:0:fb	
220	2025-12-03 06:50:01.678755+00:00	C:\Windows\system32\svchost.exe	ff02:0:0:0:0:0:1:3	

分析：檔案建立 (Event ID 11)

查看是否有檔案被下載或建立在可疑目錄（如 `Temp` 或 `Downloads`）。

```
# 過濾出 Event ID 11 (File Create)
df_file = df_sysmon[df_sysmon['System.EventID'] == 11].copy()

# 尋找目標檔案名稱的欄位
target_file_col = [col for col in df_file.columns if 'TargetFilename' in col][0]

print(f"總共發現 {len(df_file)} 筆檔案建立紀錄。")
print("最近建立的檔案：")
display(df_file[[time_col, image_col, target_file_col]].tail(10))
```

```
總共發現 102 筆檔案建立紀錄。
最近建立的檔案：
```

	System.TimeCreated.#attributes.SystemTime	EventData.Image	Event
933	2025-12-03 07:02:41.149425+00:00	C:\Windows\SysWOW64\WerFault.exe	C:\Windows\SysWOW64\confi
938	2025-12-03 07:03:14.237633+00:00	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Windows\Temp__PSScr
940	2025-12-03 07:03:15.740424+00:00	C:\Windows\syswow64\WindowsPowerShell\v1.0\pow...	C:\Windows\Temp__PSScrip
942	2025-12-03 07:03:18.251272+00:00	C:\Program Files (x86)\Palo Alto Networks\Trap...	C:\ProgramData\Cyvera\Prev
946	2025-12-03 07:03:19.682313+00:00	C:\Windows\SysWOW64\WerFault.exe	C:\ProgramData\Microsoft\Windo
947	2025-12-03 07:03:23.075574+00:00	C:\Windows\SysWOW64\WerFault.exe	C:\Windows\SysWOW64\confi
973	2025-12-03 07:05:28.774026+00:00	C:\Users\Vic\Downloads\Comae-Toolkit-v20230117...	C:\Windows\Syste
976	2025-12-03 07:05:28.918472+00:00	C:\Users\Vic\Downloads\Comae-Toolkit-v20230117...	C:\Users\Vic\Downloads\Con
993	2025-12-03 07:06:48.323012+00:00	C:\Users\Vic\Downloads\MRCv120.exe	C:\Users\Vic\Do
994	2025-12-03 07:06:48.323058+00:00	C:\Users\Vic\Downloads\MRCv120.exe	C:\Users\Vic\Do

分析：註冊表修改 (Event ID 13)

這是尋找「持久化 (Persistence)」行為的最佳地方。攻擊者常修改 `Run` 鍵值。

```
# 過濾出 Event ID 13 (Registry Value Set)
df_reg = df_sysmon[df_sysmon['System.EventID'] == 13].copy()

# 找出註冊表路徑欄位
reg_path_col = [col for col in df_reg.columns if 'TargetObject' in col][0]

print(f"發現 {len(df_reg)} 筆註冊表修改紀錄。")

# 篩選出與開機啟動 (Run keys) 相關的修改
df_run_keys = df_reg[df_reg[reg_path_col].str.contains('Run', case=False, na=False)]

if not df_run_keys.empty:
    print("發現可疑的開機啟動項修改：")
    display(df_run_keys[[time_col, image_col, reg_path_col]])
else:
    print("未在目前紀錄中發現明顯的 Run Key 修改，顯示前 5 筆一般修改：")
    display(df_reg[[time_col, image_col, reg_path_col]].head(5))
```

```
發現 194 筆註冊表修改紀錄。
發現可疑的開機啟動項修改：
```

	System.TimeCreated.#attributes.SystemTime	EventData.Image	EventData.TargetObject
8	2025-12-03 05:55:37.938672+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	HKU\S-1-5-21-3600098720-2357510703-1039409092-...
948	2025-12-03 07:03:27.392892+00:00	C:\Program Files (x86)\Microsoft\Edge\Applicat...	HKU\S-1-5-21-3600098720-2357510703-1039409092-...