Dealing with a great amount of data can be time consuming, thus using Python can be very powerful to help analysts sort information and extract the most relevant data for their investigation. The open-source tools library, MSTICPy, for example, is a Python tool dedicated to threat intelligence. It aims to help threat analysts acquire, enrich, analyze, and visualize data.

This blog provides a workflow for deeper data analysis and visualization using Python, as well as for extraction and analysis of indicators of compromise (IOCs) using MSTICPy. Data sets from the February 2022 leak of data from the ransomware-as-a-service (RaaS) coordinated operation called "Conti" is used as case study.

- Using Python to analyze the Conti network
- Using MSTICPy to extract and analyze IOCs

An interactive Jupyter notebook with related data is also available for analysts interested to do further data exploration.

This research aims to provide a view into research methodology that may help other analysts apply Python to threat intelligence. Analysts can reuse the code and continue to explore the extracted information. Additionally, it offers an out-of-the-box methodology for analyzing chat logs, extracting IOCs, and improving threat intelligence and defense process using Python.

Using Python to analyze the Conti network

On February 28, 2022, a Twitter account named @ContiLeaks (allegedly a Ukrainian researcher) began posting leaked Conti data on Twitter. The leaked data sets, which were posted in a span of several months, consisted of chat logs, source codes, and backend applications.

For this research, we focused our analysis on the chat logs, which revealed crucial information about the Conti group's operating methods, infrastructure, and organizational structure.

Compiling and translating chat logs

The leaked chat logs are written in the Russian language. To make the analysis more accessible, we adopted the methodology published here and translated the logs to English.

The chat logs revealed that the Conti group uses the messaging application Jabber to communicate among members. Since raw Jabber logs are saved using a file per day, they can be compiled in one JSON file so they can easily be manipulated with Python. Once the data is merged, they can be translated using the deep translator library. After the logs are translated and loaded into a new file, it's then possible to load the data into a dataframe for manipulation and exploration:

```
df = pd.read_json(codecs.open('translated_Log2.json', 'r', 'utf-8'))
```

	ts	from	to	body	LANG-EN
0	2021-01-29T00:06:46.929363	mango@q3mcco35auwcstmt.onion	stern@q3mcco35auwcstmt.onion	про битки не забудь, кош выше, я спать)	don't forget about cue balls, kosh is higher, I'm going to sleep)
1	2021-01-29T04:04:39.308133	mango@q3mcco35auwcstmt.onion	stern@q3mcco35auwcstmt.onion	привет	Hey
2	2021-01-29T04:04:43.474243	mango@q3mcco35auwcstmt.onion	stern@q3mcco35auwcstmt.onion	битков не хватит на все..	bits are not enough for everything ..
3	2021-01-29T04:32:02.648304	price@q3mcco35auwcstmt.onion	green@q3mcco35auwcstmt.onion	привет!!!	Hey!!!
4	2021-01-29T04:32:16.858754	price@q3mcco35auwcstmt.onion	green@q3mcco35auwcstmt.onion	опять прокладки сменились??? нет связи!	have the pads changed again? no connection!

Figure 1. Translated logs

Russian slang words not properly translated by the automated process can be translated by creating a dictionary. A dictionary off a list proposed here was used in this case to correctly translate the slang:

```
# Creating a dictionary with the translated slang words
slang = {"Hell": "AD", "YES": "DA", "wheelbarrow": "host", "cars": "hosts",
"cue balls": "bitcoin", "cr
edits":"credentials", "vmik":"WMIC", "grid":"network", "facial
expressions":"mimikatz", "firework":"fir
ewall", "whining":"SQL", "school":"SQL", "balls":"shares", "zithers":"Citrix",
"food":"FUD", "silkcode"
:"shellcode", "kosh":"cash", "toad":"jabber", "booze":"Emotet", "the trick or
trick": "Trickbot", "BC":
"BazarBackdoor", "backpack":"Ryuk", "lock":"ransomware"}

# Replacing the words in the translated column
df['LANG-EN'] = df['LANG-EN'].replace(slang, regex=True)
df['LANG-EN'].head(10)
```

Figure 2. Translating slang

Analyzing the chat activity timeline

One way to get insights from chat logs is to see its timeline and check the number of discussions per day. The Bokeh library can be used to build an interactive diagram and explore the loaded dataframe.

```
# Dynamic graph using Bokeh
import pandas_bokeh
from bokeh.models import ColumnDataSource, HoverTool
from bokeh.plotting import figure, show

df['ts'] = pd.to_datetime(df['ts'])

pandas_bokeh.output_notebook()
pd.set_option('plotting.backend', 'pandas_bokeh')

# Filter the result to manipulate only timestamp and number of discussion per
day
data2 = pd.DataFrame(df.groupby(df['ts'])['from'].count().reset_index())

# Loading the filtered dataset into ColumnDataSource
source = ColumnDataSource(data2)

# Creating the figure with the size
p = figure(x_axis_type='datetime', plot_width=900, plot_height=500)

# Adding the hover tools
p.add_tools(HoverTool(tooltips=[('Date', '@ts{%F}'), ('Nb of
discussion','@from{int}')],
                      formatters={'@ts':'datetime'}, mode='mouse'))

# Legend
p.title.text ='Activity discussion per day'
p.xaxis.axis_label = 'Date'
p.yaxis.axis_label = 'Number of discussion'

# Diagram
p.line(x='ts', y='from', line_width=2, color='#851503', source=source)

# Print the diagram
show(p)
```

Figure 3. Python code for exploring discussions

Using the data from Conti chat logs generates the following diagram, which shows the volume of Jabber discussions over time:

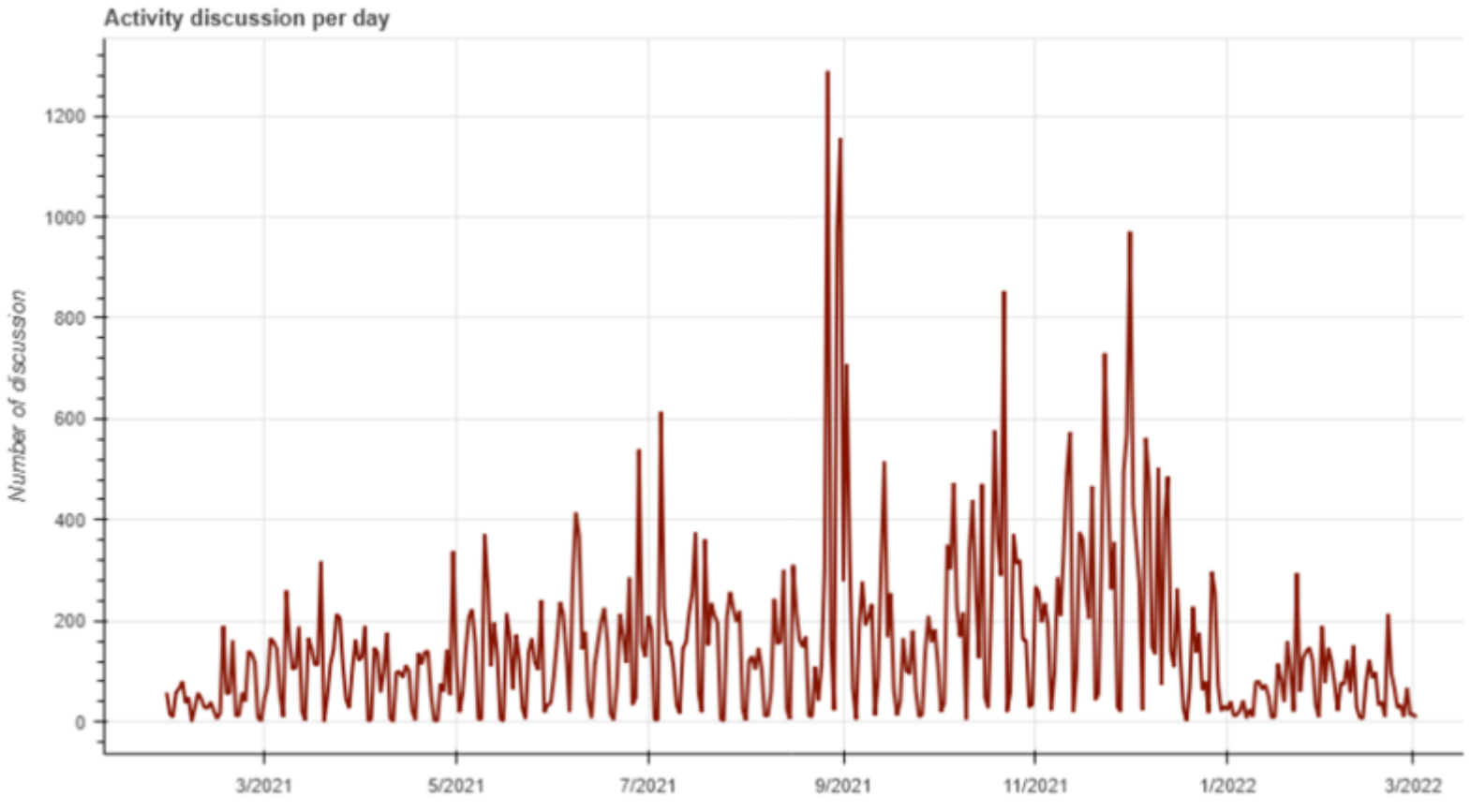

Figure 4. Volume of discussions over time

Visualizing the data as a timeline shows some peaks of activity that align to certain events. In the case of the Conti leaks, for example:

- July 7, 2021 (615 discussions): Ransomware attack by REvil against software company Kaseya
- August 27, 2021 (1,289 discussions): The playbook of a specific Conti affiliate was leaked
- August 32, 2021 (1,156 discussions): FBI CISA advisory on ransomware and labor day
- August 10, 2021 (853 discussions): Ransomware attack by Conti against Meyer Corporation

It's interesting that no peak in chat activity was observed within the Conti group after the first leak, which could indicate that the breach was ignored or not known by the group at that time.

Analyzing the level of user activity

When analyzing chat logs, identifying the number of users and analyzing the most active ones can provide insight into the size of the group and roles of users within it. Using Python, the list of users can be extracted and saved in a text file:

```
# Extracting all the users
userfrom = df['from']
userto = df['to']

# Dropping duplicate and concatenate dataframe
user = pd.concat([userfrom.drop_duplicates(), userto.drop_duplicates()],
ignore_index=True)
user = user.drop_duplicates()

# Save userlist to txt for additional hunting
user.to_csv(r'IOC\userlist.txt', header=None, index=None, sep='\t', mode='a')
```

Figure 5. Extracting list of users

Running the script above using the Conti chat logs yielded a list of 346 unique accounts. This list can then be used to create a graph and show which users sent the most messages.

```
# Static graphic
df.groupby('from').count().ts.sort_values(ascending=False).iloc[:50].plot.barh
(figsize=(15,10), title="Most active users")
```

Figure 6. Creating a graph for users with most messages

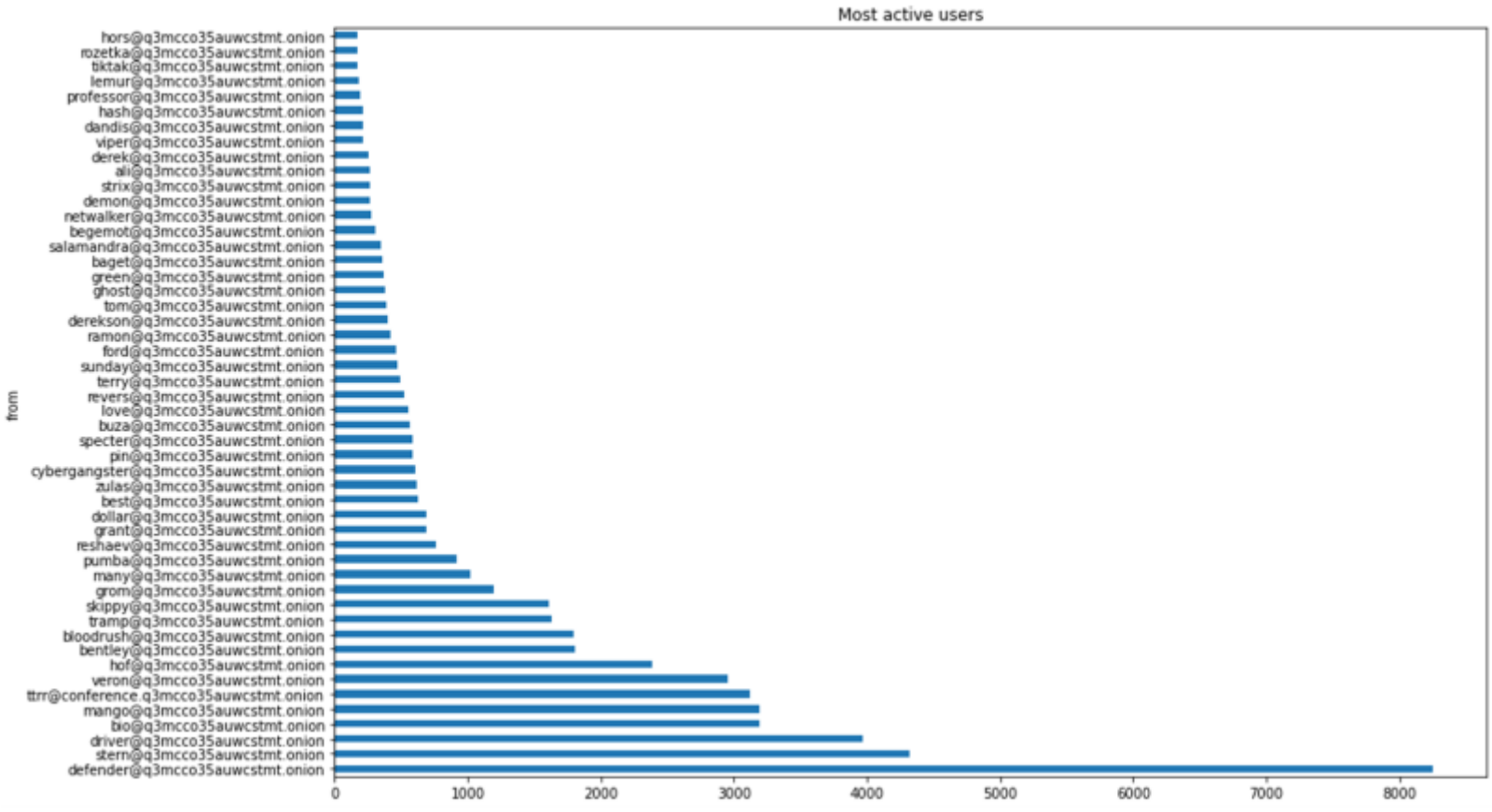

Figure 7. Most active users in the Conti chat logs

Based on the graph, the users named defender, stern, driver, bio, and mango have the largest number of discussions. Checkpoint published extensive research on the structure of the organization and correlated the user discussions with several roles and services like human resources, coders, crypters, offensive team, SysAdmins, and more.

Mapping the users' connections

Another way to analyze chat log data is to visualize the users' connection. This can be done by creating a dynamic network graph that can highlight the connections between users. The Barnes Hut algorithm and the Pyvis library can be used to visualize this data.

```
# Transforming the data, the weight corresponding to the number of message
send between 2 users.
df_weight = df.groupby(["from", "to"], as_index=False).count()
df_weight = df_weight.drop(['body','LANG-EN'], axis = 1)
df_weight.columns = df_weight.columns.str.replace('ts', 'weight')
df_weight.head(5)

# Importing the pyvis lib
from pyvis.network import Network

# Configuring the graph option
conti_net = Network(height='800px', width='100%', bgcolor='#222222',
font_color='white', notebook = True)

# set the physics layout of the network, here we used the barnes hut
conti_net.barnes_hut()
conti_data = df_weight

# Split the data
sources = conti_data['from']
targets = conti_data['to']
weights = conti_data['weight']

edge_data = zip(sources, targets, weights)

# Browsing the data to construct the network graph
for e in edge_data:
    src = e[0]
    dst = e[1]
    w = e[2]

    conti_net.add_node(src, src, title=src)
    conti_net.add_node(dst, dst, title=dst)
    conti_net.add_edge(src, dst, value=w*10)

neighbor_map = conti_net.get_adj_list()

# add user data to node hover data
for node in conti_net.nodes:
    node['title'] += ' <br> - Discussion with:<br>' +
'<br>'.join(neighbor_map[node['id']])
    node['value'] = len(neighbor_map[node['id']])

conti_net.show('conti_leak.html')
```

Figure 8. Creation a dynamic graph

Dynamic visualization shows a graphical overview of the network and allows zooming into the network to closely analyze the connections within. Bigger points represent the most active users, and it's possible to highlight a user to analyze their connections. Additionally, the hovering tool shows which other users a specific user had conversations with.

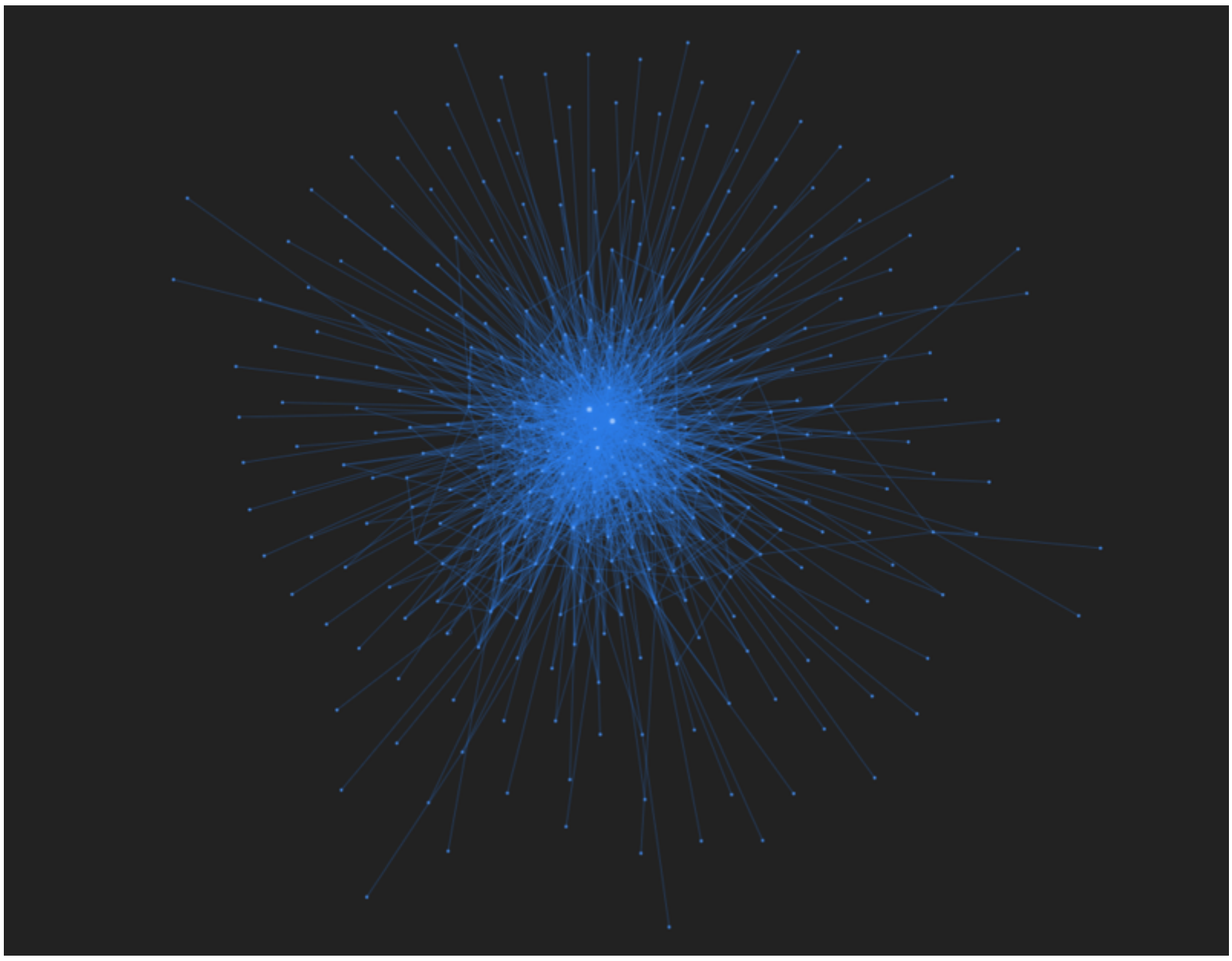

Figure 9. Conti user network overview

Figure 10. Connections to user 'Stern'

Searching for other topics of interest

Since reading data sets can be time-consuming, a simple search engine can be built to search for specific strings in the chat logs or to filter for topics of interest. For the Conti leak data, examples of these include Bitcoin, usernames, malware names, exploits, and CVEs, to name a few.

The following code snippet provides a simple search engine using the TextSearch library:

```python
# Import lib
import ipywidgets as widgets
from textsearch import TextSearch
from IPython.display import display

pd.set_option('display.max_colwidth', None)

#configure widget
keyword = widgets.Text(
    value='',
    placeholder='Enter your search',
    description='Search:',
    disabled=False
)
display(keyword)

# Configure click button
button = widgets.Button(description="search", icon='check')
display(button)

output = widgets.Output()

# Searching for the input word
@output.capture()
def userInput(b):

    # store the search result in a list
    result = []
    print("[+] Searching the chat for occurence of: " + keyword.value)

    # look for the string into the translated column
    for i in df['LANG-EN']:
        ts = TextSearch(case="ignore", returns="match")
        words = keyword.value
        ts.add(words)

        # store the result into the list
        if ts.findall(str(i)):
            result.append(i)

    # Filter and print the result
    result = list(dict.fromkeys(result))
    print('\n'.join(map(str, result)))

# get the input word
button.on_click(userInput)
display(output)
```

Figure 11. Search engine using Python

Using MSTICPy to extract and analyze IOCs

Besides processing chat logs to analyze user activity and connections, Python can also be used to extract and analyze threat intelligence. This section shows how the MSTICPy library can be used to extract IOCs and how it can be used for additional threat hunting and intelligence.

Extracting IOCs

MSTICPy is a Python library used for threat investigation and threat hunting. The library can connect to several threat intelligence providers, as well as Microsoft tools like Microsoft Sentinel. It can be used to query logs and to enrich data. It's particularly convenient for analyzing IOCs and adding more threat contextualization.

After installing MSTICPy, the first thing to do is to initialize the notebook. This allows the loading of several modules that can be used to extract and enrich the data. External resources like VirusTotal or OTX can also be added by configuring msticpyconfig.yaml and adding the API keys.

The IoCExtract module from MSTICPy offers a convenient way to extract IOCs using predefined regex. The code automatically extracts IOCs such as DNS, URLs, IP addresses, and hashes and then reports them in a new dataframe.

```python
# We clean the dataframe to remove None value
df['LANG-EN'] = df['LANG-EN'].fillna('').apply(str)

# Initiate the IOC extractor
ioc_extractor = IoCExtract()
ioc_df = ioc_extractor.extract(data = df, columns = ['LANG-EN'])

display(HTML("<h4>IoC patterns found in chat logs.</h4>"))
display(ioc_df.head(10))
```

Figure 12. Passing the dataframe to the module for extraction

IoC patterns found in chat logs.

	IoCType	Observable	SourceIndex	Input
0	dns	qaz.im	23	https://qaz.im/load/Tb6rNh/dYkYy2
1	url	https://qaz.im/load/Tb6rNh/dYkYy2	23	https://qaz.im/load/Tb6rNh/dYkYy2
2	dns	qaz.im	25	https://qaz.im/load/hzkQTQ/BTa6Ze
3	url	https://qaz.im/load/hzkQTQ/BTa6Ze	25	https://qaz.im/load/hzkQTQ/BTa6Ze
4	dns	qaz.im	29	https://qaz.im/load/Tb6rNh/dYkYy2
5	url	https://qaz.im/load/Tb6rNh/dYkYy2	29	https://qaz.im/load/Tb6rNh/dYkYy2
6	dns	qaz.im	52	https://qaz.im/load/hzkQTQ/BTa6Ze
7	url	https://qaz.im/load/hzkQTQ/BTa6Ze	52	https://qaz.im/load/hzkQTQ/BTa6Ze
8	ipv6	09:54:30	54	[09:54:30] <22> throw it right away. until March 1, whatever. and then you waste it on trifles a...
9	ipv6	09:55:17	54	[09:54:30] <22> throw it right away. until March 1, whatever. and then you waste it on trifles a...

Figure 13. Sample of extracted IOCs

A regex can be added to filter specific IOCs from those extracted by the IOC extraction module by default. For example, the regex below extracts Bitcoin addresses from the Conti chat logs:

```
# Extracting BTC addresses
# Adding the regex
extractor = IoCExtract()
extractor.add_ioc_type(ioc_type='btc', ioc_regex='^(?:[13]{1}[a-km-zA-HJ-NP-
Z1-9]{26,33}|bc1[a-z0-9]{39,59})$')

# Check that it added ok
print(extractor.ioc_types['btc'])

# Use it in our data set and create a new df
btc_df = ioc_extractor.extract(data=df, columns=['LANG-EN']).query('IoCType ==
\'btc\'')

display(HTML("<h4>BTC addresses found in chat logs.</h4>"))
display(btc_df.head(10))
```

Figure 14. Extracting Bitcoin addresses and adding regex

BTC addresses found in chat logs.

	IoCType	Observable	SourceIndex	Input
152	btc	bc1q3efl4m2jcr6gk32usxnfyrxh294sr8plmpe3ye	806	bc1q3efl4m2jcr6gk32usxnfyrxh294sr8plmpe3ye
213	btc	1MxtwUpH4cWAz4en4kqVNzAdx5gpk9etUC	1131	hello, the bitcoins are over, in total 6 new servers, two vpn subscriptions, an ipvanish subscri...
214	btc	1MxtwUpH4cWAz4en4kqVNzAdx5gpk9etUC	1136	hello, the bitcoins are over, in total 6 new servers, two vpn subscriptions, an ipvanish subscri...
296	btc	bc1qnf6drcfl786d70wlhfytyr5xg3qqgknlsh8dc3	1606	bc1qnf6drcfl786d70wlhfytyr5xg3qqgknlsh8dc3
297	btc	17mc4Qm7ka9jhQEUB5LTxP3gW3tsDYUJGQ	1608	hello, the cue ball is over, in total 8 new servers, two vpn subscriptions, and 18 renewals have...
307	btc	bc1qnf6drcfl786d70wlhfytyr5xg3qqgknlsh8dc3	1617	bc1qnf6drcfl786d70wlhfytyr5xg3qqgknlsh8dc3
308	btc	17mc4Qm7ka9jhQEUB5LTxP3gW3tsDYUJGQ	1619	hello, the cue ball is over, in total 8 new servers, two vpn subscriptions, and 18 renewals have...
329	btc	bc1qy2083z665ux68zda3tfuh5xed2493uaj8whdwv	1669	bc1qy2083z665ux68zda3tfuh5xed2493uaj8whdwv
330	btc	172KVKhMqL5CU1HN884RbArzu5DDL5hwE3	1680	172KVKhMqL5CU1HN884RbArzu5DDL5hwE3\n\n0.01523011
335	btc	bc1qc39qwc3nl2eyh2cu4ct6tyh9zqzp9ye993c0y2	1716	bc1qc39qwc3nl2eyh2cu4ct6tyh9zqzp9ye993c0y2

Figure 15. Sample of extracted Bitcoin addresses

After extracting IOCs, the dataframe can be cleaned to remove false positives as well as duplicate data. The final dataframe from the processed Conti chat logs contains the following unique IOC count, (these IOCs require additional analysis as not all of them are considered malicious):

URL	DNS	IPV4	Bitcoin	MD5	SHA-256
1,137	474	317	175	106	16

Investigating UP addresses

The threat intel lookup module TILookup in MSTICPy can be used to get more information on IOCs such as IP addresses. In the case of the Conti leak, 317 unique IP addresses were identified. Not all these IOCs are malicious but can reveal more relevant information.

The configuration file can be specified to load the TILookup module, along with other threat intelligence providers such as VirusTotal, GreyNoise, and OTX.

```
# load all configured providers
ti_lookup = TILookup(providers = ["VirusTotal", "GreyNoise", "OTX"])
ti_lookup.provider_status

# Don't forget to reload the providers once you specified the api key in the
config file.
ti_lookup.reload_providers()

# filter the dataframe by the IoCType
df_ip = ioc_df.loc[ioc_df["IoCType"] == "ipv4"]
df_ip['IoCType'].count()

# lookup the dataframe for additional threat context using the providers
ip_intel = ti_lookup.lookup_iocs(data = df_ip["Observable"])
ip_intel.head(10)
```

Figure 16. Threat intel provider within MSTICPy

Running the module generates a new dataframe with more context for every IP address provided.

	Ioc	IocType	SafeIoc	QuerySubtype	Provider	Result	Severity	Details	RawResult	Reference	Status
0	54.183.140.39	ipv4	54.183.140.39	None	GreyNoise	False	information	Not found.	<Response [404]>	https://api.greynoise.io/v3/community/54.183.140.39	404
1	5.139.220.204	ipv4	5.139.220.204	None	GreyNoise	False	information	Not found.	<Response [404]>	https://api.greynoise.io/v3/community/5.139.220.204	404
2	138.124.180.94	ipv4	138.124.180.94	None	GreyNoise	False	information	Not found.	<Response [404]>	https://api.greynoise.io/v3/community/138.124.180.94	404
3	193.203.203.101	ipv4	193.203.203.101	None	GreyNoise	False	information	Not found.	<Response [404]>	https://api.greynoise.io/v3/community/193.203.203.101	404
4	45.14.226.47	ipv4	45.14.226.47	None	GreyNoise	False	information	Not found.	<Response [404]>	https://api.greynoise.io/v3/community/45.14.226.47	404
5	64.40.247.118	ipv4	64.40.247.118	None	GreyNoise	False	information	Not found.	<Response [404]>	https://api.greynoise.io/v3/community/64.40.247.118	404
6	12.191.116.202	ipv4	12.191.116.202	None	GreyNoise	False	information	Not found.	<Response [404]>	https://api.greynoise.io/v3/community/12.191.116.202	404
7	24.53.75.60	ipv4	24.53.75.60	None	GreyNoise	False	information	Not found.	<Response [404]>	https://api.greynoise.io/v3/community/24.53.75.60	404
8	75.151.48.49	ipv4	75.151.48.49	None	GreyNoise	False	information	Not found.	<Response [404]>	https://api.greynoise.io/v3/community/75.151.48.49	404
9	184.146.91.74	ipv4	184.146.91.74	None	GreyNoise	False	information	Not found.	<Response [404]>	https://api.greynoise.io/v3/community/184.146.91.74	404

Figure 17. Sample of IP addresses enriched with additional info

The module also allows to request information for a single observable.

```
# You can also make a request for a single IP
result = ti_lookup.lookup_ioc(observable="203.76.105.227")
ti_lookup.result_to_df(result).T
```

Figure 18. Extracting information for a single observable

	GreyNoise	OTX	VirusTotal
Ioc	203.76.105.227	203.76.105.227	203.76.105.227
IocType	ipv4	ipv4	ipv4
QuerySubtype	None	None	None
Provider	GreyNoise	OTX	VirusTotal
Result	False	True	True
Severity	information	high	information
Details	Not found.	{'pulse_count': 2, 'names': ['Conti Ransomware | CISA', 'Conti Ransomware IOC'], 'tags': [['usce...	{'verbose_msg': 'IP address in dataset', 'response_code': 1, 'positives': 0, 'detected_urls': []...
RawResult	<Response [404]>	{'whois': 'http://whois.domaintools.com/203.76.105.227', 'reputation': 0, 'indicator': '203.76.1...	{'asn': 23688, 'undetected_urls': [], 'undetected_downloaded_samples': [{'date': '2021-05-25 16...
Reference	https://api.greynoise.io/v3/community/203.76.105.227	https://otx.alienvault.com/api/v1/indicators/IPv4/203.76.105.227/general	https://www.virustotal.com/vtapi/v2/ip-address/report
Status	404	0	0

Figure 19. Additional threat context for one IP address

The browser provided by MSTICPy can also be used to explore the IOCs previously enriched. The interactive Jupyter notebook includes this view of the IOCs.

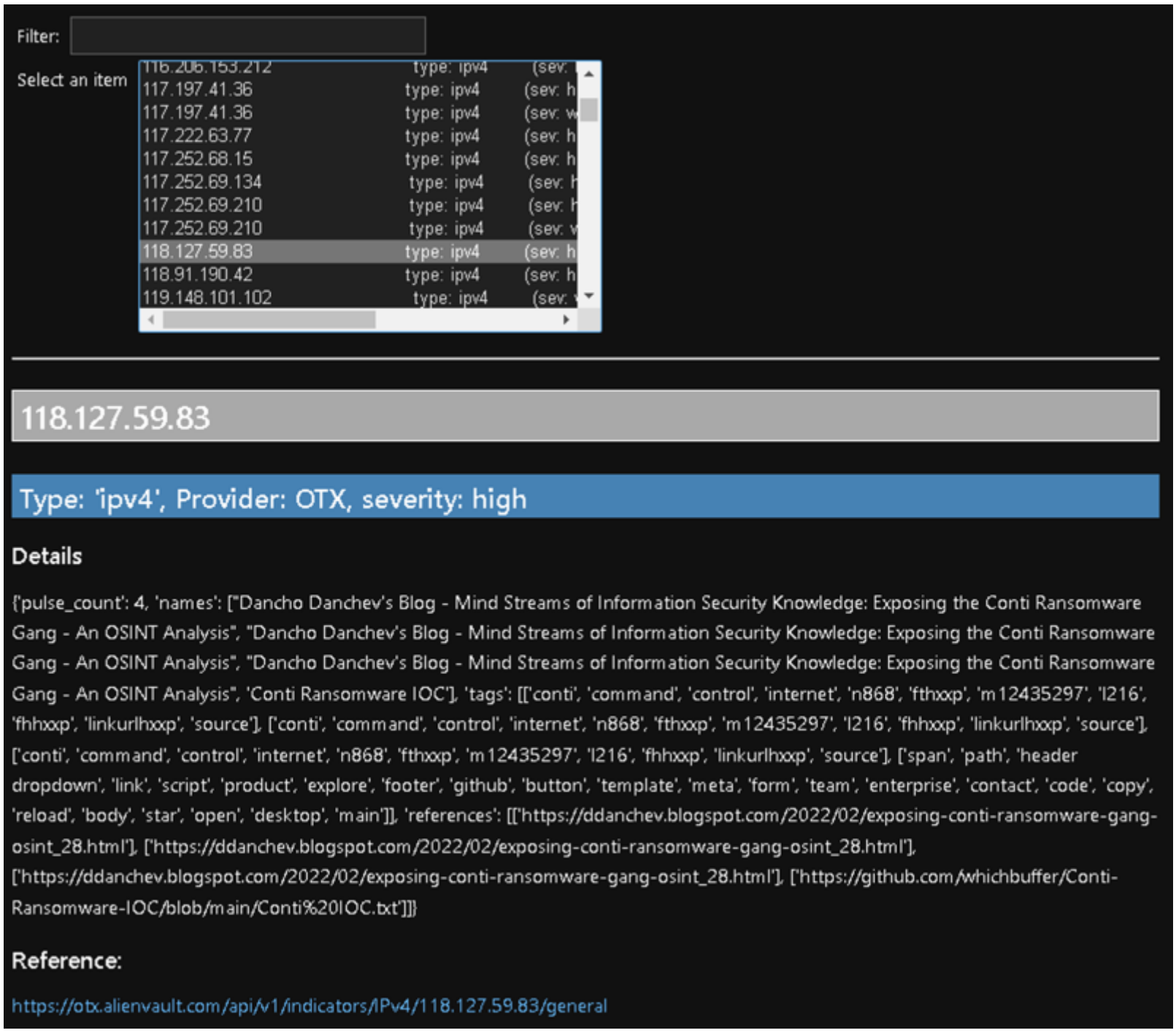

Figure 20. IOC browser provided by MSTICPy

In addition, MSTICPy has an embedded module that looks up the geolocation of IP addresses using Maxmind, which can be used to create a map of the IP addresses previously extracted.

```
# Getting the geo result
msticpy.settings.refresh_config()
iplocation = GeoLiteLookup()

# Creating the map using the folium module
iploc = []

for ip in ip_intel["Ioc"]:
    loc_result, ip_entity = iplocation.lookup_ip(ip_address = ip)
    iploc += ip_entity

folium_map = FoliumMap(zoom_start = 2)
folium_map.add_ip_cluster(ip_entities = iploc, color = 'red')
folium_map.center_map()
folium_map
```

Figure 21. Generating the IP geolocation map

Figure 22. Geolocation of IPs extracted from the Conti leaks

Investigating URLs

Extracted URLs from IOC lists can provide details about targets, tools used to exchange information, and the infrastructure used to deploy attacks. A total of 1,137 unique URLs were extracted from the Conti leak dataset, but not all of them are usable for threat intelligence. The following code snippet shows how to filter for URLs.

```
# Filtering URL
url_intel = ioc_df.loc[(ioc_df['IoCType'] == "url")]

# Sorting the value
url_intel.sort_values('Observable', ascending = True)
```

Figure 23. Filtering the IOCs for URLs

	IoCType	Observable	SourceIndex	Input
221	url	https://help4windows.com/windows_7_shell32_dll.shtml	3064	https://help4windows.com/windows_7_shell32_dll.shtml
373	url	https://oividaluxuosa.com/ke/miami.dll	5275	https://oividaluxuosa.com/ke/miami.dll , 3k copies with some neutral names so that the def does ...
564	url	https://privatlab.com/s/v/nRl7zbAAjtBeLbRqrax	6661	Check if it works\nhttps://privatlab.com/s/v/nRl7zbAAjtBeLbRqrax\n123123
600	url	https://emploimed.com/netr.dll	7147	1st link https://emploimed.com/netr.dll
602	url	https://www.ottenbourg.com/chester.dll	7149	2nd link https://www.ottenbourg.com/chester.dll
728	url	https://anonfiles.com/Ha0P8t1uc/Dolfs_rar	7742	https://anonfiles.com/Ha0P8t1uc/Dolfs_rar\npass - AF2gA52ggd
778	url	https://atlantisprojects.ca/cheryasd.dll	8160	https://atlantisprojects.ca/cheryasd.dll
783	url	https://parkisolutions.com/nerugin.dll	8197	https://parkisolutions.com/nerugin.dll
942	url	http://109.230.199.73/k.exe	9765	<off> http://109.230.199.73/k.exe\n[13.05.2021 08:33:36] <off> http://109.230.199.73/k.dll\n[13...
943	url	http://109.230.199.73/k.dll	9765	<off> http://109.230.199.73/k.exe\n[13.05.2021 08:33:36] <off> http://109.230.199.73/k.dll\n[13...
1211	url	http://ozpve456vdzplanabllomqi6fc67nlrthquvcsrfcv7z3jreurmfqd.onion	11755	http://ozpve456vdzplanabllomqi6fc67nlrthquvcsrfcv7z3jreurmfqd.onion\nadmin\n[[/JB4ccE3vzfd6
1293	url	http://i.prntscr.com/qMqzmSbHSS_QdlEUONrHZw.png	13636	http://i.prntscr.com/qMqzmSbHSS_QdlEUONrHZw.png
1380	url	http://109.230.199.73/209x64.exe	15237	http://109.230.199.73/209.dll\nhttp://109.230.199.73/209x64.exe
1381	url	http://109.230.199.73/209.dll	15237	http://109.230.199.73/209.dll\nhttp://109.230.199.73/209x64.exe
1675	url	https://bradiolum.top/aprel.dll	19733	now again on the command dll files error\nhttps://bradiolum.top/aprel.dll\n\nhttps://auk64p35qeb...
1811	url	file://157.230.60.143/download.jpg	21267	[07/27/2021 19:01:56] <rozteka> https://www.ired.team/offensive-security/initial-access/netntlmv...
2452	url	http://31.14.*0.220/230*17*.dll,StartW	33028	http://31.14.*0.220/230*17*.dll,StartW
2495	url	https://temp.sh/EIXCc/1.rar	33474	Готово. \n[20:40:06] <bentley> pass: kJHDF273yubfjsbdf973uiwhgjsnkgb3oiygbhjsbdgkjhb \n[20:40:13...
2502	url	http://bergmeitli.ch/2.dll	33631	altmann-dias.com/1.dll\nhttp://bergmeitli.ch/2.dll
2509	url	http://195.149.87.59/2_https_x64.dll	33801	http://195.149.87.59/1_http_x64.dll\nhttps://195.149.87.59/2_https_x64.dll\n\nStartW
2510	url	http://195.149.87.59/1_http_x64.dll	33801	http://195.149.87.59/1_http_x64.dll\nhttps://195.149.87.59/2_https_x64.dll\n\nStartW
2601	url	https://temp.sh/jDpqP/1.rar	36060	https://temp.sh/jDpqP/1.rar

Figure 24. Sample of URLs extracted

A filter can be created to get details on executables, DLLs, ZIP files, and other files related to the extracted URLs. This can provide interesting insights and can be extracted for further research.

```
# Filtering on some files (dll, jpg, exe, zip, rar, png)
url_intel[url_intel['Observable'].str.contains(".exe|.dll|.jpg|.zip|.7z|.rar|.
png")]
```

Figure 25. Filtering URLs for specific file formats

	IoCType	Observable	SourceIndex	Input
221	url	https://help4windows.com/windows_7_shell32_dll.shtml	3064	https://help4windows.com/windows_7_shell32_dll.shtml
373	url	https://oividaluxuosa.com/ke/miami.dll	5275	https://oividaluxuosa.com/ke/miami.dll , 3k copies with some neutral names so that the def does ...
564	url	https://privatlab.com/s/v/nRI7zbAAjtBeLbRqrax	6661	Check if it works\nhttps://privatlab.com/s/v/nRI7zbAAjtBeLbRqrax\n123123
600	url	https://emploimed.com/netr.dll	7147	1st link https://emploimed.com/netr.dll
602	url	https://www.ottenbourg.com/chester.dll	7149	2nd link https://www.ottenbourg.com/chester.dll
728	url	https://anonfiles.com/Hai0P8t1uc/Dolfs_rar	7742	https://anonfiles.com/Hai0P8t1uc/Dolfs_rar\npass - AF2gAS2ggd
778	url	https://atlantisprojects.ca/cheryasd.dll	8169	https://atlantisprojects.ca/cheryasd.dll
783	url	https://parkisolutions.com/nerugin.dll	8197	https://parkisolutions.com/nerugin.dll
942	url	http://109.230.199.73/k.exe	9765	<off> http://109.230.199.73/k.exe\n[13.05.2021 08:33:36] <off> http://109.230.199.73/k.dll\n[13....
943	url	http://109.230.199.73/k.dll	9765	<off> http://109.230.199.73/k.exe\n[13.05.2021 08:33:36] <off> http://109.230.199.73/k.dll\n[13....
1211	url	http://ozpve456vdzplanabllomqi6lfx67nlrthquvcsrfxv7z3jreurmfqd.onion	11755	http://ozpve456vdzplanabllomqi6lfx67nlrthquvcsrfxv7z3jreurmfqd.onion\nadmin\n[](.)B4xcE3v=fd6
1293	url	http://i.prntscr.com/qMqzmSbHSS_QdIEUONrHZw.png	13636	http://i.prntscr.com/qMqzmSbHSS_QdIEUONrHZw.png
1380	url	http://109.230.199.73/209x64.exe	15237	http://109.230.199.73/209.dll\nhttp://109.230.199.73/209x64.exe
1381	url	http://109.230.199.73/209.dll	15237	http://109.230.199.73/209.dll\nhttp://109.230.199.73/209x64.exe
1675	url	https://bradiolum.top/aprel.dll	19733	now again on the command dll flies error\nhttps://bradiolum.top/aprel.dll\n\nhttps://auk64p35qeb...
1811	url	file://157.230.60.143/download.jpg	21267	[07/27/2021 19:01:56] <rozteka> https://www.ired.team/offensive-security/initial-access/netntlmv...
2452	url	http://31.14.*0.220/230*17*.dll,StartW	33028	http://31.14.*0.220/230*17*.dll,StartW
2495	url	https://temp.sh/fJXCc/1.rar	33474	Готово. \n[20:42:06] <bentley> pass: kJHDF273yubfjsbdf973uiwhgjsnkgb3oiygbhjsbdgkjhb \n[20:42:13...
2502	url	http://bergmeitli.ch/2.dll	33631	altmann-dias.com/1.dll\nhttp://bergmeitli.ch/2.dll
2509	url	http://195.149.87.59/2_https_x64.dll	33801	http://195.149.87.59/1_http_x64.dll\nhttp://195.149.87.59/2_https_x64.dll\n\nStartW
2510	url	http://195.149.87.59/1_http_x64.dll	33801	http://195.149.87.59/1_http_x64.dll\nhttp://195.149.87.59/2_https_x64.dll\n\nStartW
2601	url	https://temp.sh/jDpqP/1.rar	36060	https://temp.sh/jDpqP/1.rar
2767	url	https://temp.sh/copeR/tmp.zip	39915	ADo, can I have a new crypt, please, the last build is already burning with something https://te...
2843	url	https://temp.sh/bctPM/f3cfb349.7z	41688	https://temp.sh/bctPM/f3cfb349.7z

Figure 26. Sample of URLs delivering extracted file

	IoCType	Observable	SourceIndex	Input
287	url	https://43oxsnqlub6wydymkwpn3agaaj7u2qecc4vwybgrwug46c6yldhuheid.onion/crpanel/	4306	https://43oxsnqlub6wydymkwpn3agaaj7u2qecc4vwybgrwug46c6yldhuheid.onion/crpanel/
740	url	https://dnog7cgicmkrvugrfxeco34gjkjbr54id5ikxj4r42ajftuy2hjrw6qd.onion	7860	Ready to access the admin panel (storage)\n[19:09:18] <bentley> https://dnog7cgicmkrvugrfxeco34g...
936	url	http://epyclq65gskclmpu.onion:1337	9751	http://epyclq65gskclmpu.onion:1337 - our file cleaner. will be on the SIA bransomwarechain
1211	url	http://ozpve456vdzplanabllomqi6lfx67nlrthquvcsrfxv7z3jreurmfqd.onion	11755	http://ozpve456vdzplanabllomqi6lfx67nlrthquvcsrfxv7z3jreurmfqd.onion\nadmin\n[](.)B4xcE3v=fd6
1218	url	http://crdclub4wraumez4.onion/	11827	a cow was sold http://korovka32xc3t5cg.onion support@korovka.name and a card like http://crdclub...
1219	url	http://korovka32xc3t5cg.onion	11827	a cow was sold http://korovka32xc3t5cg.onion support@korovka.name and a card like http://crdclub...
1255	url	https://xzu6o2ni3hplvpmx.onion	12630	for HORSE\nrobotbander@jabb.im\n4815162342@jabb.im\nsheppard@jabber.ru\nsectorzero@jabb.im\n\n\n...
1321	url	http://i5nxdyozq7uyotqtmcj4hxq7modmxklejqysurqzf5ixhzw444jymyyd.onion/adminjx1p8zu25dr4ae7o.php?...	14250	http://i5nxdyozq7uyotqtmcj4hxq7modmxklejqysurqzf5ixhzw444jymyyd.onion/adminjx1p8zu25dr4ae7o.php?...
1674	url	https://auk64p35qebertdsh376avhnswxdpft3kpmvsm5sioxof6bsbgryxqd.onion/logpost/more_ex/D1F299F1B...	19733	now again on the command dll flies error\nhttps://bradiolum.top/aprel.dll\n\nhttps://auk64p35qeb...
1880	url	https://xflemdsxjrjilw34dsxpvrxp5whnaut7hc5xejwuqs6eqrkt77bxkwid.onion	21800	https://xflemdsxjrjilw34dsxpvrxp5whnaut7hc5xejwuqs6eqrkt77bxkwid.onion\nganesh: fp0fqpVxJrYsorC5...
1954	url	http://continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/	23069	Here is the Tor for now http://continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/
2136	url	http://xssforumv3isucukbxhdhwz67hoa5e2voakcfkuieq4ch257vsburuid.onion/threads/50513/	28827	http://xssforumv3isucukbxhdhwz67hoa5e2voakcfkuieq4ch257vsburuid.onion/threads/50513/
2137	url	http://xssforumv3isucukbxhdhwz67hoa5e2voakcfkuieq4ch257vsburuid.onion/threads/55956/	28828	http://xssforumv3isucukbxhdhwz67hoa5e2voakcfkuieq4ch257vsburuid.onion/threads/55956/
2157	url	https://mb5fbvx72fbod2hkirfecc5nh7wq6ke7xocn7j2u7raiwbytvevpbad.onion/begemot/dero.git	29668	[core]\nrepositoryformatversion=0\nfilemode=true\nbar = false\nlogallrefupdates=true\n[branch "m...
2158	url	https://mb5fbvx72fbod2hkirfecc5nh7wq6ke7xocn7j2u7raiwbytvevpbad.onion/begemot/dero.git/	29669	(base) begemot@big-comp:~/erl/dero/.git$ git push\nfatal: <https://mb5fbvx72fbod2hkirfecc5nh7wq...
2353	url	http://xssforumv3isucukbxhdhwz67hoa5e2voakcfkuieq4ch257vsburuid.onion/threads/56486/	32626	http://xssforumv3isucukbxhdhwz67hoa5e2voakcfkuieq4ch257vsburuid.onion/threads/56486/
2354	url	http://xssforumv3isucukbxhdhwz67hoa5e2voakcfkuieq4ch257vsburuid.onion/threads/56793/	32627	http://xssforumv3isucukbxhdhwz67hoa5e2voakcfkuieq4ch257vsburuid.onion/threads/56793/
2621	url	https://ojdglzhrquash4igbox6e6wlthe3si4biabcpfopiw33uohvaufjgipad.onion	36952	https://ojdglzhrquash4igbox6e6wlthe3si4biabcpfopiw33uohvaufjgipad.onion
2794	url	https://6ypj2jljwgdxmwy4uxfaxbkjgm2txlxcb5akxn43cyxz3cjo2gqd65yid.onion	40183	jups 111111\nhttps://6ypj2jljwgdxmwy4uxfaxbkjgm2txlxcb5akxn43cyxz3cjo2gqd65yid.onion
2863	url	http://4nmxrhdtbznfr7f3q6bhd4qoxfcxodxo3h2txugojzizc4uhppdkzad.onion/private/168xvvj5/M5kuzP_za...	42663	http://4nmxrhdtbznfr7f3q6bhd4qoxfcxodxo3h2txugojzizc4uhppdkzad.onion/private/168xvvj5/M5kuzP_za...
2866	url	http://continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/6z3vSKVI_DEWEtech	42840	http://continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/6z3vSKVI_DEWEtech
2867	url	http://continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/gWu2p5H1_TTC	42850	http://continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/gWu2p5H1_TTC
2878	url	http://continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/Xa3Uo9Gk_KISTERS	43520	http://continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/Xa3Uo9Gk_KISTERS
2908	url	http://continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/QlpblFS3_Harness_IP	44194	http://continewsnv5otx5kaoje7krkto2qbu3gtqef22mnr7eaxw3y6ncz3ad.onion/QlpblFS3_Harness_IP

Figure 27. Sample of extracted .onion URLs

Pivoting extracted IOCs using VirusTotal

The use of the pivot function within the MSTICPy library allows enrichment of data and discovery of additional infrastructure and IOC. This is particularly useful for threat intelligence and threat actor tracking. The next sections demonstrate the use of the VirusTotal module VTlookupV3 in MSTICPy to obtain intelligence about an IP address extracted from the Conti leak dataset that was used to deliver additional malware.

The following code initiates the VTlookupV3 in MSTICPy:

```
# Loading the VT API key
from msticpy.common.provider_settings import get_provider_settings
from msticpy.sectools.vtlookupv3 import VTLookupV3, VTEntityType
import nest_asyncio
vt_key = get_provider_settings("TIProviders")["VirusTotal"].args["AuthKey"]

# Instantiate vt_lookup object
vt_lookup = VTLookupV3(vt_key)
nest_asyncio.apply()
```

Figure 28. Configuring the VirusTotal module in MSTICPy

The VirusTotal module can be used to get data related to a particular IOC. The code below searches for files downloaded from a particular IP address from the Conti leak dataset:

```
# get intelligence for one ip address
IP = "109.230.199.73"
ip_relation = vt_lookup.lookup_ioc_relationships(observable = IP, vt_type =
'ip_address', relationship = 'downloaded_files')
ip_relation
```

Figure 29. Getting files downloaded from one IP address

The results show that the IP address 109[.]230[.]199[.]73 delivers several strains of malware.

source	target	target_type	source_type	relationship_type
109.230.199.73	cf0a85f491146002a26b01c8aff864a39a18a70c7b5c579e96deda212bfeec58	file	ip_address	downloaded_files
	889e89b7c88b217f02e2b8ee54f7ee142aeb3fd60a1bd002482664a1dc8ba4ae	file	ip_address	downloaded_files
	a738cf48df8b168e783a8728baac0d208298361a696ef219de01faeba030316f	file	ip_address	downloaded_files
	21145b7f20221b447d2b58ca5aaa17f6eedba1f8aa2ed91ca5ffd696cc560868	file	ip_address	downloaded_files
	d2c9f693a2080c6382a0a29d74a1b5cb13a1deeb5dbe7ff1427a669ddf66f59e	file	ip_address	downloaded_files
	37ce6b6f7a4026a69784ee202283bb4d9f13651b84cb1abaec0ca4f359514a0b	file	ip_address	downloaded_files
	a4dc4dd1ddb449490d236dd1cbf087fbdf7f923616a9948bf32b28eff03e57c9	file	ip_address	downloaded_files
	61ca39fe6ad7c054484810ba7ca1f292efab2399a5607f42006d088302f07efc	file	ip_address	downloaded_files
	fe52c23ae690d0dcf2bda89c7ed75f798d2d94beaabed014de5b76159f336f5e	file	ip_address	downloaded_files
	83e285b9347fd74af8cb9c1962f584191325a98b50b2a6df6738aacd0c8054db	file	ip_address	downloaded_files
	1bad6b8cf97131fceab8543e81f7757195fbb1d36b376ee994ad1cf17699c464	file	ip_address	downloaded_files

Figure 30. Hashes related to IP 109[.]230[.]199[.]73

The VirusTotal module can then be used to pivot and extract more information about these hashes. The table below shows information about the first hash on the list:

	Attributes
authentihash	0d10a35c1bed8d5a4516a2e704d43f10d47ffd2aabd9ce9e04fb3446f62168bf
creation_date	1624910154
crowdsourced_ids_results	[{[TRUNCATED]'alert_context': [{'dest_ip': '8.8.8.8', 'dest_port': 53}, {'dest_ip': '193.204.114.232', 'dest_port': 123}], 'rule_url': 'https://www.snort.org/downloads/#rule-downloads', 'rule_source': 'Snort registered user ruleset', 'rule_id': '1:527'}, {'rule_category': 'not-suspicious', 'alert_severity': 'low', 'rule_msg': 'TAG_LOG_PKT', 'rule_raw': 'alert (gid:2; sid:1; rev:1; msg:"TAG_LOG_PKT"; metadata:rule-type preproc; classtype:not-suspicious;)', 'alert_context': [{'dest_ip': '107.181.161.197', 'dest_port': 443}], 'rule_url': 'https://www.snort.org/downloads/#rule-downloads', 'rule_source': 'Snort registered user ruleset', 'rule_id': '2:1'}]
crowdsourced_ids_stats	{'info': 0, 'high': 0, 'medium': 2, 'low': 1}
downloadable	TRUE
exiftool	{'MIMEType': 'application/octet-stream', 'Subsystem': 'Windows GUI', 'MachineType': 'AMD AMD64', 'TimeStamp': '2021:06:28 19:55:54+00:00', 'FileType': 'Win64 DLL', 'PEType': 'PE32+', 'CodeSize': '115712', 'LinkerVersion': '14.16', 'ImageFileCharacteristics': 'Executable, Large address aware, DLL', 'FileTypeExtension': 'dll', 'InitializedDataSize': '69632', 'SubsystemVersion': '6.0', 'ImageVersion': '0.0', 'OSVersion': '6.0', 'EntryPoint': '0x139c4', 'UninitializedDataSize': '0'}
first_submission_date	1624917754
last_analysis_date	16365918529
last_analysis_results	{ [TRUNCATED] '20211110'}, 'Tencent': {'category': 'undetected', 'engine_name': 'Tencent', 'engine_version': '1.0.0.1', 'result': None, 'method': 'blacklist', 'engine_update': '20211111'}, 'Ad-Aware': {'category': 'malicious', Edition': {'category': 'malicious', 'engine_name': 'McAfee-GW-Edition', 'engine_version': 'v2019.1.2+3728', 'result': 'RDN/CobaltStrike', 'method': 'blacklist', 'engine_update': '20211110'}, 'Trapmine': {'category': 'type-unsupported', 'engine_name': 'Trapmine', 'engine_version': '3.5.0.1023', 'result': None, 'method': 'blacklist', 'engine_update': '20200727'}, 'CMC': {'category': 'undetected', 'engine_name': 'CMC', 'engine_version': '2.10.2019.1', 'result': None, 'method': 'blacklist', 'engine_update': '20211026'}, 'Sophos': {'category': 'malicious', 'engine_name': 'Sophos', 'engine_version': '1.4.1.0', 'result':
last_analysis_stats	{'harmless': 0, 'type-unsupported': 6, 'suspicious': 0, 'confirmed-timeout': 1, 'timeout': 0, 'failure': 0, 'malicious': 47, 'undetected': 19}

last_modification_date	1646895757
last_submission_date	1624917754
magic	PE32+ executable for MS Windows (DLL) (GUI) Mono/.Net assembly
md5	55646b7df1d306b0414d4c8b3043c283
meaningful_name	197.dll
names	[197.dll, iduD2A1.tmp]
pe_info	[TRUNCATED] {'exports': ['StartW', '7c908697e85da103e304d57e0193d4cf'}, {'name': '.rsrc', 'chi2': 51663.55, 'virtual_address': 196608, 'entropy': 5.81, 'raw_size': 1536, 'flags': 'r', 'virtual_size': 1128, 'md5':, 'GetStringTypeW', 'RtlUnwindEx', 'GetOEMCP', 'TerminateProcess', 'GetModuleHandleExW', 'IsValidCodePage', 'WriteFile', 'CreateFileW', 'FindClose', 'TlsGetValue', 'GetFileType', 'TlsSetValue', 'HeapAlloc', 'GetCurrentThreadId', 'SetLastError', 'LeaveCriticalSection']}], 'entry_point': 80324}
popular_threat_classification	{'suggested_threat_label': 'trojan.bulz/shelma', 'popular_threat_category': [{'count': 22, 'value': 'trojan'}, {'count': 6, 'value': 'downloader'}, {'count': 2, 'value': 'dropper'}], 'popular_threat_name': [{'count': 6, 'value': 'bulz'}, {'count': 6, 'value': 'shelma'}, {'count': 3, 'value': 'cobaltstrike'}]}
reputation	0
sandbox_verdicts	{'Zenbox': {'category': 'malicious', 'sandbox_name': 'Zenbox', 'malware_classification': ['MALWARE', 'TROJAN', 'EVADER']}, 'C2AE': {'category': 'undetected', 'sandbox_name': 'C2AE', 'malware_classification': ['UNKNOWN_VERDICT']}, 'Yomi Hunter': {'category': 'malicious', 'sandbox_name': 'Yomi Hunter', 'malware_classification': ['MALWARE']}, 'Lastline': {'category': 'malicious', 'sandbox_name': 'Lastline', 'malware_classification': ['MALWARE']}}
sha1	ddf0214fbf92240bc60480a37c9c803e3ad06321
sha256	cf0a85f491146002a26b01c8aff864a39a18a70c7b5c579e96deda212bfeec58
sigma_analysis_stats	{'high': 0, 'medium': 1, 'critical': 1, 'low': 0}
sigma_analysis_summary	{'Sigma Integrated Rule Set (GitHub)': {'high': 0, 'medium': 0, 'critical': 1, 'low': 0}, 'SOC Prime Threat Detection Marketplace': {'high': 0, 'medium': 1, 'critical': 0, 'low': 0}}
size	181248
ssdeep	3072:fck3rwbtOsN4X1JmKSol6LZVZgBPruYgr3Ig/XZO9:fck3rwblqPgokNgBPr9gA
tags	[assembly, invalid-rich-pe-linker-version, detect-debug-environment, long-sleeps, 64bits, pedll]
times_submitted	1
tlsh	T110049E14B2A914FBEE6A82B984935611B07174624338DFEF03A4C375DE0E7E15A3EF25
total_votes	{'harmless': 0, 'malicious': 0}
trid	[{'file_type': 'Win64 Executable (generic)', 'probability': 48.7}, {'file_type': 'Win16 NE executable (generic)', 'probability': 23.3}, {'file_type': 'OS/2 Executable (generic)', 'probability': 9.3}, {'file_type': 'Generic Win/DOS Executable', 'probability': 9.2}, {'file_type': 'DOS Executable Generic', 'probability': 9.2}]
type_description	Win32 DLL
type_extension	dll
type_tag	pedll
unique_sources	1
Vhash	115076651d155d15555az43=z55

The results indicate that the hash is a Cobalt Strike loader, which means that Conti affiliates also use the penetration testing tool as part of their infrastructure during their operation.

In addition, the VirusTotal module can also provide details such as detection rate, type, description, and other information related to the hashes. The code snippet below generates the list of domains to which the hashes connect to.

```
multiple_result = vt_lookup.lookup_iocs_relationships(ip_relation,
relationship = 'contacted_domains')
```

Figure 31. Getting contacted domains

source	target	target_type	source_type	relationship_type
cf0a85f491146002a26b01c8aff864a39a18a70c7b5c579e96deda212bfeec58	125.21.88.13.in-addr.arpa	domain	file	contacted_domains
	130.155.190.20.in-addr.arpa	domain	file	contacted_domains
	137.90.64.13.in-addr.arpa	domain	file	contacted_domains
	150.32.88.40.in-addr.arpa	domain	file	contacted_domains
	197.161.181.107.in-addr.arpa	domain	file	contacted_domains
	83.188.255.52.in-addr.arpa	domain	file	contacted_domains
	zizodream.com	domain	file	contacted_domains
889e89b7c88b217f02e2b8ee54f7ee142aeb3fd60a1bd002482664a1dc8ba4ae	krinsop.com	domain	file	contacted_domains
21145b7f20221b447d2b58ca5aaa17f6eedba1f8aa2ed91ca5ffd696cc560868	1.155.190.20.in-addr.arpa	domain	file	contacted_domains
	106.89.54.20.in-addr.arpa	domain	file	contacted_domains
	152.68.35.23.in-addr.arpa	domain	file	contacted_domains
	226.101.242.52.in-addr.arpa	domain	file	contacted_domains
	234.151.42.104.in-addr.arpa	domain	file	contacted_domains
	41.69.35.23.in-addr.arpa	domain	file	contacted_domains
	48.193.43.104.in-addr.arpa	domain	file	contacted_domains
	80.69.35.23.in-addr.arpa	domain	file	contacted_domains
	83.188.255.52.in-addr.arpa	domain	file	contacted_domains
	prda.aadg.msidentity.com	domain	file	contacted_domains
61ca39fe6ad7c054484810ba7ca1f292efab2399a5607f42006d088302f07efc	fanklez.com	domain	file	contacted_domains

Figure 32. Additional domains retrieved from previously extracted hashes

Doing this kind of analysis on the Conti leak data or similar data sets can lead to the discovery of possibly related domains that were not in the initial data sets.

Conclusion

This blog outlines how Python can be used to find valuable threat intelligence from data sets such as chat logs. It also presents details on how processing data using the MSTICPy library can be useful for enriching and hunting within environments, as well as collecting additional threat context. The interactive notebook provides additional code snippets that can also be used to continue log exploration.

The types of information extracted in this blog provides insights into the various elements of the criminal ecosystem that were coordinating their activities. Threat intelligence from research like this informs products and services like Microsoft 365 Defender, translating knowledge into real-world protection for customers. More importantly, the methodology described in this blog can be adapted to specific threat intelligence services, and the broader community is invited to use it for further analysis, enrichment of data, and intelligence sharing for the benefit of all.

Thomas Roccia Microsoft 365 Defender Research Team

References

- https://krebsonsecurity.com/2022/03/conti-ransomware-group-diaries-part-i-evasion/
- https://research.checkpoint.com/2022/leaks-of-conti-ransomware-group-paint-picture-of-a-surprisingly-normal-tech-start-up-sort-of/
- https://therecord.media/conti-leaks-the-panama-papers-of-ransomware/
- https://www.breachquest.com/conti-leaks-insight-into-a-ransomware-unicorn/
- https://www.forescout.com/resources/analysis-of-conti-leaks/
- https://github.com/Res260/conti_202202_leak_procedures
- https://readme.security/the-conti-leaks-first-rumble-of-the-ukraine-earthquake-thats-rattling-the-cybercrime-underground-7abb23b0fb04
- https://medium.com/@arnozobec/analyzing-conti-leaks-without-speaking-russian-only-methodology-f5aecc594d1b
- https://github.com/soufianetahiri/ContiLeaks/blob/main/cobaltsrike_lolbins
- https://twitter.com/TheDFIRReport/status/1498656118746365952
- https://www.clearskysec.com/wp-content/uploads/2021/02/Conti-Ransomware.pdf
- https://blog.bushidotoken.net/2022/04/lessons-from-conti-leaks.html
- https://www.trellix.com/en-au/about/newsroom/stories/threat-labs/conti-leaks-examining-the-panama-papers-of-ransomware.html
- https://msticpy.readthedocs.io/en/latest/getting_started/Introduction.html