

Lightweight Python-Based Malware Analysis Pipeline

Author: Marco Gfeller, sans-whitepaper@chroot.ch
Advisor: Dr. Tim Proffitt

Accepted: *January 14, 2023*

Abstract

Sharing threat information through Indicators of Compromise (IOC), Tactics, Techniques, and Procedures (TTP) used by threat actors improves cyber security by actively protecting and monitoring an organization's network. The well-known Malware Information Sharing Platform (MISP, 2022), initially developed by the Belgian CERT and NATO CERT, is the ultimate platform to share such information throughout the cyber security community. To generate new IOCs, companies need to have an automated way to analyze suspicious binaries, generate high-value indicators and share them. This research paper provides a self-developed Malware Analysis Pipeline tool to analyze potentially malicious email attachments automatically. The attachments run through a plugin-based Python Malware Analysis Pipeline and are sent to various sandboxes. All the information gained by the sandboxes and their reports is then shared in the MISP. Furthermore, all binaries are shared on the free malware-sharing platform MalwareBazaar (MalwareBazaar, 2022), run by abuse.ch.

1. Introduction

Malware is a massive threat to all industries nowadays. The popularity of compromising a company's network, exfiltrating its sensitive data, encrypting the devices, and extorting the company afterward has become a criminal business model. Helping to protect organizations from initial infections or Command and Control (C2) communication, analyzing emails with malicious attachments (malspam) is crucial. The gained information can be used to protect the network and can be shared within the cyber security community. Many cyber security vendors sell Email Gateways with Advanced Threat Protection to protect a specific company. However, such a product's tremendous price is only affordable by some organizations. Furthermore, these vendors will not share the information publicly, as this would disrupt their business model.

As sharing is caring in the field of cyber security, an Open-Source Lightweight Python-Based Malware Analysis Pipeline (LPMAP) can help the security community to fulfill the goal of an automated step-by-step plugin-based tool. Such a Malware Analysis Pipeline contains the steps of analyzing malicious email attachments from spam traps, finding C2 configurations, and sharing the information with the community in the Malware Information Sharing Platform (MISP). In addition, it strengthens the relationship in the community, builds trust, and makes it easier to find malicious traffic in an automated way.

1.1. Malware Spam Trap

The main goal of malware spam traps is to lure spammers and to receive as many malspam emails as possible. There are multiple options to deploy or use email addresses as spam traps. One possibility is to use newly generated email addresses and distribute them as far as possible, including websites, forums, or mailing lists. Another way is to reuse employees' email addresses who have left the company. A third possibility is to use the whole domain of companies that have rebranded the company name or filed for bankruptcy and do not use the domain anymore. All these options allow researchers to collect a vast amount of malware samples.

1.2. Malware Sandbox

A malware sandbox (Sandbox, 2022) is a specially prepared operating system that can execute malware in an isolated environment. The advantage of such an environment is that the malware cannot infect any other computers within the organization's network.

Sandboxes can be either virtual machines or physical computers. In both cases, the sandbox tries to mimic an end-user environment with additional tools to monitor the execution and the behavior of the malware. The monitoring process output will be analysis reports in different output formats, malware families and signatures, memory dumps, and network traffic packet capture (PCAP).

1.3. YARA

YARA is a tool initially developed by Victor Alvarez of Virustotal (Malwarebytes, 2017) and is mainly used by malware researchers to identify and classify malware samples. It is possible to search for strings or even binary patterns and use conditions to match the search patterns.

A simple YARA rule has the following format.

```
rule example
{
  meta:
    description = "Example"
    author = "John Doe"
  strings:
    $a = {61 62}
    $c = "c"
  condition:
    $a and $c
}
```

Figure 1: YARA Rule Example

1.4. Malware Information Sharing Platform

The Malware Information Sharing Platform (MISP, 2022) was initially developed by the Belgian Computer Emergency Response Team (CERT) and the North Atlantic Treaty Organization (NATO) CERT and is a de facto standard for sharing Threat Intelligence (TI) in an automated and secure way. MISP is an event-driven framework that allows users to use tags to add metadata and attributes to store Indicators of Compromise (IOCs) with additional comment possibilities.

2. Research Method

Many companies have sandboxes in place but cannot share the IOCs in an easy and automated way. Without this sharing mechanism, much of the potentially interesting information is lost and cannot be used to identify, protect and monitor organizations' networks. An automated Malware Analysis Pipeline is the solution to simplify the process and the time needed to manually send samples to a sandbox and share the results of the reports.

After researching, only one Open-Source Malware Analysis Pipeline is available on GitHub called Karton (Karton, 2022), developed by the Polish CERT. The Malware Analysis Pipeline from CERT.pl has its own ecosystem with a sandbox called Drakvuf (Drakvuf, 2022), a searchable Malware Database (MWDB, 2022), and even an indexer framework for future YARA searches (mquery, 2022). The project has grown in the last couple of years and has many components which need to be kept up to date to have no vulnerability issues, especially the web frameworks. The operation of Karton has implications for the workforce and organizations' budget, which needs to be provided by the companies running Karton. The LPMAP aims to reduce operational tasks and minimize the attack surface.

To fulfill these research requirements, the LPMAP is developed in Python only and provides an easy-to-install and secure tool without the need for additional frameworks.

2.1. Sandboxes

The whole research project depends on having access to malware sandboxes to receive new IOCs, which can then be shared with the cyber security community. For this reason, two different sandbox vendors were chosen. One company, called JoeSecurity with the Joe Sandbox (Joe Sandbox, 2022), is located in Switzerland, and the second company, called Recorded Future with the Triage Sandbox (Triage, 2022), is located in the United States of America. Both companies provide researcher accounts, and most importantly, both products provide malware reports in the JavaScript Object Notation

(JSON) output format. There is no product comparison or ranking made in this project, nor was an evaluation of the different sandbox products on the market made.

2.2. Use cases

2.2.1. 500 Executables

To utilize a quantitative approach, a set of 500 executables was chosen to run through the LPMAP. These 500 samples are attachments extracted from the spam trap emails over two months in November and December 2022. All samples must pass all plugins, and most importantly, they must be uploaded to the Joe Sandbox and the Triage Sandbox. The sandbox run will reveal the functionality of the malware executables and may show additional downloads of other malware samples where the initial binary acts as a dropper. The malware samples will be executed after arriving in the spam traps to achieve the best results out of the LPMAP. The prompt execution is the only possibility to explore the whole infection chain as different malware download websites could be taken offline by law enforcement or other institutions. All findings need to be stored in different JSON files for further processing.

2.2.2. MISP Events

The main goal is to support the cyber security community. As sharing is caring, all 500 malware samples analyzed by the LPMAP will automatically create a MISP event if the sandboxes detect a specific malware signature and a C2 extraction is available. This helps reduce the number of MISP events and increases the quality of the events. The MISP events allow the community to identify, detect and monitor new IOCs within their network.

2.2.3. MalwareBazaar

To further help to improve the security of all organizations worldwide, the samples need to be shared on MalwareBazaar (MalwareBazaar, 2022), a freely available platform run by abuse.ch. MalwareBazaar provides the ability to upload and download malware samples for free. The only constraint is to allow MalwareBazaar to access the Twitter handle in case of a sample upload. Even if MalwareBazaar has access to the Twitter handle, it is still possible to share the samples anonymously, which will be the

case in this research paper. There is always the possibility that another user was faster in uploading a specific sample which means that not all of the analyzed malware samples are reported anonymously.

3. Findings and Discussion

The LPMAP is developed in Python with as few external libraries as possible to reduce the operation time and the attack surface. The setup includes an email server with spam trap email addresses and an analysis server built on Ubuntu Linux Server 20.04.5 LTS (Ubuntu, 2022). The email server and the analysis server were set up in advance and are not part of this research paper. The default Python Version is 3.8.10 on an Ubuntu 20.04.5 LTS, which implies that the tool was not tested with any other Python versions. Additionally, a Redis Database Server Version 5.0.7, Pipenv (Pipenv, 2022) Version 11.9, YARA (YARA, 2022) Version 3.0.9, and Suricata (Suricata, 2022) Version 6.0.8 are installed on the Ubuntu. These are all preliminary server and software requirements for this research paper. YARA and Suricata are optional and must only be installed if the YARA and PCAP plugins are used.

3.1. Lightweight Python-Based Malware Analysis Pipeline

The approach chosen for the whole LPMAP is plugin-based, which allows a very flexible structure that can be adapted quickly and extended with new plugins. The core builds the main.py, where everything is controlled. Each analysis gets an ascending task ID stored in the Redis database. After calling a plugin, the dataset in the Redis database is updated with predefined progress to track each task state in the Python Rich Malware Analysis Pipeline Framework (PRMAPF). Each plugin can be enabled or disabled in the configuration file with the possibility of defining the running order of the plugin execution. The data flow with the default plugin running order is illustrated in Figure 2.

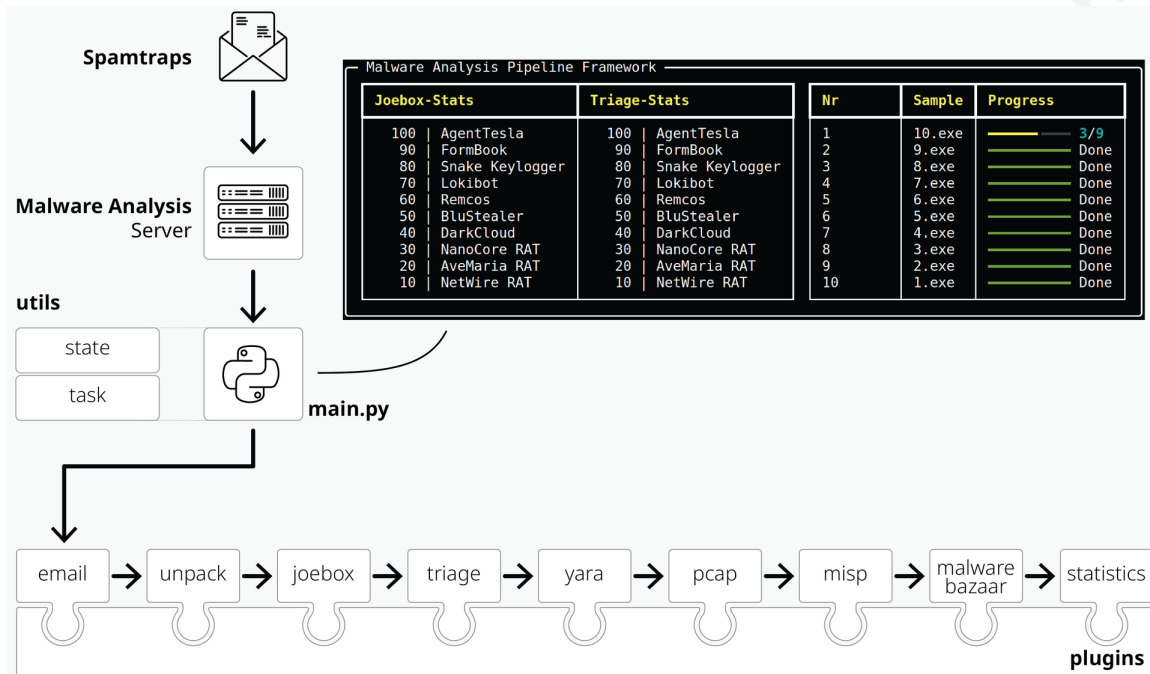


Figure 2: LPMAP with Python Rich Malware Analysis Pipeline Framework

3.1.1. Pipenv

Pipenv (Pipenv, 2022) is a tool that manages the virtual environment of Python projects and can install or uninstall packages from the Pipfile. It also creates the Pipfile.lock, which produces deterministic builds.

3.1.2. Pipfile

The Pipfile has a list of Python packages and software source repositories. All the listings must be installed to run the LPMAP. The packages in this research paper are installed in a virtual environment with pipenv. It is further declared to use Python Version 3.8, which is required to install and run the LPMAP. The installation happens through the Linux command (pipenv install).

3.1.3. Config.cfg (config_template.cfg)

Config.cfg lists all essential variables needed for the order of the plugins to be executed, to analyze emails, to create new directories, and different API keys or access credentials for JoeBox, Triage, MISP, MalwareBazaar, and Redis. There is a

config_template.cfg provided for new installations of the LPMAP. An example is listed in Figure 3 below, and all keywords in capital letters need to be adapted accordingly.

```
[plugins]
run_order = email, unpack, joebox, triage, yara, pcap, misp, malwarebazaar,
statistics

[emails]
new_emails_dir = PATH-TO-NEW-EMAILS-DIR
analysis_dir = PATH-TO-NEW-ANALYSIS-DIR

[collection]
analyzed_samples = PATH-TO-ALL-ANALYZED-SAMPLES/collection.json

[misp]
url = https://MISP-SERVER
key = MISP-API-KEY
verifycert = True

[malwarebazaar]
url = https://MALWAREBAZAAR-URL
api_key = API-KEY

[triage]
url = https://TRIAGE-URL
api_key = API-KEY

[joebox]
url = https://JOEBOX-URL
api_key = API-KEY

[yara]
rules_dir = PATH-TO-RULES-DIR

[suricata]
suricata_dir = PATH-TO-SURICATA-DIR
suricata_config = PATH-TO-SURICATA-CONFIG-FILE

[stats]
stats_dir = PATH-TO-STATS-DIR

[sandbox]
sandbox_list = joebox, triage

[redis]
```

```

host= HOST
port=PORT
password=PASSWORD

```

Figure 3: LPMAP Config Template

3.1.4. Python Rich Malware Analysis Pipeline Framework

Python Rich (GitHub Textualize, 2022) provides a great way to visualize the last ten samples of the LPMAP. It reads the different states of the already analyzed or currently analyzing samples and provides a graphical overview. Furthermore, the top ten recognized malware signatures of Joe Sandbox and Triage Sandbox are listed in a table. This framework runs entirely independently and is not attached to the main.py. It only reads the data provided by the main.py and its plugins which are stored in the Redis database or the statistics JSON files.

3.1.5. Main.py

The main.py controls the LPMAP. It generates new task IDs, imports utilities, and loads the environment, which calls the abstract plugin to iterate through all the defined plugins of the config.cfg.

3.1.6. Utility: State

Utilities are helper tools used by either the main.py or the plugins. The main functionality of the state utility is the config parser (config.cfg) with the ability to receive API keys, credentials, the plugin load order, and all paths of directories to store or load files on the analysis server.

3.1.7. Utility: Task

Each time a sample analysis starts, the main.py generates an ascending task ID and stores the information in the Redis database. The task utility is responsible for this job. Furthermore, each plugin will update the state of the running task with its progress which is then available and visually illustrated in the PRMAPF.

3.1.8. Plugin: Abstract

The Abstract plugin is the overlay of all plugins. It reads the configuration file (config.cfg) and loads the plugins in the order they are listed. This flexibility allows the user to change the order of the available plugins and add or remove plugins.

3.1.9. Plugin: Email

The Email plugin parses all incoming emails from the spam traps. Every email contains much information in the header and body, which helps correlate the attributes (IOCs) in MISP. The information and correlation in MISP can be used to identify different threat actors and their TTPs.

The following email header information is written to a JSON file, which is used in the MISP plugin for the event and attribute creation.

Header/Body Field Name	Description
Subject	The Subject is used for MISP to create an event info name and an attribute with the type email-subject for correlation reasons. The information is stored in a JSON file for further processing.
From	The From shows the sender's name of the email. This information can be faked, but it still can be helpful for correlation in MISP. The information is stored in a JSON file to provide the MISP plugin with the information for creating an attribute with the type email-src-display-name.
Received From	The Received From is usually a chain of sending and receiving mail servers/clients. The relevant information is the first sender Internet Protocol (IP) address for MISP correlation reasons again. The IP address is stored in a JSON file as well. The MISP plugin will use an attribute with the type ip-src.
Attachment Names	If there are attachments in the email, the attachments are extracted and will be further analyzed in the Unpack plugin. Furthermore, the email attachment name or names will be stored in a JSON file as

	email-attachment for further processing in the MISP plugin.
--	---

Figure 4: Parsed Email Header/Body

The header information (To) is not used in MISP as this would reveal the spam trap email addresses.

3.1.10. Plugin: Unpack

The Unpack plugin checks if the extracted attachments of the Email plugin are archive files. If this is true, the archives are uncompressed. It is also possible that the executable was directly attached and not compressed. In this case, the executable itself is used for further processing. The executables are then uploaded to the sandboxes in the following plugins.

3.1.11. Plugin: JoeBox

If the analyzed email has an attached binary in combination with the Unpack plugin, the sample is uploaded to Joe Sandbox. The sample is analyzed in a cloud-based virtual machine, and Joe Sandbox tries to reveal the malicious activities of the sample.

Joe Sandbox produces multiple report formats. Only the JSON report is downloaded and parsed. If available, the malware signatures and C2 configs are stored in a newly created JSON file. In addition, all memory dumps, PCAPs, and available binaries (downloaded and dropped) will be stored on the analysis server. This information will be analyzed in the following steps by the YARA and PCAP plugins.

The parsing of the JSON report is very important as the extracted data is used in the MISP plugin to create MISP events with the sandbox IOCs automatically. In order to detect changes in the report output by Joe Sandbox, the parsing needs to be checked regularly.

3.1.12. Plugin: Triage

Emails with attached binaries or binaries in archive files are uploaded to the Triage Sandbox. The Triage Sandbox runs the sample in a cloud-based virtual machine and tries to reveal the malicious activities of the sample.

Triage Sandbox creates multiple report formats. Only the JSON report is downloaded and parsed, and the malware signatures and C2 configs are stored in a JSON file, if available. Furthermore, all memory dumps and PCAPs will be stored on the analysis server. This information will be analyzed in the following steps by the YARA and PCAP plugins.

The parsing of the Triage report is essential as the MISP plugin uses the information to create new MISP events and attributes. The JSON report structure is usually different for different malware signatures, which implies that the plugin code for parsing needs to be checked regularly.

3.1.13. Plugin: YARA

YARA scans all memory dumps and binaries of the sandbox run and saves the output in a JSON file. The freely available ruleset of Malpedia (Malpedia, 2022) from the German Fraunhofer Institute is used for this scan. Only TLP:WHITE and TLP:GREEN (Traffic Light Protocol, 2022) YARA rules (YARA Rules, 2022) are used. FIRST (FIRST, 2022) updated the Traffic Light Protocol (TLP) to Version 2.0 in August 2022. The TLP:WHITE has been replaced with TLP:CLEAR, but Malpedia did not yet adapt this when this research paper was published.

Malpedia is an invite-only trust group. Only members can access further Malpedia YARA rules classified as TLP:AMBER or TLP:RED, which are not used in this research paper.

3.1.14. Plugin: PCAP

Suricata (Suricata, 2022) is a network analysis and threat detection tool. It can scan PCAPs with a specific ruleset to find malicious network traffic. There are multiple high-value rulesets on the market for free. The rulesets from Figure 5 are used to scan the generated PCAPs with Suricata.

Ruleset Source	Reference
SSL blacklist from abuse.ch	(SSL blacklist, 2022)
Feodo Tracker from abuse.ch	(Feodo Tracker, 2022)
URLhaus from abuse.ch	(URLhaus, 2022)
ET Rules from Proofpoint	(ET Rules, 2022)

Figure 5: Suricata Ruleset Source

Triggering rules are deduplicated and saved in a JSON file.

3.1.15. Plugin: MISP

The MISP plugin is one of the essential plugins as it reads all the generated JSON files of the other plugins to create a MISP event with its attributes. For the event creation, the following information listed in square brackets is parsed out of the JSON files.

MISP Event Fields	Value
Event Info	[MALWARE SIGNATURE] Spam Run ([DATE] – [EMAIL SUBJECT])
Tags	tlp:green, Crimeware, exe, [MALWARE SIGNATURE]
Threat Level	Low
Analysis	Completed
Distribution	This community only

Figure 6: MISP Event

After the event creation, all the relevant attributes are added to the event, which MISP automatically correlates. The correlation allows the MISP users to see if specific attributes were already used in other malspam waves of the same malware family or even from another malware family, showing that certain threat actors share their infrastructure. The attributes automatically added, based on the information of the JSON files, are listed in Figure 7.

Category	Type	IDS	Value
Payload delivery	email-subject		[SUBJECT]
Payload delivery	email-attachment		[ATTACHMENT NAME]
Payload delivery	email-src-display-name		[SENDER IP]

Network activity	domain/hostname/url/email-dst/	x	[C2 CONFIG]
Payload delivery	Object:file	x	[ATTACHMENT FILE] and MISP auto-generated hashes, size, and entropy

Figure 7: MISP Attributes

3.1.16. Plugin: MalwareBazaar

All executables tagged with a malware signature by at least one of the two sandboxes are uploaded to MalwareBazaar (MalwareBazaar, 2022). Samples not yet known to MalwareBazaar are reported anonymously.

3.1.17. Plugin: Statistics

The statistics plugin parses the reports of the different sandboxes and updates the statistics JSON files of each sandbox if there is a malware signature available. Each sandbox's top ten malware families are printed and updated automatically in the PRMAPF.

3.2. Research Results

The research results are based on the use cases from the Research Method. Five hundred executables were sent through the LPMP, and each plugin created the predefined information needed to generate statistics out of the data.

3.2.1. Sandbox Signature Detection

The following tables in Figure 8 and Figure 9 illustrate the output of the malware signature detection. The sandboxes are called Sandbox 1 and Sandbox 2, not to make any attribution as comparison and ranking are not part of this research paper.

Malware Signature Sandbox 1	Count	Signature Reference
AgentTesla	150	(AgentTesla, 2022)
FormBook	90	(FormBook, 2022)
Snake Keylogger	30	(Snake Keylogger, 2022)
Lokibot	30	(Lokibot, 2022)
Remcos RAT	16	(Remcos RAT, 2022)
DarkCloud	10	(DarkCloud, 2022)
BluStealer	6	(BluStealer, 2022)

AveMaria RAT	4	(AveMaria RAT, 2022)
NanoCore RAT	3	(NanoCore RAT, 2022)
NetWire RAT	2	(NetWire RAT, 2022)
GuLoader	2	(GuLoader, 2022)
Azorult	2	(Azorult, 2022)
Quasar RAT	2	(Quasar RAT, 2022)
HawkEye	2	(HawkEye, 2022)
DBatLoader	1	(DBatLoader, 2022)
No signature detection	148	
Total	500	

Figure 8: Sandbox 1 Statistics

Malware Signature Sandbox 2	Count	Signature Reference
AgentTesla	205	(AgentTesla, 2022)
FormBook	83	(FormBook, 2022)
Snake Keylogger	24	(Snake Keylogger, 2022)
Lokibot	22	(Lokibot, 2022)
Remcos RAT	12	(Remcos RAT, 2022)
BluStealer	8	(BluStealer, 2022)
AveMaria RAT	4	(AveMaria RAT, 2022)
NanoCore RAT	3	(NanoCore RAT, 2022)
NetWire RAT	2	(NetWire RAT, 2022)
DarkCloud	2	(DarkCloud, 2022)
No signature detection	135	
Total	500	

Figure 9: Sandbox 2 Statistics

The output of both sandboxes looks very similar. AgentTesla and FormBook are the top malware families, and there is no detection for about 30% of the malware samples analyzed. No detection means no malware signature was found — this can be the case if the malware cannot be identified or if the sample did not detonate. However, all samples with no detection were also flagged as malicious or suspicious by both sandboxes, which means that they are probably somehow malicious. An excellent example of a suspicious file would be AnyDesk (AnyDesk, 2022), a remote-controlling software. AnyDesk is a legitimate tool, but threat actors often use it to control infected clients. If a company does not use AnyDesk throughout its organization, this should be a red flag. AnyDesk is very likely to be used for malicious purposes in this case.

3.2.2. YARA Detection

The statistics in Figure 10 and Figure 11 list all hits of the YARA rules for Sandbox 1 and Sandbox 2. Each malware sample has multiple memory dumps from the sandboxes and at least one binary that can trigger a YARA hit, which means that different rules can trigger one or multiple hits. If there were multiple hits of one rule, the amount is represented as 1, but it is still possible that other YARA rules have the count of 1 as well for this specific sandbox run.

Malware Signature Sandbox 1	Count	Signature Reference
win_formbook_g0	102	(FormBook, 2022)
win_formbook_auto	102	(FormBook, 2022)
win_agent_tesla_g1	36	(AgentTesla, 2022)
win_lokipws_g0	28	(Lokibot, 2022)
win_lokipws_auto	28	(Lokibot, 2022)
win_karius_g0	24	(Karius, 2022)
win_remcos_auto	17	(Remcos RAT, 2022)
win_vobfus_auto	11	(Vobfus, 2022)
win_masslogger_w0	9	(MASS Logger, 2022)
win_agent_tesla_g2	6	(AgentTesla, 2022)
win_nanocore_w0	5	(NanoCore RAT, 2022)
win_ave_maria_g0	5	(AveMaria RAT, 2022)
win_ave_maria_auto	5	(AveMaria RAT, 2022)
win_nymaim_g0	3	(Nymaim, 2022)
win_retefe_g0	2	(ReteFe, 2022)
win_netwire_g1	2	(NetWire RAT, 2022)
win_netwire_auto	2	(NetWire RAT, 2022)
win_suppoBox_auto	1	(SuppoBox, 2022)
win_hawkeye_keylogger_w0	1	(HawkEye, 2022)
win_hawkeye_keylogger_g0	1	(HawkEye, 2022)
win_hawkeye_keylogger_auto	1	(HawkEye, 2022)
win_cannon_auto	1	(Cannon, 2022)
Total	392	

Figure 10: YARA Statistics Sandbox 1

Malware Signature Sandbox 2	Count	Signature Reference
win_formbook_g0	83	(FormBook, 2022)
win_formbook_auto	83	(FormBook, 2022)
win_xfilesstealer_auto	37	(X-Files Stealer, 2022)

win_lokipws_g0	26	(Lokibot, 2022)
win_lokipws_auto	26	(Lokibot, 2022)
win_remc0s_auto	12	(Remcos RAT, 2022)
win_vobfus_auto	9	(Vobfus, 2022)
win_nanocore_w0	6	(NanoCore RAT, 2022)
win_masslogger_w0	4	(MASS Logger, 2022)
win_dbatloader_g1	4	(DBatLoader, 2022)
win_ave_maria_g0	4	(AveMaria RAT, 2022)
win_agent_tesla_g2	4	(AgentTesla, 2022)
win_netwire_g1	2	(NetWire RAT, 2022)
win_netwire_auto	2	(NetWire RAT, 2022)
win_hawkeye_keylogger_auto	2	(HawkEye, 2022)
win_hawkeye_keylogger_w0	1	(HawkEye, 2022)
win_hawkeye_keylogger_g0	1	(HawkEye, 2022)
Total	306	

Figure 11: YARA Statistics Sandbox 2

Compared to the detection rate of the sandboxes, YARA detected only several AgentTesla samples with the provided memory dumps and binaries. The low detection rate could either be because the rules provided by Malpedia are not triggering new versions of AgentTesla or because the memory dumps would need to provide more information to trigger any search parameters in the case of AgentTesla. In either case, this will need to be investigated to develop an AgentTesla rule that reliably triggers.

3.2.3. Suricata detection

Each malware sample analyzed by the sandboxes generated one PCAP file per sandbox with the captured network traffic. The statistics of Sandbox 1 and Sandbox 2 are merged together as the rulesets of abuse.ch and Proofpoint, with an amount of 135898 rules, produced a high number of hits. The statistics are illustrated in Figure 12.

Suricata Rule Message Sandbox 1 and 2	Count
ET MALWARE FormBook CnC Checkin (GET)	2577
ET MALWARE LokiBot User-Agent (Charon/Inferno)	2086
ET MALWARE LokiBot Checkin	2077
ET MALWARE LokiBot Request for C2 Commands Detected M1	2030
ET MALWARE LokiBot Fake 404 Response	2023
ET MALWARE LokiBot Request for C2 Commands Detected M2	1492

ET DNS Query for .su TLD (Soviet Union) Often Malware Related	661
ET POLICY HTTP Request to .su TLD (Soviet Union) Often Malware Related	659
URLhaus Known malware download URL detected (2427338)	362
URLhaus Known malware download URL detected (2427336)	95
ET INFO DYNAMIC DNS Query to *.dyndns. Domain	90
ET INFO DYNAMIC DNS Query to *.duckdns. Domain	69
ET INFO Observed DNS Query to .world TLD	68
ET HUNTING Telegram API Domain in DNS Lookup	59
ET MALWARE LokiBot Application/Credential Data Exfiltration Detected M1	56
ET MALWARE AgentTesla Exfil Via SMTP	56
ET HUNTING Observed Telegram API Domain (api.telegram.org in TLS SNI)	51
ET MALWARE LokiBot Application/Credential Data Exfiltration Detected M2	48
ET POLICY External IP Lookup - checkip.dyndns.org	45
ET MALWARE 404/Snake/Matiex Keylogger Style External IP Check	45
ET HUNTING Telegram API Certificate Observed	34
ET INFO HTTP Request to a *.buzz domain	31
ET HUNTING Request to .XYZ Domain with Minimal Headers	28
ET HUNTING Suspicious Terse Request for .bmp	19
ET POLICY DNS Query to DynDNS Domain *.ddns.net	17
ET MALWARE Possible NanoCore C2 60B	15
ET DNS Query to a *.top domain - Likely Hostile	15
ET MALWARE AgentTesla Exfil via FTP	12
ET JA3 Hash - Remcos 3.x TLS Connection	11
ET INFO HTTP Request to a *.top domain	10
ET INFO Commonly Abused File Sharing Domain in DNS Lookup (filetransfer.io)	9
ET HUNTING Request to .TOP Domain with Minimal Headers	9
ET POLICY DNS Query to DynDNS Domain *.hopto.org	8
ET MALWARE Remcos 3.x Unencrypted Server Response	7
ET INFO Commonly Abused File Sharing Domain (filetransfer.io in TLS SNI)	6
ET POLICY IP Check Domain (icanhazip.com in HTTP Host)	5
ET MALWARE Remcos 3.x Unencrypted Checkin	5

ET MALWARE Ave Maria/Warzone RAT Encrypted CnC Checkin	4
ET MALWARE AgentTesla HTML System Info Report Exfil via FTP	4
ET INFO Namecheap URL Forward	4
ET HUNTING HTTP POST to XYZ TLD Containing Pass - Possible Phishing	4
ET DROP Spamhaus DROP Listed Traffic Inbound group 13	4
URLhaus Known malware download URL detected (2431551)	3
URLhaus Known malware download URL detected (2431550)	3
URLhaus Known malware download URL detected (2428660)	3
ET POLICY Observed DNS Query to File Transfer Service Domain (transfer .sh)	3
ET MALWARE Win32/AZORult V3.3 Client Checkin M14	3
ET INFO Observed Discord Domain in DNS Lookup (discordapp .com)	3
ET INFO Observed Discord Domain (discordapp .com in TLS SNI)	3
ET INFO HTTP POST Request to Suspicious *.icu domain	3
ET INFO Commonly Abused File Sharing Site Domain Observed (transfer .sh in DNS Lookup)	3
ET POLICY External IP Lookup ip-api.com	2
ET MALWARE Common RAT Connectivity Check Observed	2
ET MALWARE AZORult v3.3 Server Response M2	2
ET INFO Observed DNS Query to .life TLD	2
ET INFO Observed Discord Domain in DNS Lookup (discord .com)	2
ET INFO Observed Discord Domain (discord .com in TLS SNI)	2
ET INFO HTTP Request to Suspicious *.life Domain	2
ET INFO Commonly Abused File Sharing Site Domain Observed (transfer .sh in TLS SNI)	2
URLhaus Known malware download URL detected (2427337)	1
ET MALWARE Ave Maria/Warzone RAT Encrypted CnC Checkin (Inbound)	1
ET INFO Observed DNS Query to .work TLD	1
ET INFO DNS Query for Suspicious .icu Domain	1
ET INFO DNS Query for Suspicious .gq Domain	1
ET DNS Query for .cc TLD	1
Total	14959

Figure 12: Suricata Statistics Sandbox 1 and Sandbox 2

FormBook and Lokibot are the top malware signatures. AgentTesla, which has a low YARA detection rate, is better detected by Suricata. It is either flagged with AgentTesla in the rule message itself or Telegram (Telegram, 2022), which is one of the multiple exfiltration mechanisms.

3.2.4. MalwareBazaar Upload

All malware samples analyzed by LPMAP with a malware signature were uploaded during the analysis and are available on MalwareBazaar. The Hashes (MD5, SHA1, SHA256) and the executable names are listed in the Appendix.

The samples are either searchable directly via URL on MalwareBazaar [https://bazaar.abuse.ch/browse.php?search=sha256%3A\[SHA256-HASH\]](https://bazaar.abuse.ch/browse.php?search=sha256%3A[SHA256-HASH]) (example of a SHA256 hash search) or via the input search field on <https://bazaar.abuse.ch/browse.php> with the syntax sha256:[SHA256-HASH]. In both examples, [SHA256-HASH] must be replaced with the actual hash, including removing the square brackets.

MALWARE bazaar
by ABUSE

Browse Upload Hunting API Export Statistics FAQ About Login

MalwareBazaar Database

You are browsing the malware sample database of MalwareBazaar. If you would like to contribute malware samples to the corpus, you can do so through either using the [web upload](#) or the [API](#).

452
Submissions (past 24 hours)

AgentTesla
Most seen malware family (past 24 hours)

601'198
Malware samples in corpus

Using the form below, you can search for malware samples by a hash (MD5, SHA256, SHA1), imphash, tlsh hash, ClamAV signature, tag or malware family.

Browse Database

Search:

Date (UTC)	SHA256 hash	Type	Signature	Tags	Reporter	DL
2022-11-22 06:17	a78731da2617fe258f08...	exe	AgentTesla	AgentTesla EXE	Anonymous	

Showing 1 to 1 of 1 entries

Previous 1 Next

© abuse.ch 2022

Figure 13: Screenshot MalwareBazaar

Author Name, email@address

Sharing all the malware samples allows the cyber security community to either analyze the executables themselves or to search for the hashes on the devices in their network to detect infected devices.

3.2.5. MISP Events

For all samples with a malware signature and a C2 config extraction, a MISP event has been created. One example of the most occurring malware families, called AgentTesla (AgentTesla, 2022), is listed in Figure 14.

AgentTesla Spam Run (2022-11-22 - Order Confirmation : 4170027132)

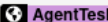
Event ID	18502
UUID	77cf7921-4ac6-4693-b915-c21d60e4333b 
Source Organisation	
Member Organisation	
Creator user	
Protected Event (experimental) 	 Event is in unprotected mode. Switch to protected mode
Tags	 x  x  x  x  
Date	2022-11-22
Threat Level	 Low
Analysis	Completed
Distribution	This community only  
Published	 2022-11-22 08:14:02
#Attributes	23 (2 Objects)
First recorded change	2022-11-22 08:13:56
Last change	2022-11-22 08:13:56
Modification map	
Sightings	0 (0) - restricted to own organisation only. 
Correlation	Enabled (disable)

Figure 14: Screenshot of AgentTesla MISP Event

The MISP plugin used the information gained and stored in the JSON files of the different plugins. For the event creation, the malware signature, the current date, and the subject are used for the event info. In addition, the tag tlp:green is added to each event to share the information as far as possible. The Crimeware tag indicates the threat level, and the tag exe to provide information about the analyzed sample. The malware signature tag

AgentTesla in this example was added to give another correlation or search option in MISP.

Each MISP event usually has one or multiple attributes to share as much information as possible. The AgentTesla example event has 23 attributes listed in Figure 15 below (some attributes are redacted due to security reasons).

Date	Org	Category	Type	Value	Tags	Galaxies	Comment	Correlate	Related Events	Feed hits	ID5
2022-11-22		Payload delivery	email-subject	Order Confirmation - 4170027132							
2022-11-22		Payload delivery	email-attachment	8236537.zip							
2022-11-22		Payload delivery	email-src-display-name								
2022-11-22		Network activity	ip-src				Sending Mailserver IP				
2022-11-22		Network activity	email-dst	opmg@jackbarber.com			Exit Email Address				
2022-11-22		Network activity	email-dst	officestore2022@gmail.com			Exit Email Address		Too many correlations		
2022-11-22		Network activity	hostname	mail.jackbarber.com			Exit Email Server		18432 18462 18484 18501		
2022-11-22 Object name: file: C: References: 1											
2022-11-22		Payload delivery	filename:	8236537.exe							
2022-11-22		Other	size-in-bytes:	504320							
2022-11-22		Other	size-in-bytes:	492.50 KB							
2022-11-22		Other	entropy:	7.7477879702149							
2022-11-22		Payload delivery	md5:	e488c28429957e192b805d58a20c17c							
2022-11-22		Payload delivery	sha1:	d0961a064e5c9581a5d1a0a98c8a8d8c7574f2							
2022-11-22		Payload delivery	sha256:	a78731da2617ba25808a308a02775a0662b363a5b4197aa61a0ba178a0							
2022-11-22		Payload delivery	sha512:	12759889eae95632d7bca573a6f8ba542bd30f6ae27b190e8495710a08							
2022-11-22		Payload delivery	malware-sample:	8236537.exe							
2022-11-22		Payload delivery	malware-sample:	e488c28429957e192b805d58a20c17c							

Figure 15: Screenshot of AgentTesla MISP Attributes

Essential attributes are the subject, the attachment name, the email display name, the email sender source IP, the attachment, and the C2 configuration. AgentTesla often uses hacked email accounts to exfiltrate the data, and this is the case as well in the listed example. Other exfiltration mechanisms of AgentTesla are either Telegram Messenger (Telegram, 2022) or POST requests to native C2 servers or hacked web servers.

The information about this MISP event with its attributes can now be shared with the cyber security community.

All analyzed malware samples generated a total amount of 330 MISP events for sharing purposes. Events were only created if there was a malware signature detection and a C2 config extraction by the sandboxes. The malware signatures sorted by their count are listed in Figure 16.

MISP Event Malware Tags	Count	Signature Reference
AgentTesla	234	(AgentTesla, 2022)
Lokibot	31	(Lokibot, 2022)
SnakeKeylogger	28	(Snake Keylogger, 2022)
RemcosRAT	12	(Remcos RAT, 2022)
BluStealer	9	(BluStealer, 2022)
NanoCore	5	(NanoCore RAT, 2022)
AveMaria RAT	4	(AveMaria RAT, 2022)
DarkCloud	3	(DarkCloud, 2022)
NetWire	2	(NetWire RAT, 2022)
QuasarRAT	1	(Quasar RAT, 2022)
HawkEye	1	(HawkEye, 2022)
Total	330	

Figure 16: MISP Event Statistics

There was no MISP event creation for FormBook as the C2 config extraction is out of scope by both sandbox vendors. The reason behind this is the functionality of FormBook, which uses many decoy URLs, and according to Check Point (Check Point, 2022), the samples would need to run for at least ten minutes to determine the actual C2 server.

4. Recommendations and Implications

The LPMAP is developed only in Python and uses no third-party frameworks to reduce operational tasks and minimize the attack surface. The whole pipeline is based on plugins to be as flexible as possible and to provide an easy way to develop further demanded features.

4.1. Recommendations for Practice

It is recommended to use the Pipfile for installing the dependencies and the `config_template.cfg` to steer all plugins, working directories, and API keys. Furthermore, the LPMAP was developed with Python Version 3.8 and was only tested with this version. Any other Python version needs to be tested thoroughly before going live in a production environment. An installation instruction is attached in the Appendix to set up the perfect environment for LPMAP, which includes the operating system and all packages needed for the LPMAP.

4.2. Implications for Future Research

There are many possibilities for future research, mainly in developing areas, which implies that the plugin-based approach is used to avoid complications with the whole process chain.

Further research could provide a common way to deploy the LPMAP as a Docker image, which would verify the dependencies of the Python version and the required Python packages to work correctly. YARA and Suricata could be deployed through Docker as well.

Another possibility of future research includes the development of the plugins. Many sandbox providers on the market have different JSON output reports. Additional sandbox parsing plugins would increase the benefit for the cyber security community, especially parsing plugins for Open-Source Sandboxes like Cuckoo (Cuckoo, 2022) or CAPE (CAPE, 2022).

The Statistics plugin is based on an overall signature detection of each sandbox. It could be further improved by comparing signature detections of different sandboxes or daily, weekly, monthly, or yearly statistics.

A future research area would be developing new YARA rules, especially an AgentTesla rule, which would provide the same amount or at least an approximation of hits as the sandboxes.

5. Conclusion

In conclusion, the LPMAP improves the analysis and sharing process in the cyber security community. The operational tasks of updating and maintaining the tool are minimal, and without any web frameworks or other internet-facing services, the attack surface can be considered low. Furthermore, the LPMAP is easy to install, use, and develop further.

There is no easy-to-use and fully-fledged malware framework available on the market to analyze email attachments in different sandboxes and to share the TTPs of the reports in an automated way.

In particular, the LPMAP can analyze many malware samples a day. It provides an automated way to extract IOCs from reports, scan memory dumps and binaries with YARA, detect threats with Suricata rules and share all the gained information in MISP with the cyber security community.

Finally, operating a Malware Analysis Pipeline can also help to better evaluate the current threat landscape with the information gained over time.

References

AgentTesla (2022, November 21). *Malpedia*. Retrieved from

https://malpedia.caad.fkie.fraunhofer.de/details/win.agent_tesla

AnyDesk (2022, November 25). *Remote Controlling Software*. Retrieved from

<https://anydesk.com>

AveMaria RAT (2022, November 21). *Malpedia*. Retrieved from

https://malpedia.caad.fkie.fraunhofer.de/details/win.ave_maria

Azorult (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.azorult>

BluStealer (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.blustealer>

DarkCloud (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.darkcloud>

DBatLoader (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.dbatloader>

Cannon (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.cannon>

CAPE (2022, November 21). *Open Source Sandbox*. Retrieved from

<https://capesandbox.com>

Check Point (2022, November 29). *FormBook Functionality*. Retrieved from

<https://research.checkpoint.com/2021/stealth-is-never-enough-or-revealing-formbook-successors-cc-infrastructure/>

Cuckoo (2022, November 21). *Open Source Sandbox*. Retrieved from

<https://cuckoosandbox.org>

Author Name, email@address

Drakvuf (2022, November 13). *Sandbox CERT.pl GitHub*. Retrieved from

<https://github.com/CERT-Polska>

ET Rules (2022, November 24). *Emerging Threat Rules from Proofpoint*. Retrieved from

<https://rules.emergingthreats.net/open/suricata-6.0.8/emerging.rules.tar.gz>

Feodo Tracker (2022, November 24). *Feodo Tracker Abuse.ch*. Retrieved from

<https://feodotracker.abuse.ch/blocklist/>

FIRST (2022, November 13). *FIRST*. Retrieved from

<https://www.first.org/tlp/>

FormBook (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.formbook>

GitHub Textualize (2022, November 14). *Rich*. Retrieved from

<https://github.com/Textualize/rich>

GuLoader (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.cloudeye>

HawkEye (2022, November 21). *Malpedia*. Retrieved from

https://malpedia.caad.fkie.fraunhofer.de/details/win.hawkeye_keylogger

Joe Sandbox (2022, November 15). *Joe Security Sandbox Company*. Retrieved from

<https://www.joesecurity.org>

Karius (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.karius>

Karton (2022, November 13). *Malware Analysis Pipeline CERT.pl GitHub*. Retrieved

from <https://github.com/CERT-Polska>

Lokibot (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.lokipws>

Author Name, email@address

Malpedia (2022, November 16). *Malpedia is a free service offered by Fraunhofer FKIE.*

Retrieved from <https://malpedia.caad.fkie.fraunhofer.de>

MalwareBazaar (2022, November 16). *Free Malware Platform.* Retrieved from

<https://bazaar.abuse.ch>

Malwarebytes (2017, September 15). *Explained: YARA Rules.* Retrieved from

<https://www.malwarebytes.com/blog/news/2017/09/explained-yara-rules>

MASS Logger (2022, November 21). *Malpedia.* Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.masslogger>

MISP (2022, November 16). *Malware Information Sharing Platform.* Retrieved from

<https://www.misp-project.org>

mquery (2022, November 23). *YARA queries CERT.pl GitHub.* Retrieved from

<https://github.com/CERT-Polska/mquery>

MWDB (2022, November 23). *Malware Database CERT.pl GitHub.* Retrieved from

<https://github.com/CERT-Polska/mwdb-core>

NanoCore RAT (2022, November 21). *Malpedia.* Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.nanocore>

NetWire RAT (2022, November 21). *Malpedia.* Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.netwire>

Nymaim (2022, November 21). *Malpedia.* Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.nymaim>

Pipenv (2022, November 19). *Python Development Workflow for Humans.* Retrieved

from <https://pypi.org/project/pipenv/>

Quasar RAT (2022, November 21). *Malpedia.* Retrieved from

https://malpedia.caad.fkie.fraunhofer.de/details/win.quasar_rat

Author Name, email@address

Remcos RAT (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.remcos>

Retefe (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.retefe>

Sandbox (2022, November 16). *VMRAY Sandbox Company*. Retrieved from

<https://www.vmray.com/glossary/malware-sandbox/>

Snake Keylogger (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.404keylogger>

SuppoBox (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.suppobox>

SSL blacklist (2022, November 24). *SSL blacklist Abuse.ch*. Retrieved from

<https://sslbl.abuse.ch/blacklist/>

Suricata (2022, November 24). *Network Analysis and Threat Detection*. Retrieved from

<https://suricata.io>

Telegram (2022, November 25). *Telegram Messenger*. Retrieved from

<https://telegram.org>

Triage (2022, November 15). *Recorded Future Sandbox Company*. Retrieved from

<https://tria.ge>

Ubuntu (2022, November 18). *Ubuntu Linux*. Retrieved from

<https://ubuntu.com>

URLhaus (2022, November 24). *URLhaus Abuse.ch*. Retrieved from

<https://urlhaus.abuse.ch/downloads/suricata-ids/>

Vobfus (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.vobfus>

Author Name, email@address

YARA (2022, November 16). *YARA Documentation*. Retrieved from

<http://virustotal.github.io/yara/>

YARA Rules (2022, November 16). *YARA Ruleset Download*. Retrieved from

https://malpedia.caad.fkie.fraunhofer.de/api/get/yara/tlp_green/zip

X-Files Stealer (2022, November 21). *Malpedia*. Retrieved from

<https://malpedia.caad.fkie.fraunhofer.de/details/win.xfilesstealer>

Appendix

Installation Instructions

Linux Command	Comment
sudo apt install python3	
sudo apt install redis-server	
sudo apt install unrar	
sudo apt install p7zip-full	
sudo apt install pipenv	
pipenv install	Execute in the same directory as the Pipfile is
sudo add-apt-repository ppa:oisf/suricata-stable	Add the Suricata Repository
sudo apt install suricata	Optional
sudo apt install yara	Optional

Hash list of all analyzed Samples

MD5
01a6762f4e8956f4446e57f25a04e91b
0294badebae62dd7e03d329fb35b2395
031b6863d3bba189c39af5dff13a811e
03dbf0ff9f2eb35f107cc02c36087a4a
0597c2f492733078cc3231b33dfc284c
062c994cf266e117ed44718ad6fc3be1
06529107d330136c7e1dc44e7e3022d0
06d253413aa62c1eb72edb9fdc6e2a87
06e01e2bed6e92c22c2b1ba2c229b87e
071d5dc67ed8f6be856d7f23d77382fc
07ea7c18f551e8f176334f4d3c9ebfde
0925c5c4999f045c2e9892ac472aea07
094fd52eec0055205bddc82c5f78ad87
095a9596cfc5ec3df2c07fe963813774
09e55db332706365496bc97d217fb1e0
0a8b7aea0bee15d7b7c6b2d2d1964c6f
0ac349c67a718638cfb1fe2f70d43a90
0acb5bcb968b08f9fa0275337eaf9d81
0bb87df8b4cc8098bae7d574c201d60f
0c0ae190534c38a5bd7e496614533f90
0c1873c2115a4b98653c2a0c7be992eb
0c4c8f9ac7e81192bcdce5ca14ab4b79
0cb9ae3bbda860d66aecf80bb0ecdded

0cefb2a275a4d5ae0aff3578abf2859b
0d7486b187950b8ceb9aa40a9c9d554c
0d89761c0575ae2e87f25c1e2c49dd36
0dc0a3e0e5e66b83a24c2172bdcf3132
0e30fecaae8ee23fe43098d0ee630e20
0e3d9093a78c09e35a878781b8324dce
0e4b935de0d3a5f7f55dcc7e2895e8fa
0e91382bb8a69d18863e7e10fcc20131
0eb0b95e7aec335c3c43d1be5b97ba12
0eff673b1a7599f671ce402422e8b2de
0f46ecb9b5ea6f5fcbaf7056c316259c
0fbac84ae97242d3336c486f3886d058
0fe9d83b8acf61cbe617559bc3a9050b
121b82c12327e064193cbd25839526d8
1366f09969c727410c578b1cbbc3d7bb
1427fec2cb5f8c7e4f2af9fe5bb48086
143bcb812ed5583a70d2a5c425ed0b81
151d8349aa1f8d3697c55b1707ca2789
15e24631ce2fa4348558873e04be83c2
160157f45f4eab159c8d7b45b3993d11
1631c00ad821f9e97dcde8313a4be766
17563b233e39ca7bf0609eb018b905be
179f1f83612b983110b77fa97a7ce3d3
17a3633b6f0ddc4b4429163f3469fa81
18bb5d9bdba44a27341244d0d988e8a7
18bc83da8bfabb01740276062d6e014e
19081ef2a08f678a3203b29124043c41
19e9113d2843a2dc847c96a0a318c145
1a245500e0696bb8d89aebd5acd1bee1
1d52db123623884f2e1d373111726fb6
1eb74696c7f84fbb061c31eb5095e421
1ef2d59b86e6dfa3ac440b35acd5a3a5
1ef6b427350035123f4377766dfbe5fb
1f295eb290b724602850f7dc434a6b1f
1f2b86c70d821521cbe006e55e60e0a7
205083f15a3856bc55543a1584fc5e26
205d07f059cc7a32e840104cfeec332b
20c4471b875e64c7943c225a359eb227
21eb8736dd89da5191e78fde4a3ca3da
22b87e053fdb1c5b68d98b102f572d96
22eda30b572ee1c13f93a00d2d617d23
2485ac95e5abe631852b04c33a753661
250a617db5c9bd32442f17d32193799e
25d443aa756943edbc65d1ad7ffd5dbe

27450bbe35168d0879acedb48e58fef6
27cef81014b5be71f943e847feb417f1
28698eddeca31ab58dfe2bee8d0c2064
289419437b2e4fadbe5e5782091424ab
28cb97f71b423eb54c7382aba89d92d5
28fa19adec2fa9a8d1e8d67310d58f0b
2954613f38bdc982aa2c91e6f40c89e3
29d854c5c94cc9c0f35b50187fcf9e23
2a48228a37fe7bflcf3f147d07884e30
2a8437657c19ea56776d1757f3ba14a9
2b076b3cb90090de6feaa2741a266538
2c27fdafaa47f61b79d8b5513c8e4d23
2c948d99fb2c74bc2e1065a83c9ac423
2d13ccc54d7cc8d4d73febd4827575be
2daa48f1b8e1b5ab35c8085b82dc702f
2dacb53068ded68ee956fdd504737609
2daffeb4bfe82105ae4f2a8a0285e452
2dcae6df92afaf02c7b46ab27b15a2a9
2e0056d9d1dcdcca4dd886cb37e40a52
2e20cd4c0bacb0215d89790119b4c07d
2f3d22c63646f27c67fdc037df70fc8a
2fe4a36cee28261ccdc336a7618e4749
2ffe57a9542324cb976388289368e47f
3016dd22c9597fbf6c14fb3161325e5d
301addb86ca3c942e69305684bf5c91f
30721415f4d64272f30452e60fa5d5fb
31b8df5525ad7115281ebd2dfd9350ee
33285ede30ca1312a7e7e33e2830dd2e
33b9a77f32a9203a010adf8ecc700717
344e038bc93ae48700830b37b78beb36
34abbef1de3ceadeba78308423309942
3658c7ca2a9b3772906f1b5302cb9d6a
366445ea6bb0779f3003ec5c773f3547
36775a9ec5a3137929b7e84c25192916
36ad0d4a388265a5c56f4f44521f7e02
36d6ab065f6f526fla163cc757431a0d
37a2892bad5d99ee10da3a9b1bcf8688
384c1108356687200fa75e23c9880279
38a401260efb9031eb5d52a849eb005a
38b875adacdb9c591c5fddd6ae6bf5ac
394f30cac9eef76036a281aba4a390ac
39e8d664f0bb5dd48475c56c12736f5f
3ba416e6e7658f9974da264d1c907736
3c1446661a4699bd8b1e30e50edb5033

3c3936efafb137c48ce328997735d9ab
3c41bd7a89f4aaa559fd8d54455d8f9a
3c4a41de7f3ffa23b12148a28b4ba8a4
3c8406ac154bba363837d323a2a5ccee
3cbd0f4d5317f7be2027c49c21c4f7b24
3cd5112e9d9cd2bc8ed794401cc271e9
3d54cd16d34f2285ec1d903a12665c19
3e4780b2020437038d5a2b1d1908f374
3e8c3834f4beddb299985d328f112324
3ea9f0ce1e3390844bfb4ddf764af981
3f486093d92efa95488e0a867c13198f
3fb4490c63991e8f6a6a7e80fe89fe05
3fe6259ed37afe425f5062f917897fe8
4048c5dd9ed4814eff5ba22c7066556e
423bfb5d292ce0429c67cb4a3c177afa
42a3b27df3ba1849780e526c42de50ab
42d5a7c3be7f0c927360d76c8f2179da
439aea645f0c55ddefe8ec84fea83669
43a4f2ccb8e422d51aea43852ce0c6e
43e53eabb6adf70e71847186b77f6e06
44b06cdec6dbf4a26c40219a71b61574
44d628546ab1eff55064627d70a3cb27
45422b5f2939489d060f6be8338ffd4d
455635d44d69c0f56aad4f8376b485dd
45bfc44bb1648f19fd5db3a19be64af5
461f8ee62f06524cdc29cad3170314f7
463a81fbfbca4cf281d7f58730a8067b
46932de3d79b8b31004320d4a8900ece
46ef6e22c43ddc8b0331a4fd871076a7
46f51ef279ef81ceb6dcd1557284644a
476864654b5278a227f4b345f3a8b581
47b60cbad0166a779665d56fe5c03596
48acdb724116bacb0c42033d25c0ba4f
49b30367cc4e82565b22cf3299d673c0
4a453e7884ac6833d222c0085d719b0c
4a5535975bf6294c7f513b06032f15f6
4b0aae802eb153655ab48663ac2ba56e
4b157f58016445399705b1e1aa57d282
4ca463548b17053bde8149d3a693d832
4cf810e991daa5f34fea55f77dc6ec8e
4d4bc8005b5de5341caf4a1f33b4d51c
4f2f5a6f0f1755fbc84a6c4ad5aa8cd3
4f88b5c2537d2286b6efee0d354bd0f3
505852f2cd67a14131d2d6e927d92889

50c2557efa346a375062c2a9ac4d7e8f
50f2cdf103afa0e34d2e51b8e16c5375
51a9b11d435e2ad05acb7507e5c42478
51dcaal09223f28e429fd5d6c2403b0
523dc49d6c29ff597f7fae946ce4ad86
52bafc4196a3d4ae8082e80e4608f73b
5302a8a452ce7bc2e00673418a0b9e0c
532984481d6e07209aa8e07623e015d4
534bdf7a4229c0a9e3c6959aa871e666
54090c16cbf3c7bf4721d4b2c8e4a99b
54126c264a28a5801b11f9d5bbffb2d5
543b7546bcbbe7c99b7fa7f8ef30c0e0
54494734edb9e21c60095627040c2579
54afe6b13bfd53a4d5d4280a59a3ba11
55d6460392408d1325c18b69a91c28e3
56018cdb2a9cc076124385031b2a2580
5689c83e71e5eab65b067e1910a938d6
56aed2990f9b4939304c6a3e86f92883
5700f09933c40244225e5cd14da128c3
5735d0654b36ba6e6bff2eb4234699c9
57e39c48e784b3a979493e1945139cb7
5800963f5a63052d2ff1a52ba6d4f089
59993b705c2e08e57ae2aae2ff3fe6f0
5a3f411d3149c56950e8c9d54f29330e
5b18fe7d9aa3fdcddec0d0932827f7b05
5ba3bd07e3f1f15fdded6b22e7d4a084
5c9b267f543046412151a02f0bd0294c
5cd6d4a5fc1b37d7bd936f5b075831e8
5d651c3d02ee8cc934ba8751b04bf8c4
5d766f42a531ffcd5dc28bb53aa241ba
5d7a89206d72fe7e010df3520bbabcf8
5dd70e90425e2491163052fff62d83de
5e614380d01c2f0f36ee265fe8dd9d9c
5e835a1077f20ea48cbceca3ff7160cd
5f3e43984c7a7d8f6d59b84bd08e95dd
5f76f0b41ac9b298d26f44826b1e4a0d
603fe249c0d10719a8d2fe5985e85d51
6056f18ab8a6b4fce4e2f5c6df55b481
61672650363565ad7ce71c5a261a5e7e
61c5e0d21fa7bb60eb6b7d0a63e65ae5
61d198a66cfb49204b9b7ae0eb1ee918
6200b93ef968a2fadc7541988a58e202
62743a9c1c8921f4d0004c73802b12a5
62e00c3a2ab7efb375892ccf1024a3fd

632ccd3b96acb7f2216f4e132930cde3
63f39ad95738f82aeldfe8a9ab2a7272
63f5b41acd46d5be96eb0da2799dd9cf
6478902b96a9903a68d95257abe82d6a
64b2a30bec89e89714b391908fb8ceff
652477b1b34d67d811ec3498bd029a8b
65a0453f14f1ae36610b2828a9d0c520
66025ac6fd60f45bbeac781a6dd87b0d
66ed7ba0a4ca21ace023480e4015a9e3
671f6fa2476117ebabadfbbabe5a4009
6732d27706bd0aa2fcdba4ec8abdbda7
681a8bd3f1458bfe0806aff4fddf8060
6841ea4e082c9a222e3c1146321cbc72
68641dda684fa0e75812fcd786eb24a9
68efea212241f870e86b9fb2ad495fb5
692dfb735398e58ab046b3a7629ae57d
6948508a9070fe89a4906e42d6cac4e2
6a77739a31832f55efc64257e4234cc6
6ae057e60d3aeaf74fb13a13e1d628f2
6b60e279575d9963d83dfd132fc6b04e
6b7e8537d34f54a025a5433f16284f89
6ba4f0f360a3c3b983ebe85c14a1006e
6d4c80ae0bcc986dbd7439993ae10e54
6e1462cab6301d4912ed2477f5906f1d
6e3c68a999fe77fd8b4f7ed169a5f91b
6ed7df2afc4d1e860bb0afe5f5e1784c
6f703468440a6dfce398a4fd2f5395a7
70089ee71f366b852ab6fdd1f6b57b55
70e90926399154c2708801a73cf53d99
71181a308717c6ada609d17bd8702841
71eb23faff09c04883dd2dd9ecd6bf
7427d4062c9dc701cc6fab313280ec08
75439d645dd11fa76fae0473207c2a38
75495e1a833a4c3268c667496c579421
76751b9fc511861782a48f653e6c88fc
7721437ab39629ba0cafd4a010d4a05
773d31893fe6bb3bb540008841e1d324
776c0ed18a5394a2764c7212b996fd80
779aba07d9e38600f5d56b1cdb4b13b3
78ccda890e42e4ce595f025afecf76a8
78de3f5af5b85325db88a780860b01fb
796280ec19f4cabce99bbb14b4a43f28
79cbccc635c242652e6b1c98fd1cf98f
7ab60b99e9c99846321476ba190cffe5

7ab82dc302d66b8a7d413f2aa57dba4b
7b4bfd9711478d907727eabd3e0e3ce7
7e25d0c57fc2fa19ec71605dd231a1da
7e5fc59ed31009242b69e1eb345a67c1
7ea06cdba97109d8a421ddf4dcd1b264
7ef14b5d4eb2a640ae4bb81500a1fca3
7f2581bb16bc297ff080f153bc5be382
7f546b561e2e0b8490efab53b488c37c
7fb0054fe8ab33a1ae308d8ff5de9acc
805ade542b6b57b9cc0f4fd4ab34881d
80f75e1bf82f40911bd58bf18fcc73e8
839d91ad6b1ad96150fb2c31431ab21b
83d9403ab2294c63a33e8c9af95098fd
848a8ad712afb3844ae1f698c98b29c4
858a2aec27f4bb9bde43c71fcb78d2fd
865ffbbf9721906c2f4802622261d34c
876cdf36f3096235e8a7d2d226939dfd
87f1fa2cbb6d89478f3410e4275ee136
8817e38498b253934ca938b9532bc216
88672904f298bb825c0a4877d6071f9e
88987a164b8da9a0e468965352b76cb8
88db3dc248680191ee02bc6e5e5322d8
893aeb0f5f4438e2748e580c130e100f
895ebaf32f2cf5f84f9cfecf203c9d33
89734ac36ab27e7a09ff736a58b90896
8a8dd3ae4c5731acff6cd279d87702eb
8adef204222f1f04ca7cab99cc19acd0
8b1515b8a1383499a1a50a048d6c8a5b
8bfedcd7d025aa96faa2334b5b85d959
8c9cf2d7436c524449d3559a379a9b7d
8d38003a1a35790a116f039a0998f4bb
8d3fe536c97e1030886d193c74a45150
8d6fc11e4b5dcd985824edf781f93e4c
8d936312a59afa52e18a4d665755690d
8e137d69ea1060ddc427135e612b829f
8eabbd8a98fa5f832cc89aa37fb596ce
8ecc522f8617adaf469f173400806dcf
8f50fe7fca46de669c1d388b253559d5
8fcd14bbb9d091caa0d93fe9e3e132b1
905201355ea21ee452f8149fe9b7aeff
9086933fb695c36c4894adf26d03adbe
9136e197d0dd9d2add2777b641b08ffb
92169aff13bbaec43fcc478a73f8a288
92657911ede136c3113dbded03babc15

92cd84033f1df47f64a22ec82744d9bc
92ec3ce5c18ac906b8105029e1792e14
92f4d58c433103764476f6e0ce7cd516
93374e475d4d1009cb155fd5fa4225d1
93d572f4af087a6ecd3dd36c3fc9a4b2
9416ed7b3bae21726a238a4985c427f6
94a7ae060dd2244f3e523ef87ac573d5
94ca86a154a6353a463c492acac59089
957901808a01c8f38f5d99f2cc8180fe
9667d7a57dddf5103b0dafd3463cc1ef
9711076738ca42d16799cd2acc580fde
97ebc55be1229c2fdec3edd582a4b7f5
986a1af1ab2e09fe17db9c19cf5affab
986d201be0de714e6e89fdb89af8f172
98c9765d987b8639f1a7a3b564baec0d
98f963b9d7225413ec18f48a473c1f40
99306b523906fb0638b692ffc3a64a37
99b61d1223377eb0a459e3d44738ed83
99ccd9dbec4f2fac986712f71123be1a
99ded55f618740a2d0d01d058bbefeb8
9a45a210cd7713382cb3b16d2ef0ab87
9b63265c257167aa077e0155a7e9c301
9c238a69462c10fdc3e363b31fd9f450
9c9543736a245130d2219c23d8e96394
9cebcfb64b5db38123a7fb8a343a0f9e
9d852c3ed188a4bf538deab348763d57
9d97e728e9d190e4be44cd0e2b6af94e
9df5178bc7542526e614fbf1c67b2432
9e0337bc403a707cfc2630e327135bf6
9efe0c0008a1f89d679a3dd8d9726710
9f03bee13b59f586ba74417a8a631b33
9f93361c38dd1f62db504c7a519a8dc6
a07c5b8e7e7c3708ee38f900bb0899f8
a08ae7fbb48f322ba4f74d7b7b7fa072
a0cc3e492c63a5f2a28fcd8c4c8b8224
a192512d560035a6f5d02ec30e15c1f7
a26bb2d5b99a73769caed28b418ac220
a2a688a14c940e969e235eedbfc7551e
a3fbbefc3844b6bda8a3bd3d097dc0f9
a410b1ccc96e6af31f69ebe5b72aef82
a42c2e2b9c34fa278e47f0f7d3fb42f8
a48454c3d614a62de4aab36a70704037
a544a7161b2a3d5a170a21baec510393
a55ae472a9e2695c98028454d6bdf06b

a58b322c74d8dfba9c022607b0d0c26b
a5fe6e3248c27ea0eb59419e2b90a621
a6020934eb354724ef7a78068ef4351e
a60cbdaf4f35894d7970f9bcf606acfa
a69457962eb83af6b4aa00ee27a53e1d
a784cc2e3a8999f38ea116e197be8e54
a953c5f203ec0678d1a2807fdcd2cflc
a96f1d4875fac60c08b50b232af77479
a9ee4f66b0686b6696d2853667da9c18
a9f14f552b7a3b6d62e9b8b58d69f938
ab6cfe6f3cafc3364b092a65c77fcfd
abba6f3a66795534c78e8f4a9240affe
ac506b9b66dfac14f67ca92c679d85bd
ac5eac84181813856e7699c2e12ea78
ad30c2986af94ac48383ec1b1a23dbc3
adf396d13d4eefe67bed074c406c0b9d
ae6ca0aecd53caab392cdf692cf5f284
ae6e5c25ef61b57b0474c12fb74b9880
aeb1becc0f251e643e27c95d2fa1d91b
af3f0a8b0c0568f110f3b1fdcd70ad42
b0421b2db54dcd41c96477f4d7c1b80
b04f4d24b71dddee852e1ac939ad84c7
b0b8c1a3c52c3eebb14478d5bed843e2
b13ca6c287bfc42a8fe4fa28c2ddae87
b162a30b2fb5f93bb0954871eeab8e66
b185ab1a2b4309a489c658d5231c3ce8
b1d3fd78e907eda6633cff65693d6e8d
b2bfa7b9a850a1342886d59a79bc0c7d
b3aee889fea837ae474063ea8c4120b7
b47a8512f2a44782d1e81e5290bc4efd
b5a0914788f8416e6dabdf9b429dc11c
b668249efd71e41a4a4f2f5fa30df2d9
b6dba866f24d460a4aefb0f590138c89
b745d7f6d5c77877ed7d04f4f5f509dc
b81e890d40a94462a1fb21908ec58e42
b8433ddf50e21d90061e85c320af1dda
b8462af1afb6c95d62624002fcc05726
b8ced572dd46dbc9b44067448c98c113
b91ce38e66989e0e84595d24ecac5334
b978636f34d0d34bfd6a64932df1cd2e
ba6ad4c086850609727c94de7a52429c
bb118d0e36716daa4affb6527f0beb74
bbc011ee70c273ddc1c43019979e8446
bc19430f73ce7c06468bb450f99203f2

bd50419823213f1f7b3a8e74b051b379
bd7169be4e1760861e67587f321ab44d
bd8229dde5d5dfd5bded095911b835ca
bddf080aacbc30fe69a24cb8cec532d7
be4e3d43f8de1959a773021ca0a0d390
bea491c7cf33af2f6741fd360e9c421a
beda2ed686ec7984a8ede0840628c540
bf25adfaa34a60cfaa81b6ea040b3540
c05ca56e4619bf0281ffb207d903207a
c12d4e3cbb1366c5c4fdcf8a61186f1
c20ea89a8dd690dd656113e9bb2cc3e8
c24295d54545cb814e9f40080a58b515
c352d3c5652bf8893986c83dfc60bb79
c433d73cc1956df36a509a653c253682
c4575cf4d7ead1f4b89b5b6d811031c7
c477fed6490fdc0406ad07e2c119f73e
c4f1d1f805bb5ad8c123680bb1550aac
c50d76787f65192b2e6eef3bd92464f0
c5278672cc637d4e8ccaa7378b5619b1
c53fe95335557c85a988778b3296b449
c57e9585bbe179e45833e8b896c73e29
c5dade5efee91cab151ee42f1294c860
c5e5b84d5a886418146a3a60007b9d25
c602a8c1d11236ae15cef6d0506019e3
c6158b027d97a2b4c111945566366838
c6915177daf05150797282dd8a897642
c6d36b2f913c4062fb22717eb16d69d4
c6de6c6bf618e381621a009e0677c80c
c9a0be18b88fd741fe6328f4796e6047
c9b1ecfd1ec4a4ebd841f77ee0e74ca0
cb839afd6383cc9dcc5530cfefea00db
cdfbda9ffbe8063dbd24476cc2b5f189
ce1dffef051c2ce170cf5c5a83394021
ce7d806c0c013d7a322071cb5e9d3ab7
ceabc960e9abd4fed9dff3820aba62ca
cf2c2d40ccfc20dc211ea6eeba6de9ad
cf5699f6323c5ba870cdd395d1e19c18
cf7feae0720b3acabf7eb50dcb1b2830
d19b22df7eb03f1be9f2359742386587
d1c1aac83a4eb44e1f5a150b1f9ea01a
d490c17df0e0dfbcfd13c7a0dec8bd68
d4a1e9a56c52defb53402c08964d4569
d4ac7358288cb2dac4363c24eda15267
d5f88e847bdb7f0588c3258fe31758fa

d5ffd38af109fcb529f70d70e789be47
d6527aa70c59daa326b4a7458ab98507
d684ffec7438d50b76869761acf3672a
d6875f02899f08e00588128af54af71c
d714daf1151304f21337ac057f4b1807
d7647797381bfa84ccd377fb5d1a6f34
d794051f2d1b5df0a01bdf176edf7d1d
d7efd600d714d6d2f6b7fad6adec0c9
d87e05169b42accd21ad728f48c03620
d88119055f1234dd4459d7347b045033
da88f31c52b2e7db62ae5840870b2a37
dadd4cd15af7ce8b977eaec8f05693de
db8113681e395345f71223c1d18e40c4
dc20b63fcdcf6e0d3f81c21941e979e7
dc31aa9583c9a6ff67e9e33bb92566d6
dc85ffa9d1a73663274aa89f3eb195e5
dd7662b8324a69979331adf5fd87b86f
de5cc0521eafb16ffe63ca4c08169353
de7f4d762051a07d509d94c0772e718f
deb1dc09d58d089a99792a9cc4bcad4b
dee22604ff5c70b766d186f787dc6bfb
e0f10f49a518510750cbc1aea42d2fc8
e10777c560dcacc0537c588c2999bd7d
e23921fa574147ceffa37f08b5452866
e24cfcdd72282a10aee82ff056a83ad2
e29c36c4346204173e2ad4ad1d265869
e2bfc471bfac9614e01c06143ebda50f
e3c36c506e8451142504893bf0d6346e
e488cd28429957e192b805d58e20c17c
e51c97a1244ef4c82431b05bc414191f
e5c77ec8612a05c11e1667685c03c4e8
e6227bb374a9fd4672a7164bb3234072
e666703d6c7b837f913c6ed5eab6b1c0
e6e41621c6d9eec6d08d5deb8bcded5c
e72d94b2fcbd6cebee1bf76b2552b411
e78a007b4a9befa903f31bc219bcbd75
e823afd1e909b794b2f2602ecadd15cc
e8649aeb6c2d6056f378b0ee5249adeb
e91e2a1f3e7d3bd979d83936b6358392
e93b21f54c96e4ea8947cf386d961f25
ea670d2f2b5b772c59b790bcc65c59f6
ea8507247f690a6432b5099d097b4ad7
ea957fec2bfc2448b11f998cbb72beec
eb0364fd92c4d27b41e49d3918d414b1

eb2edc1021c615f855b9695e29b79433
ec76fcfd17008678b172f32eabe71d7b
eccee50ed45d206ba3c2e75dac5607b8
edfba071e966aad86c057f228f01e723
ee8be0edb19eda90f052a45bb34ad74d
eea7723fba646b22bef4ab429d1013f9
ef732ba436126b21b2cc4132d9399db8
f053457492f033766966011df7763fe5
f12bd5e3f54e9bdc4622d42440a9ea11
f1fcb87af67759fea831366417e2a05
f295a02e44fed1e13cd317fa2a89ddae
f31f472045926140030e8552c6114a97
f3bfb77dd3b498e491bf72693cdfb3e8
f4f863c583635f96b8e673218c1b3127
f5878c095a9c7a286df8834c23ac9083
f5aaecf78a7f8aa64923c3186a4e639f
f5e6ece215da09d55f144585c2f44808
f640aac019a1e4d4c2087787a183ce9e
f683d81b0a64bbd11cf5f21ec12da54c
f7a0e3f3b22d8217ab66885e1cc0d0c1
f93e56f59bcb14ced89778ca311ed51e
f98d644ab8f19948187c6189918dcc1c
fa546e8210840a9a2aa535467dab5c79
fb35b17f84c0ea2a044962324a6373de
fc070bd06536119e66f2fa83b22acc65
fc65f5d9f403d8393f42c8431a42ee7
fcbf2c7ad881fc73845f1b228662bbfe
fcc9d2a0bec3ec897e429b62e9da8ebf
fcd8c31d4cddc5ce08f1cc04b1588732
fd1a0c0792c6e8e6c1a2c4b2c1585232
fd4a1db355d38e4ea9152b5633c7e5d0
fdd48c097333633bf3c926a55fdf6b9b
fe1aa7fa995970ebb34465d5dc0d8ce1
fea929b9b964865249f8d39f8b432a53
ffb5e59dbd2ddf6e8f0b3d3c56293cff
ffec9f3d109cd570f9a2962adc6f14c1
fff4cf5e0199ff11b091ba9fc52ec041

SHA1
00165449e24c5ad2b30070328d0d6f884097df34
012f2ddf5f9f049fb2f2b77133ada93b6953f908
0141d58dc4327faeb76da5152c2d83003c2dd90c
0254e7859a88131131fafd7c583637108b62b6fd

033bfca36ec34b5d71ae8c071fd1ac09cfcc2087
03cf0320be7b2384e17d7bbd417fd826f8a1c22c
0451ea33f7280ef253046bc142d9475737ed6841
04bdd20c87e9b5b7fe6aebb06dc29fbe43645c3c
04c04fb395f15820a5b22b6a5d0b377d04d95f98
0525367476774b82fc2aab18d647d43fa6e6fa43
061c12be6433a1c9d0a933865b6da611df637efd
06622045c5f36ddb7e78a856e713781e6a71aca
0705596d2595a22dab5d717cc4b114b5cb509f23
0766b849444649e692bc31dde2afd441f5f3fc54
0788f3436168e51dccde2e6fcf3b73490b3923b3
094eff2ab11ae19699d7dc5c22bab0429beb50bb
0995ac8e8e94d63d3a206766857cc89ed6aef80c
09f4a1032067ec2c4c12206b67f4348fa8fc8432
0a93f1db6f7b10fc39d816358c179b8194ddfe27
0dc4a3ba895028f86382c60ee49ac8460027a4af
0e57da428669db68f70ffadf869339db534716f6
0eaff8f1cde17a392d9e7935bae96f21c91acc3c
0f8ec82770d645ad13b5804f2c44f600c76c5655
103b54bde0b5a996cd3c3713cd779c5b8c3540b4
104e5d48286576742b08a760d208ce13ad141594
1232483ee0d3d6a4ece0237247d9cd5b9c36dcaa
123602af0b3f78c94bc16348f9e9e80005086355
1272577d90b8d212416732e54258b136cbd2f3d3
127b2a6444a323070ec31643dabfa3803ffe2ec5
12bd9e27c6f82790176859474e80c344a215d363
131af9283a990d7dd801609b81c6e1641696bb6a
1342f274db0c3ff397d466e69c84e8fbcfe7f72f
13b5ebf1ed285a2261175be26307b39ac0691c0e
1456c28cb5a86c2c9273200270070a1a017a862d
1581e588673fc0b1c956a277cb0a9d0608058df1
15b2e70e408341efa978ac7bb6bb190e76516359
16b48d262908d6493348c716664c7ed3d2e6579f
17d20dee353c474436c58be66e5399eb1b5457a3
18248a16b4d521ccd038c57853ee89a11c3c654a
1893ede2aac26c7ffa36e0165d6a1718be87e3e6
195c67fc80582c27d9bf2ddf262f48be6f8d730f
1970a98a7836897dabfd4d7ac3b559d3654d4753
19aff3a1b435064dade4cc095f9d10a5b6ba9859
1a60cb5d026f29a1009dbf2287ae6bb70f663d73
1abfeb24edd346228e56ef94bf5241f2a39b8a4f
1b26eec6b6a43f14e1d910efb1f81a0caa870624
1c0aa52a527f919f0e575e00dcb7023c553f9d5c
1e459f7e6d31e28dd43c6d49509c70ab765ae5a0

1e82110038724b016595d2b334829502f32f0cac
1ee3b44562f12d7236ad8b635f282532a7586e7d
1fb49306b8b02b52887fa4c924656cd22c4edcb6
1fee23932ed23988176ed1b594af4a50aebf2dbf
203e61d0e28a1be834cfdb01a99f5f775dbad0d9
20673c50aa2bf55ea2a9f0871acd0bb6f600f66b
21674862aa715cc41d750b2bc6138de2034620b1
216ad9c28a00b035f173c29340549c7b0f359c4a
223c08fa084d4334e3f2fb5fd91a097c0b1d1476
2296db4bde732ee09af6e61b6796835eca3f22fd
23a1efd07c8cd8f9522b7f87b12acd352984a137
23ff0083ebb6f35ab2614e91338f401fd6c44a43
24f2edb6cbf12276e97ba83106de5739a195e96a
24f732da2cb5892203b4daebe79d7572e137d50a
25fc95e0df884986e87d608c6879086b2cbec08f
2605ad8396b5ea90c8a371fdb76c58b12931d66c
262e6cb688ad062bdde2b6702871a5a07e4409f4
2652c7c6f5a45fec4bcb14b35346507f2f55792b
27981aa015340cf83cac3ac632382b8fb10cbc49
28243ece16917acc27fea562f6ea6eaf7d535e55
28c9d7176ead66aa9aa4e9889d49e4576405bebf
29b4d89b1e3e0112fc25e53973cda2ab7cfd373c
29eb3a1c687f8116af363a021620ca48662402b5
2aacc31c06a2b025b6cf755c6bae0a385568398
2b3a15671f065e877aeab9b2a7d3998c16e02ef5
2b6074e7b2c27eba3afa1da95b6283b84d7c9f00
2b9e4c4d82fcb91ec317ba1ea94a43c99c1a88f3
2bc9bcd8e9f868796270359a11d4786e5e2832ee
2d27e899704d74a88696c402f638f83d6b733f3e
2d9d42ea7ca927b8c14f8c4165f4ee6b7dab96b3
2dee4d24b279d2abee5d2aa0a89674cfe4968bae
2ec5834deffa3e4bc9fc82429940e43db9066cb5
2fab44d080175e14768c7433e06dfb4480bdd3c6
30053cd2f009804c297b1943ca10747f37f179ee
303518f8ee040aea50f9f9958294e315aec6cf77
3074d590c3ffdf65c0516cf18d816da9978cb321
3236d7bfff0c43c0fd40d68945706fa1eeb2ac581
32d34fe7f7e6f707767c9496e549ad2f3357c21a
33207d35ffe3bc6b19ffc016b06432c43961e791
338d96c4ce8e0ee66b898a78c00c53bdc8e31599
339bdb04108cc4caa405afafe26581a104c3616a
33eafebd419ccfb6f3c1d9f4d93521c5a0bb18f5
33f8bad6e602922a890319168d96bc1bb94ae4c8
3551c6ddbe76927927a22262af48c225a9c28d6b

362c1af3d3f43b4185e017b9a20f1bf73d0c2ab4
362effb467e507db186bf21f7481bfa23392dc0f
37dcd5092b3e4d7c198e1a33d05e90c453bc1cf
380f438ad2620f1582fe9b7d5b2ba84c3d6f3349
3985d216f58d59eade8f9760941addceaf87d6bd
398bf16effefabbb813377493e1c67410d042d7e
3a6517ca768cb7c868f5de68230f9f5b2736da9c
3a71d0fc2ece9081cf433981d90665e82b552a17
3b4c0a2ccda11a35a42d67c7821db3b8bd9ceb3d
3b8ad06147226f50ea3736592c553d4824dcc3b0
3c6c37db8fbf189adb1fcebc8a52e70358a9b25
3cef598ac69d88e736998d9f082abeaa04be7c91
3d5a0adf745b6b93e3eb631eea3dd8c4f604acf8
3daf473c34d4819067cec204e22160d8054d6eb9
3e0e91d05158eb8f7393925a0659e38d4b4b45ec
3e4b3e35b7bc38d373e3cfbddd86e32bc38a119
405847d03be406a0025eda76852dfd46420a8d7a
421682470a987e8494007b97abbde9a1b4dc9541
42af804bfd7eb0b2b5c5403ace3d8b7278f28658
43460dbeac565dfa00d524bbd083cef5fdf64371
43b57de4fef7246cc298de4001b2da4e5054a48e
43b5c3f368a74ff683a12e891f5866125af7cc8e
4428ff2dca82a79453ac64af960d94905f008594
4429e74bbcc0481dda5dfe4945b2cef4e78b84166
444ec66e38b93028041e17f6f951d03564c6b230
448752fcc5e625883d6896abc8fb46d655f28244
44cd083830f8b5c6d0e05acc31cb66536c4049ac
47705b3f884be98a207e4d7695d181caa792efc5
4778d360087f2caf3462945ec5c77a96429c70c0
48f229afb963a5bfaf52d456265b9e7776a86ff3
4907871ee88fd8aeef900bc54fd26deb0049db63
4943bf54985728acb24d81536bac9d15dbac7bca
495d40d17e233d13212397b07a17c86f8a0bfb7e
497b33585436121ca8da7c1fc0ddca7fb054d54a
4a2df41def7a45e1e5568f95efb90f4e624c560b
4ab5de3c9df69b32bdb2c7b39fa36ab5f14ec97a
4b770a99ec1cab529345d8ddb181a0aa5b8e1689
4bfef5be1eeb772e7a9e1003765678c5c9613ba7
4c6d24c796faebc2b54f981e2f8127187b08aa4b
4ccebe23ab6c634861a82e24ee19db60499154ff
4d20edbdb8cfef8d035152d9205a2b8a6b31d1fb
4ddf3aa471f22d926c2a89ae1323a18de64001b9
4e12d1c98e419ca17dfdf6cebf1d820c253ed245
4e6d5ef96bdec0a35a88847911c623ec76bac8d3

4e9d18d5c9f1d8518483ffc3eac24bfc2dae26e2
4fb2d144a4941828df72cdc76617ce5b38d3229c
4fbeadd88966d6f66a84eb33cecdbeafa7b446f52
4ff5776425fa957e56937dcaecbf94ebec98f8cf
50b145b70fb04530132dee8b0dc0eec980744001
50cde747841e64bd77a3d10493d8448691e4ea25
5106e7b89f19888cc5c5e6e61d32518e9c654729
5165a504e35ac83fad1e5915c96acf7e7351c692
5173feb0360d628a1fd33e00b8860cfe576dff3a
519ff74cb97229a1b1fd7c5515b165cf30472e28
51bc3becc236098b9fee553a6f811dab513f585a
51c77914a98383469de313ca66a1f085e26eb7eb
528a4a25bafbc76d05e946b3769d322d82967668
534a799c083a8e4cbddd91aaffc67c1b640832c7
537bedfbb6f6f2b544bc4892437f4a6050d932a9
54972afe738a2255c2f319b3077a6413a812809b
54d8a155fbf50720ec99817a4e5dee29d021a0ed
554daaf07aa9f44e8105588bda2ec08ffda1f820
55b19f353d02e9f00ca1ec22140466fb3ae88d22
567601b9520f05e4dec7559212af1fd863e328a0
57ec3ef06f64636004f18fccfb87e527b9a1e28c
58586704ac109833b18aa30118cb7e44964f73dc
58600e413e73adflbb708318dbb24ffe69c05f6b
587ebc690239bb64b910469515f1ce3639b3f9e7
58e964eec035d120bfdd5c8968f227fc516f8893
5accb3182865a75312a64e87431278f1072b81ca
5ba7b078b5c4f83582f4b5fab738d2bc40b3caac
5d1be7d1288181497dbea5d0ffb6aca65c1d0b62
5d6cbe48a8c913d4dfc431e893a40b4391382648
5da779c51ba99bdd6d116aa07ca85d16ee1a857a
5db8587135cae319f63d7e7ddfa9229e9501a6c9
5de079ca99452414f7c68a985e9b114a16bf3f94
5e34dfaa2aa5f699e1fc08908a76e89245fba535
5e6443132bb31fff16d94f823bc3df467de8dc84
5ebb33c6381ed5f23fab03b5b1bba0456c2af2e
5ece993b14c1381b0d2457e9cbfae097042a416f
5f0462ee1ba3888fe5315f03403d6e5d5f0579a8
617ab388348ce4fa1bd203caa5558e570467db11
61ba043a59c49a063f036342db18e595f2e1034a
61e6ffda193221c646528fe23b6934b35967755f
62193cd710f02310baf7860ed57c6336b16bd371
623ce4d8e53ab454e36c9260bfed2578196602ce
63158f1b89d6c76e293b878941e28f85521d6e4c
63691f614618e101c120cb83b21fb56e22301960

638e6c4c773fed8743b7cc3aaa299e5bf9f465df
63a8297ce7605be8442de588f8f51cc384233a54
63ba8d686084769ed11b63a3f02e73990684a895
63dc212653c872d98ae01387b272e753322673ad
63fd1c86b6173b300bd4f7515841279e51abecfa
646df1c5b6d45e01e1e042ffe9e46f820ecc5b4c
64d47415cfb64fd5657705263a4d3041f2ac2d7e
666c5b23521c1491adeeee26716a1794b09080ec
667706a5f1592dd98199e700b245be078032c6f5
6683d3616d79c92313fb8894900040fb9411ccb2
66f71f1f920899aaffe32f996f25df8c031cbd8a
676325f5090ffa97be8232401f43f29d3edb1fbf
67ca4066cd527edc67c4d690d49c1a5eacd8119d
6859782d7778af2c26fb55ae08e68948c5bdbf00
68bac1d99f2937acf8e588f9b52c5ba54cc04de9
68cc9aeb577bdc4913a5d7f6c7c693ea60481685
69fca5f5e52ab261dab32f926a197593fed56b37
6a5862283d1c7f3430cede20d7339e542333b4ab
6ac5ccc5866a77318e7a317b7b81f559a34b95d8
6b0d2560f4b3a354c8916604f9bebada131555d5
6b20680dc0f178e3b47feed411b13f5d15d65e49
6b5045282bfb648754bbd3d40706fde1ee6dd7c3
6b9eb66cedd25665627c095b40d41913447ca60b
6bb8c94df04c1713f12b4f2ca2d141a013881f8f
6c4ec8743ac76cd05fc33ba3d05e82c47d87ecce
6d16e1f6412532e51d75cadaa8fa69eee4292574
6d654504ba1ffd82d338634995bfe375e3ed8cae
6da1cf7a0e4a708e6c986a810cbe87ac73bbf5e1
6da2e6f0811c087bcc7c5af3a60ca2a65cf4174f
6dafbf05ef562ae8cbe4816e47e51ad5ecf9ac0d
6e034e99faa9847aff1a1597e71e0e0658099ff1
6e23253bab8fa75819371c8f35128e431e276f34
6e76abbb389df860dc543a61ab28d42131f93da2
6e827965879b0a574573e380cb9087f7c111a224
6f54a2734123407e84473c97f489f6de88d66dca
6f8e7149a12e8f21e469cceb9ec7c59e6df334a6
6ff17d18aec649b6a584e9df80f27df8715e299e
70019eb810bb92099a510ce4a854578f81b7d3be
70205301521da7fed3ba519649d07dcffa4469d7
708d8c5b6b173a67e2b54ce7e3a4e1a0293939cd
7159178162a6153620b71e732a795956ed3a7b67
71dc1d4c776a01b2b348a0c0159dcd2d1d1de599
732c300ac93704f194ac403009ffb9034f5e1957
73a953bf41fd21731e2d8fd9d4da3dd4a2951f6c

73df34bd2bdd4a00476f5cfe7da4bed436d60843
74c96fc0fdda4654cd9cb482a136e72bc5e0d8a3
7505b261cc9df8c6ab8f10e035cf8d8319043cdb
751dc4baf8c4ae5b219807935349ece9a322d307
7525a8ec3403695d4b54c6fe392c073c9f49e4d1
76dff305a3093b266e41bf5b6e959e69f5846808
7781220fc1457690f6f8ea102200d4628898a0b6
77ac70070b042f511c168693426cafab6fd9581
77c102d3dad8675d05666f0738e9afe251ef23a3
7827209344a47a8365c01e9189b418a4a44727da
7938ea46e93d1f7d80e84687955fc2894f6fb051
797ca633afbbd217f34dae515415665da2336a04
79a6ab1b6d5844187d57c99e234a1876087411f6
7adcef228867d6375bb5766f6eeba76c698622e1
7b5e72f10cb6c61f6cc3deb205d86c866055ffa0
7ba2fc17c47159045f52ca317c581f2a2b79939f
7c8f17495f3eed7ba37cb215870b3a43ee2820c2
7d1337b844f5593e49c8b37c4e30117c1d8529dd
7de05fd3a996f5d804c25e8ce9f9721c56c86b48
7e089ea2820ae01e64112c7a038f42300ebe8758
7e1c32a8e6072d4a647fc598b8efd0a8ad2b3f17
7e3ee5f27c43041f020597fc66b1ababf9b0126e
7e9c812074643cf04e33ee3be47d0aa0049a27ed
80286ab408cc08343d073d440e2024627b65d327
81123d820807007a1d7ebf0286f02c0120f5cdc4
8113ba05ee0017ae5a8e3075f57853b1726b89b1
812ba37bf384e0bea85097441ccd024f683fa1c6
8249785351dcb12df7f2c277f509dd8807f577d
82c54ef8840018043be63da501005dbffa3676a9
8338d7a2616a04e059d229867874e0d96dc6bb16
83af4612bbcc7f72aa094847b1eb1680d5d95926
83b805ee492790f99a2dd5b51f3bcf3b0fe41482
84d71968b0c177b37be5661a60ddc03af01bf820
85cc28034f1d33bb4c557c9fcf7c8e92cfe4f8b7
864c7e96cb4f82f63219fe6374af68271a654ed8
86ad9a97fc25ff6f2d6e12a9dcc59868b2f4fb21
875e6cd10c8a9f1d2440a6b69d38eaa9735cd46c
88b6741fa7441402de46359ba3e0b6e621f25687
88c9ee7dc5ff6c22fc94a1dc89afae81ff26fa24
8b23bb90ec0c72cabe6969e703386e2368c80eb6
8b59d545abbb0113dba8a7a97a2fa236888a32b8
8c8454f39c2028be1d52e9c229f22f0655d71018
8ce07d6c03f81c6e9447c3dab8ea5e568b92de38
8d6cbb4ac71da3b45fa308f4943bf5ae20476ab1

8d73ef70edf9b4c8140d6b80708ea69030c5d242
8dac1f87c7129b04a8ea1f06cf44ed0f51b7bbbe
8df618838b77273d6997249be56d36454bd40cc4
8edccec93f1f626b72794d41308dc019d79c361b
8f10196462c460575b7907198b0efccaa57fe184
913e2758bf5eab1596efe219c7eadf580cda3d1
9140352be53761f871f45af33fcb7b1c6c84be54
920c0530a34d319ad8859238603f52a6a0c2ed63
92269b2ad71e7cac4eff7dc810f2989b93ac74b0
9233491ccf82cccdccbae1a2ce48aece3973b01
927325caa14d2c8c02f66ac9379c437b2b2f8b39
930d149ca9e4932216c7489561e126267bb80380
935f6acc98fecc12303f754280d8251244bf9edb
944ca846cac6d90275a104d1467881efdae28e46
94cd4f4e3626598be807eb68eaaa9caade3fb021
95f4ffc07db4709c9d148273320cee228fbf088a
96fadafbelce02f4841f114ca10a0be5cd7dc31
9773d971973bc9c1ae0c43c2e2ff1f88784dde17
978dbdd1de6938658fd2bb7fa62504bc9854e7df
98874c8c11693d1264bac5e0ad9e84086f273bbf
993e3f753f91d34cf47577de756faff550c57ea9
9991fe427d8b8fd0e6475f1c4381efa2914ce5eb
99ef151d996ed3f09e66891ae304a2f36299dd39
9a3f50bbe8493e1001aa400c915c9ab51c4150a7
9a5e79a01df75bcb04705ababdd062c2074486e2
9bd15b86885b9e714492a97cbe84ddfa6451991
9c3a70adb046f462d2e347d00e6a3e11a1e46cb4
9c764db3dccde0a69d915cdb1e1d8041c2353f8e
9cde38aee96e945ab060de31ca18bf90620c135f
9ceb1d194fab5c600401b82213ca74974fced3aa
9e09ef7b618d841ba4fc0cc17d0a76a24acde115
9e56421f5fc412caa2c598f701dd0198b4c87c33
9f348fea62af656c8101f3eb09a00ed8e47c1faa
9f4ddecf735aaff059c43978427507e8fb5d9e61
9ff40194659288c71ee7ff01435eac29d5d55004
a014fb99b275a15dd5b0908f218362fc783b4794
a16442a7163759bdcbe32d163ed6e45549e7eb12
a1b8959fb7b4d32a3adae8071f3a1bb581879737
a1e30138ac919375cc3bef6f3fac6cad9a3cee0c
a2ba66dc06d416d2e6b60abc2e307f35e11d0e33
a3c698616b7336263fabda2ffcaa504776b3b0c1
a3d8c88b5b9709699c183925ce3ad653491aee9d
a54121f29218dd822503020efae91c9cdfb14643
a5a009bbc993772c28ef918ff313fc565e40f936

a5d80bc3d0055639e13a9a13166eeea1e95b6981
a600c73bda1a5d28e68e707e94a380d3623f0af3
a64bea4be2c84112e3af6a19010dcc9be0a88161
a657a491aadf6c409448e7f76a2a2ec172288b41
a693fbefa05c33c5f377941df6b55090e907e6e6
a6abd8fdfaaf1c7722d6b8ca39161624aa7c2c98
a6c8cf5366e5715cb6ea2cf7f566be7d94737655
a6d28626b6199d5e74cd403b601971bdc94c31c8
a7062897a84533c30705eb6667d352c78a43b9f6
a73f82eeef1422bf012ef7af853411a60bccb1e4
a9b921a70ec4b464b7ebf36dfe3b127e98723bf3
a9e5af8b2fd2fac5cd781a8d56c66551043cf8db
aa34c27cdf073eb5c54e371714d3d68a626bbbd8
ab3611621e581cca9c102489f5843d91c68dff49
abfc539e39c05d2fecbbd37028ff8991c01e7c6f
aca2cb25b18935700a2813f29ec56314ca8ac781
ad17a95b6a56377f8b4c663f31da968864f46ad9
ad928912a1ce4948cf10139adad801bb4d1028b3
adbd1d468e165bed2d3828b62899ef753d1d4723
aed2802d4e4bd8a5bceb9626120366c0dc7f3715
af18994c3f0e7aadc73e7467b8a741b06b5013d5
b063fc7d85cef9ed216f0341938bf5f2141c2d3
b0e20184472df0a75f097fb8d489eb15131b2ee3
b2416a62a12f92a6de666d45642a125ae024fd66
b38a7723ad392573af26ac67b3dcf5844debff54
b3ff3299abc99595b9cce72d3caa8f680f03980e
b424454a849f041d9b22c418138ac925eb92115f
b473556f8a96894a46aa3593f41ad3594fc40c3b
b4b26d9674252eb6dcdafc00135761c142a8f210
b5e0ed3c9c2096e8b032178430772e1071bfb7fc
b604eb4789bd591c940b4e193dc2f543a55cc422
b63a5067eb88da9f0d5cdd62d82969f5e0d2e5f6
b6710870db3b3c8a2b6dc8fe7cb2011fdad33097
b701c0a30f8ccc425bdabf3962d208796bb998e7
b724b65b6bb16cb956ffa693e3b66da99696ef3f
b7e89456fdf93efb3211d83a7ee4654bf9056bec
b84c05bfaaf6e98566479d6a990eaa79baf95b48
b93dcd8276da9c177e42749d84d8c67cc627d45b
ba35f96a741637af1d1319074396f1bf891f13e1
bb39da67bac5c77002c3d88e80888786fcd83acd
bb6a32c77ceb6d5b2c32731b0109c177c7aa6d96
bbc388e10c1240e469b2a16d74f49dc6305e30d7
bc171b3c828566a705e82dce3a96f78e836c2dd1
bc2a58c20b63f302c0c053711074ae80c9af2e82

be3166a203ccbf1ce56ee05420a1972f115d8620
be554a56d24cfe2b204fd9380011fb7c0dbe12c7
be5f4745eeba4115a8104331347bae348e595e9a
bea49b007df1ccfe21be8e3991cd6f0ab8a26d2a
becd8d9257efa5df71f3a49edb5ea07da0511755
bf23ff01dee72f856ae74b67b64f7373f9917a59
bf2ac1b26edb83492c37e6fe43d779a1f3ca2555
bfaefa468f0091cee44e53b1d20b5d3af0d25ab0
c06a2da7c13bcebe40b4cd46d4afaaa856b0de95
c0b243c6f8bde7f51f794fc1b1eb12c327914969
c0ee7bf269244aa284131b3d3d09d8bbdc68f2d6
c118269debaa79341968895fe21080d5faacdc57
c18b50db3c100d0adbc7a60530e515eb35483389
c1d808028449413e1868f376a08f3929858bd058
c1fdb6b6829afe4f2bb9ddac12cb384bb2dff286
c2241c79e46edaa60fdc4afdcfbecad8cf7b57d0
c2911934ece108745f314ca7f9e763b7986001a2
c33773d28c2766f471b33fea9aed3fe8bcf53faf
c3377d53a1f92949125a3c8eace5227fc68561b4
c36442a4da6b0b67e5855fa503b599e07028d31e
c5780f4ec843783b93635852aab5839aaf9a525d
c5a5f873dd13f35d1ffbad46a7eb307dd72559ec
c6a6f832b9f2b7b7ecd3c0f3501a4935b63e72b8
c6ea5620895f47353cfc84fe5302bd9ada97e20f
c7ae7f220e36bd058e623d2000ca29c8ab9698e3
c8a4ff3de64e8caf83b71ccadfe9da5697737c35
c8bd79e8ae3f84d0bedc8c5c832209c95a73fa74
c8f9c31c70aa6a20dc7e20a909e23c2672af9014
c907fd3db66a9ea9dfe45dc9fe3c706a18b4b3d9
c96857cc2d0177ec028bba3e467f3930e5c02f85
ca78f3dd3c28177446235146b60d94855ef604a1
ca9f9f95b42dd59cb9f6025b2b2bca16519fc00f
cab1aa95fc92f9db3a6ca858c3bf2c3fd6f3b955
cb07a374e9ae7a1c1709f408f64275aa098d9ccb
cb29af3a814e40a809528f10076152ea2f450651
cd8242152d2c51192a8ae008bfde01c2d25b7e3f
cdcdf3dfd729da8ed2a7e58b641b0326f48823a8
ce223acd7bc0c3018b0b8aa48a5a25ea29a51d3f
ce437459e1c65dff1c56b9ec646f6aca42bffdd3
ceaacb7b3e131e7b24c7a5eeb5963fd22718088e
cf35d3c0bf02cdf7abd8528bfea7cb8175e8cb7b
cf6f49e4580367bacdc8b0ad2c7d156456c5ea82
d0961a86e4b5c9581dc51d0e9d8cbafd8c7574f2
d0b980c107e2e20f9494b7fa534366db2a7e9a78

d1570beefc64a9182697b31bcb7ab93cb7892b85
d1fe454b264a8508dec0f16563df6b8662236f71
d2064e03c2d638130e90ec3b8321d6816f6927d8
d23aabcb567c266ab18903f6db21e0cb7a8f0fcd
d25600f109fb49d70267c40bb99105f01f92805b
d288372750a8b1e1c8f46fb609cc5fc152cabe29
d30470ea2080c14b74e1c0d2758d2849ab2127fc
d32947941833558ec0d5d1b2e83ca853b9500363
d47ee6cc0ae6fa46caac87d5cfc3945aa4397367
d4b93ad9205f9164ba5804691060006ac60d7c2c
d59d6cdd0424293052e4b09c782074004bbbcd1c
d5a87abd0c88a67940df6e038a680e181153ba5a
d5eda70f6761009e63e4d047e76f894c0d39d696
d742bd8e3c27cb28167bd2e35581b6ffa43506a7
d74e7e2375fbee23fb927961cdcfc889666a387c
d7b50f4057a56fc66f8e84a857ff59657add1929
d909880c41590c25dc2b6fba7636ec7531bb7c6f
d92ec09b5f8aa5d354bf473372d64e0e31168596
d985438163ca837a7e3ea211749427087230eff3
d994820625144db20ad0ff08abf3ede71b6232fa
da572bd099a7460175eedf87740307f17ece548c
da70e0ed691217615c57963c58e18de927c13294
da8d170d46db6321fdc8f8daef88bb6e24e471b
db69a715436bcd3dd06770b9c31ce5d8176d90c0
dc3af52603d81f7605b1b42fa27db11b9876addb
dcaf23172dbfa5cb2b84cd4081c66ac7929111cc
dd860682cb09d762e0726343001dde6b85e91305
dda56b8b97574caa956abb1eb291c4cafda92ba3
ddaadcc914946c21ddfb9d0b52e964fa048ed1be
ddf2ae38a918cf2ff8ab523376faaf439f923379
de7d610c08a079869d4f8828771678016b2d99e2
df51c505b57eb61c80aced5cdf45655dade5f37e
df8564e5f043a0d3e321da69a0bf54392a52fcf6
e173a6d224a97430b89bd06df5b2a4fb50b17a30
e181789066bdaff32544ffb454761ce7af3577db
e21d9b711a7b2842049fce3c5ddcc8d8551a122d
e249b9f9757a852c981943376cb1b1067170f6a6
e29acce554e691a9de0d254746783da2053209d1
e2ebe742bae87efca8021c301af89c7f8d180c13
e30216a7c291213decebd903d3e8d9c8d61fa06c
e3657efa4d1a4e8a0889c083d6c2abc511adc0e2
e39a4d7ff0cb5a1d09d80d8d5828557a001f2944
e3e3f9f4b3101566874a1c90f67b5fa860107ebe
e44bae4b5c71c79a11c22fe97d45bb027822b4cd

e5055d4b1909f9366149dc0870e301bb665c1d47
e56f8d032fad1e77ddee46c8cc912d77ab4884a4
e57f2fd79379929c5445d2b33f3553e4a0971248
e7539267a67be7677d78f6e87bf6b0e9f3707acf
e84de5b78472f2c7672e93c8fd01be716bc089af
e86acea06a600f170402a0c1020c25ac2550ffa0
e9c2a9aac287d9ea24ea74f0a63de9e8cfcddb9
ea2b41ba0965aa7ec1431e95ba5cfa0e425bab8b
ea972d2b7dd60fa2886e1edc3e30c584c0d2f84
each35c401ddeb85fda32afbde69f80c6ef0d23c
eadfde48ac259605190da64fb577314e744f7e40
eb5540a11a3c83a5bdf779f7cfd5c21a0f5121e
eb8643711493829c1b8bbcb454aba86757ba8b27
ebbd388008a94ec8a20724ea4f6ee48bfc93e238
ebefe0cadb4c15fdf3e989b7e7ffb2aa5884dc40
ebf47fcf73dccc29e73aa048d3aed174a622b431
ec26ebe03fe48704736124fca926f969d63d685
ece05a0ce3cb7b1524e607121525dce53d5ef362
ed730734cec4225d60cc83a1cc4c4449f9bb7329
ed745b72c682d4b74dec09af7bce32d7acebab4c
ee47e681f71f612885d84a72e4cd7ef7dca86606
ef97665d6d8a2ca89d5ab8b621e059e9d1fc908f
f0c9f72f87df577de95ce820b34e227dad189ed2
f12489eed73f78bf54941c0058f1e23632baa77f
f14bbcb296f1da0a6e11993286871f2e9bacff72c
f14cd79d94d5edf2f9b006689f468baa25e971c4
f167154cbd6d504463fe11f89d1f49bacb77db05
f27a20e08c66eccf3f230c4b3cefe824b8d08a9a
f283c20a4fa1544d705a1dbedccef160fe381fe
f28fee2fea1ff815d079feb176daadd5fe00c28c
f38936225c35843224695ed18cd96b1caf205e6a
f397c1323a7ec12f5b4c76c15dcaba06907b7313
f3c692ac88342c6439c6391a880216b014d15ed4
f45b94c1568625c4b3507ea6c37c70a35344cad7
f4ff8f39be7d20b659226f38b318bd8ef626c198
f5a5891e90f8dfe19405287420a0a7506d139494
f7036f2af804553eaa65470b6534f46af8ee2b59
f72afed49cb74640936ec79108c6dd4781b42119
f748569d8abc4f63ba2e13e61221b6f539340f13
f7d3cfeef02d4f1f03c9f83749f27b724eec6c19
f874fcd17517e88bec44350a17309ec26b58d8fc
f92ecb4827c429295fc5a948202c0465804fe3e6
f968cd7230b6d37bccfdc3d958ec871604a4bb0d
fa3d67066ad84410aeb1618c7f753f1a2896390d

faad12efbc04532ad6997abd6cd47b8beb4e0dbb
faba8b900937d79a1a0f913cfbc62078f2bd3928
fb3ae6f78e211c6219f618b4c607b1916c5921fd
fbd38393b171712b0b944e512f92e9778d884f10
fc09b42732f4882bc43845aa16448db259db2820
fc19b79c0090de13ec2632901b2d15f9f7d0b746
fc5b74e5a6e1e4289532c6501ed9b22811528072
fcfe250f21f7cdbebe62b00decc25eb38ce283fb
fd74adbc0716f2039cd16eac2f6e91f968c0973
fe4a22af06fb8d122065dfb0683a7dd9c6baae9a
feb3c0ced829b8ab62d94267f4ec0578ede5e58a
fffdfa210ad415d829e14b4ef81edd5b5f509520

SHA256
0032f7e4b264ebd2285cc52705e82dd93caed258069f3c32b341d7c1d7948397
012ab2c73b3c2d4b4c0d3c9cb89bcac0e8de1253370ce2ed535d019dba83dd06
0154435a9ca1ab793fb98c95b9fc1cef9b64cb6e54e7b0af6cc358018476ee11
02a2a44548d03be7cbc7276014f36ad14b6238694864f0ba481b2303ee387cfb
030140bb2a243fdc177478bdfa34f7f388103a349ea0de737925e69def4f4e68
038e75fa865a3c80bb63a5fd0ddaf57b882d4c4e6ef96cb10b58113234747663
042c9a8e5b4ac0ec1f2a0ee0c9fa0035ee54379eba96ac1293272bc30c991651
050f3d23f59375261c93f01e73d08cf2afb9d84363aac0289525dbb17049d7ba
056fefb924727f883dacfa1b86cf30fd3f62c5802d4ab3c6a0ae3493e9aaab98
05740669c4c15fd382ace1ce3a03d78ad0bbd9b1dafa5d22d13fea990e07e65b
06cf43a592776ad9e34f1efc2cddf859599710b25f395d871b5a5e13d88b30d8
06f53f01ead589bb90b3f54518d5084d659ca3b5aab9a27c3160fafefbf3622ed
0722b11416e0594b0a98890379724ab43c9830afa040920f8a97fda815b6121a
08a87793c7ca10af688ef68cf54f4e5a632bef11145a60c6e48027ca91c386a5
090bda30655e11ac22679db9e9470c9b4b5af8bd9a53eca5e6409484922f7920
0930e19ef2b03366535482588c1756a7a4d759274e91d591f13ad397d9ed8df4
09f3a3ec989361e622f1ac9b42bc380846518c270dac7783e6e38aa1f12ccee4
0a87339cba74896d097ec10a18315d66dc3f98121ba968ff571a29e241646b60
0a945bb9d22fe4330c70ef916aeaf9a375661c758a8ae6e60effbd725b95baf0
0b4359771fb67318c3cdc36e1c33b202aec9595d1a66bdd214ca54f44f478906
0ca92ca94e8cee6d8867a9013fbb90ec9a322b39300b189e49cacbec15307b38
0d65bd3f562fa127be5f009203fed5b0da090648f61d10d03ded5c89228e3766
0d68d580e0b5eab78723411ce4a8a0cca360d1495951dde7243989044d47d5cc
0dd53c57c72aea305d1b83fa26283f21caaa5f8263f41d84a43884fb2eb62ce3
0e73ec157e7a0a55472934856ccafe17fe94b863172fa080ef3144ed99369c45
0f271e19f44c1a2535e2010c6c9d25cacfba120bd75fab85e01feebe961dd4c7
0f364d13694b059d313ad904f4d3ed461abf894d846c3d358f8ca2b99e50ba24
0f371fd55d2f262b3e904d6249299f9b259b37e7f1d6b972c65d258390fe198e
0fd42f3888582345dc0cd69736a925162733b9cb543f1ab1ca55fb80c2eecb8b

0fdebbaa78b24f881ace26131d5b968594fbc9abf918e7d38ac2dbaf1fd2eec2
105f447bcfe0bdb2eabd39ed8dc020b572403a57d73fa12d8d43970d0cc3e994
1194e9b5bdfb9fe0a522ef82e32c1bafdd8c5a577005de308e8248542e799492
11a9e54664878390b31ad3c032bd2970a062f04cf54d07bae699882c54856480
147b034ddfd353fa772f1d1cf91625013fc1e1f0620d3f1a859fa96f7b7cc9cf
15ea2d1f8a74fbe0dac3d213241fc210f77779e6681ec7aa570a56a65c197537
167142b256a43661a934dac8d2e84d1167e66022acb9e374e4542adbaceff753
17fad325e9717e20c930f698f08f711320a505560e239b5de9df67c62258a3bd
1864101f2feb1fb2f307ef13f0259290dca56bbe6cdb272e4fc4c1458dfa256
18759f7353b70785a05372f45cf7c6092ca649a67b7f139883143cc4a0f01387
191ba5413e8cab449cfb39d083b2fa780c5fd331b6e7a0654e536fbb23d20458
1932562dc1fc3dcea7da6d86ef26c12a34b0acce777807a5f0b4b5922950df05
196cd89ccd6af1425ed35398c2050e260b273e9d5c5789fb9ff789f34849a14c
19e17a4a2e84b9c69139249d1891e3385f7667d8f5612928c6c863bccf494fbb
1a7e6a15cb68a7921d4dd6f694f653ff2635ddb7dcc64dc4a3279f0bf7294cf7
1bba79e46b3a57c57208f5b0c0ea9601e7e63ed219fb9c93eb2d75fa1045fb05
1c458dbab474aa55c42e465e34a64faae2b567910490621aeda7ffb51b7b29dc
1c5652b45fb09ba916039154acf76577fa141165511243407e112a7d8d27b78a
1e52736e367a32baa3f72762da49a9987451874e34f4fbcc8d4afcf15a91e537
1e71e56913f6531c96445bc90b51d49e9a69574236d97b3c36ce99db4f9a9ee7
204f5127fffc492956eb77c01f5196c8b57a3d084d575594376909d22269018
20b17d6dd7af65d01372906696d4d7313374f2a12a8a012d1cbc76cdd9d629aa
20d3073814656292b0e0604dcc4ff918ec607af69c41a0d8400376cdeea6c236
2106aa8235bc1e05d367bae1e65c50bf8d11c4b4106ceebde0f9307ffcf273c0
21291205dd26554cb4537292a0e095a126db0a11d2ec462c86632990cdb15380
213fbce6c1598d4cd9a54ec4008f6d531f317e81ed125a046ac1812ab8181f53
21b03041f46e3e62f2500998745e5c1fb98996dd622ca046653fa0501278ced3
21ffa6cab8603732b5f615cd0db3e5e6deef95d75fa33598815e4bcbcb1da691
241f9811f1af09efd755e68dbed1decdbbacab284ad1a3860dde710f17e60f94
24704e7cf611284dbe00a83b66ade87369dfb967016da89e78add0f004e5bcb9
24934cfda7f567024000d0992958608ecfcd576e72f9b2f49767f9cf87d8802
255169d7b1e38899a7ab422e4c44cf11b39b0259144178e24b437cb864674a4d
2577fed0bd18acf6b39442264f92fd584f3ca4f7b14615ec8915a74879a625aa
25de2b1bb1ce9d01365c666c9f04b8d021c6e93ce4551c545eae8de66fb92944
2654563a4f5b682055177fe3c7c51a5ac9c1f6a7789fd96a3cb3b6158546f0e1
2654c41feea51b45a2178689043103ff6b732c3dbb727b8987205a7e393017ec
2666afc4946c89ed6fae860821ebbe0a0f0c0621b5f6f07ceccf5d390658205b
277642ac517029807bf7eaad2943b6e6ae9bf29a3d06a568408bab65e48081c2
2777c2ab1358ff442a0744634600581a71c0ea57b983437aaf1b2b184e249c3e
27ebfb9c8f82ef351acf50cb434134cf6cfb3378baf389ab84e7f5a397ea3df
28deb46904cc62b51f198c1a962a1d0d9f91745de1a242a4711c1fbc956e6322
2980f607058a2ba908559bdedbbd019edd4b52377e5f884d36a62bf9aac8be00
2b11d3a570fd45fda3e8c062f447e09dbabcfafec5e89f4258c0ac7c6e294fdc
2b9b066e3049207623a9b58439eedeb0c226895e8a37431fb65ab15fc168636e

2bfd8562d48b7fed321eaf2be83f37271d08b16341d37990e0782f36c98c9efc
2c57f3f9227d165321ad3ec29060b58358f3e95968cc1b4b6eff7eb978a993d1
2df2f95d0b480aa7d2f86ee162298a55d7f24dbdaef2e06664ce992c35a5ab03
2e2086c44eedec39ef9247840af6dc6845a2618de4c3fd0bd6b293da9890854a
2e2f674241de56d4404a8fdbcb470fd9738d48c855fb0785deaae99ea3af383f
2f356283c209400c6385a24450f266b59477e035e9389c8d1af4843cd1ad2374
3092605445c3acdf8d63cc7db1924b8ff09f039c1c427dd5aed008c37b9d5575
30f816e55b01b1e22af3e9d7755bb75279b7e6f38eb2ffaf790d6234e5036709
31d9c25d965fa6de72c0c376159c19da21bc5dcd579bc038b2134d19b1ca9ab2
31e49f1d5620418ef41da80331c1f3bfb578566ab0fdf53fc9890919a9896af5
322a82cb52dd04f516915780a9eb9865e24bd284fedecf938e345270fa0f83b3
324021b49d9f06b14892438a238801442dce7f23fb2bdfb252319b34243213bf
3276e7ff57ccbea104651066c1d45301d52fbfce23d0d48f5238d82a51abd852
3294760e6a8a87629ab677b3269533d741f4cff17a92060cd10659a6b0b3e78d
32ad4debd99aa2f13ce76dd248ff89710130e18e0427466179217256fba79357
32ae7f0efc33184c2466e08be66cf3a2e2c663d533c76a12751ee26602a6041f
32c43439c840feb8dee477a50e80745068c13a938654ca9bd786cb6f74a49558
32ea408d82f03b7c6c9e65a72a1c83eb8934e8179e0205760834f489058ce0d3
330a5459a24ee685949c0900fe5b882e57ded7f1bebd02b03d82a8301f6feba2
33709e54333a4b6cc832c1cfbc3eae6d7c256642931f2fb6e47d26bb2d191cfa
33a39c038e941e77bb0129b765cc7ed91a4126a8bafb435149ef4912afb62a18
344ae4ecfec1e2a0ca7c2e6b970f077cf55fd1c7cbe41d8963b83848ccf0b229
349e867cfbab437ac7ada1e7c25e093a180135d34dd5365f5d0e838fb8e713ac
34cb99613940f2408bc3ca05b9fef7b8d490cd8cada151b65251a1f76fdddc81
364d0bdf248f7f0e275ccb178e50e55cf76b13ec131a0f2156f20526ef24991
36b5580f0239d0e3ab71541b1bdbc26f6e3e397cb1978b91c9e358e52a03de79
36f515bac3960c07aea759f03208f901b84050cd57c84a2fce20e92b83158b3c
37572cb6f2bd3ef772a437f0e91cd813bfd270988f205c7a24f200ba4df6a609
38b404c76de7b2c70cc770583b5917deefe31be51acf2caafc39b3fd884a9ede
3987c7ba08298c6fc3d6007468e751b3b751e750ea3a0ae5b2c5e699bac97002
39d5ed129220b55cc86878b7ed4076407195a7c1e4f0dea37f8985e0444cccd5
3a079fa5db89453bf743d288e9d2613427c8a849abd4e6b7c18d1d80158fa3b4
3a5e2c67c21dcfb129ce78a036ae4bf136a05493001ad7326c7fe9aab6a444e7
3b6283a00901c0709734f4d16a2fb4ad3fb93c913dbf45637664c74320c2d385
3b980a607bf3bf7e57c2f0745195b2040ea0aafd0a9bea23d0985bff045c72db
3bbcba7b1e14f51123d2d83c17cf5529861c110dafc4d18b45acbf8f8a17cf54
3d4072b7826c27f5aa407e7ba64a3ce64beeb00d1fd5d02042afb9cc95b18835
3d53dd72c07a1d6fbc44f658f9971383573ff28bb10e1f86049b79bf4b55011a
3d63832817a34bfe4f72a9f22ab220254b00578bbdc0dbabed657207a3c6af1b
3e3615ec6c5ef753c75fc1448d7e5a49136e3c7b5d2016e107107f17c4816aa6
4060ca25ee645493d07079398edda938c6d2765348874fe5359f6a2b556f6799
41543487dfd0081c50bc9dbb1c347b9a2c21763413ea1d46dd62bfaf6b6afca7
41ee49f33dd7161925df43cdd4f0a0d06f19db763008b63e1b66742289e5dfbc
43b7d703f9ea0049dbdf0754eabc0cfdb9e8e7e21fd423a5a800ea957d624b3

43e7901e0f50ea1ad4a609f04769dd0a3384443f685090e370e1d728595fc85f
43f134157aa225468a1200672472d1cfe90f5ea534eb645660f6975156a67beb
444c1edbdbac15ae07622c50601aedef7c381526fcec95f4f20cc051650905d9b
4658e3fbdce008655176741c9d433fd3f42e79990c72fab345fbf5777b1cc291
473e99cdf2dc25a6bf43a56e9b095639776294bea38321c079cceedb3678c28d
49a81050cab889308811a7f2f471ea7930784bf8e5db126a3affb0bb21aa98b8
4ac30ca3142675b81e4490da111a69336cc0b41b21049d3a8bb6b38e7851b529
4ae331af8225461783a3c83a8b64224311c62dabb41345fb4a81d5b06a0a04ed
4af278d681249a1d292e0b0eb662f20ec919f984bflc3d3ed0cff961fe0ef887
4af45684633faa90fc68b62flaa361e80d1f7791844d53643c1756f9b29486a6
4b01d8e4729b07277f8f71037f9fbda1f8d817d9688850d941e7832727bb0276
4b051e1fabce6f9926c1e6af9327a3b7177c092b1acc522fb98bbc90b33bee12
4c1286920e0fbb0e4269f4b64ec6ca052076414a24af72f2e1a82f516a21bf52
4c32e72bb790b0ab450d99b8523364c399d694de1f141511e9051d33b3d860db
4c988891326eea9e5fe8aeaa93ffb3f7001bd3cda048a72ab24c8b54dd1834e4
4c9ff241698ba8436fccfffe86eee42a7c330d3bc378152592c13a1b41cb8a3
4d0f24910f3230e1ed76355defd83a866ed82b6006884d63f4a3647c8704edcc
4e099dee4b6248ffd4a4a86bee02311e91b065143138b6bd87e1fb5bb882bcd7
4e1a0a523e4ee04c32358c3299ce34aa5b6e70e45382b46baa11cbe5d32887ce
4e917ed708e6c20ca6f74372e37680aac6af8a9fc214e903ff4297438cf94261
4ed72ab52529fccb45200e4d4b178cfafc6c959ab34755bcaba9fb79a8ee36b8
4fb3fe547fbc853b4adcb7afa2e8ed7a4aef26e2fd7600c80afb543853dabe6
507cd22c5bb850abf4ac9782a63660025b0d27075d52a4a3bdafb40389fd9948
50c5c887aece86fb6b2dd1184c183c87082ae8a126d52a7e735449f6be8fb68c
50ca265fdfe8cf164553eab678b3d6491cc940fa0adf369aceb55a66fad1d4f9
512041d2cf26aa8cdd11e5c0edd3a5047e887d18204e554875026ae850a49ffe
5227918355b47bf8f0cf6eabb0628ff59339d6fd6a3a56ceca4dcefd06dd8d00
5257052626b2ddf18d7747e19be5425748344cc57e4b297a4d862ff5eee84e46
53a2a156f71274fb0d724533c57fca357e4d16a8dea55395e58a9f0f00b3e82a
544bfbc907da2d25976337ae0bfb6f7135e16f01cadbacb62c21e9020f953f35
544ccf26cf0038cc8d76b952e349f3e49db27e4d2cdc89ea72fa2fb8e8996038
54b9e3319deae6b78387742f00603330618ff2845df127b3a6290c7d295ae5f
54fb14c0a6205f4a9b4f88fd9dc53b8bb9a9791618baf80ad5152e00d9379241
54fdab5014a889a2185845468a664b3bbd39c7ebe97202795de6449ea48cf9f1
57bf2f3c435a4941efdc94295930e59e9eafc970ef8422d42ba20265d81d4930
590277f960d74add2860c7332ae427eaea968bdf3bb4d5a8da5563a3115b16f7
5ab030dcda390a18266d9cb2834fdca0380b5aed30aa67ecdbc06369f555315d
5b03580eb279892ac9a03742ea838ffc55bfde4c4a1edf66a03cceed400c81e2
5b086d5fdd4e2bcf856eb5ee73d1b139589c12d3416abf3ce75042bbdf4ed374
5b72b74c569f9a1593b4bf313a02db7e10f9e8ffb5ed73ee73ed76b1c53ad13f
5b7fcd72ebba075d2cf0387b7d23636ab102808e9bce0e17b9bb435dc2d91f
5bb96d0e0036475868552414f3645cd7bd779629f1c9ffdbb5da32af26fd641f
5bd660220c19c632f61015729a2b2d94a062a3f0d84ddd49854407e9a5965449
5c50dfe4776a3d34649ea834cdcba8b880c5651706473143900964e540436fc6

5c808f0920131d45867eceb3aedd79c4bc90d496ce2a67f2eb12081eb59af9be
5c925bf7470aa424e16b1293c55215ac430d36ec0dd6b21ceed0343a5424c9ad
5cd5cbdb883fcd9f6fbcc823d3e5473ca2e993772b4dacc45b66564f556eddc7
5d01cd68bd03ac141e8fa6e428028bb8ac569988eeea8b96b78b06249e4f4e5c
5d5656bf50bf4d14a6b4129c7f3dfd9f446b98df3edeaf2d9036a77d49f52349
5e4cdb1fe04c466a44dce83bc1e48ccfc9f8e6baf9a989815642bc9b8950ce8
5f2236ca0b43e80e806c02bf3d5c2a35f0b31d9f620b9bf19604d9e47bb3cd44
5fc28dba0030fcbcc62c9ca9c6da94dbecd6e50ee64f4f5380e6a73adfl6f627
606919d284b35ccf877ed348d03cc9f2000434761d3f593b22aa4202da452f7c
6121c5bbafe4f3afba5ca58695f5951e99c5d638362fc9536e32a249c16f6442
6140005553a006e1fc0f489312d1ba70c84851f9e9a503daaf9665c72c1684bc
623c410c316befe7e6cf4d8459bc0d112d7462b40dd31c524f10f268f1ef378e
6244e1ccfa5a2df0cf729efd89ad8ffa7558d44ba9c01363c3886cc4ec445a1b
6265a83f4aaf18f5a5da3cbed44ebe3329a499704c7acc455c2d7d3b8ad6c33
6271c48e0ac2fa210f270581b7952c9e065722927745d52b5959bb4097040a70
62ae5b0e2dbbcc5c0d78f027b877861e6aa345e5827d579e13028d8eeb67d315
635462342c88f30c79a7bbc5a8e594a3c72a9bdcd33fef4a64228ead8de93399
63816bb5d6e665cefcfaeef55c8978de75d972e2e710830d23b809f7ccc81cb1
648048695290a99e4da3a55882aa6c2a62c2ee1b4fc780ebf87213f61909c34f
64e0057d190f33bf5ff58de46d663ab4d075b3176d1c59361f931ab7f7dd7185
655b12a219d0f0e39a84fe44483e25411be852ce2bb0d451a1cb1a9a670f70b8
65a7ff3a2581af3a0a010b3542196d481441d02be98d9ccf44c0e1557d16a540
66242b095b2cfb53b52d1743a42aaa9fd94c6b53f58869c4b1c9d893a541e3a6
663796a5ce4127446f60d77bca21aee44037d4c0eaf3b14449f4a54ecb359428
66af175d79a1cc8fc062a0bcb87b2c14ed79ec79d7b26be017b7a46edb24489e
66bfc82c4f69f36538f7ff9f5949f72288805850f4b64980c17ec930c6baf224
672b918c5c82cfed617eeca2ab662854d2e9feef0031f7a72025962f3ee867ca
67600d161c7a30d3a06b1f49332e26a4e9010536dca825ba294802934c4358e5
68076b615a9a88dcb4a614a82412301dfc132d62b63a42d1bb7e0bc8bb66c8c9
68509fae7cbee5faa8b965ef91252537e15c2fe96e76885092dbf575a9a1e98e
68dc5714a90d2f0b2b91d22127aa518ce6667386b133777a0e36fac29d7a19af
69bd9ed0925ad3545005a89eefe926a70a1db5fa1ef7044bbd652d1f8580e069
6a3dc848c0ad77c98fce00a7de5649bdb23bce7c725c10b8cb75fa2f11c7e8df
6a7b8ed7f2cfe7f8a3030df54f1ba443025c54e5a15d416469aac2a696d7bfl1d
6ab1f2ca3b65e6cdddb9e96235f41bfa1a483912074cd82af14d59e698655d96
6ad914e303ab4e74eada71293402d83756eda6e0556ba0b9e2322c39f82f8a42
6b1ee9252cc101156e585e7cd6c59ea3850a010c57a7f1b1915d5438cbc5b4de
6bd4365625ded783e8eabc6a1366cdfb975b5a9dc2758ba18bce9a2812f1c8e
6bdb216a191a6e23f36ed558221cdcd5bef56ee67c2b6a3cc2082b83b8402562
6c5f49eb7032822f64a980ba03f7a369994e46082085edc447e55758a5894cf4
6cc138d608ee3008a55e3e771991125b411c02cd2c3c914193c2b0524f031246
6e148ab00c0201c04395c4709b948ca849f04bf10060b882170986307ed4207a
6e20050650b7239b89086f3473de916805452d82e8c0bde446952a4faf4847a3
6e6facc7f1054d6369228713877e4bd7dbd4daa2c153370a1abc3b7c1b548ab2

6e9e7f85765b936dbec0d489d4b30048881a558489c4c8691187a781117c9b9a
6f22c078ac85a58a121a9f601bb88bbdb1664bee8845c5b66f42f1c23d13a5ef
6f6ccd70e277efd92f43533ce35f403dad62d3dac77c9d502d756894552043f7
6fcdadd8cf08fee0a4554ad2d03bd01e910bc4460626910da7f1e29ae2b6faf8
705eeb755944e5ed11114af1a6f01a2955214275fe2ba7c9d1f6b88f6412a8d4
70612d4332ec94d8443df6fa1cb9fa43502e54fc7a46ade65cc52058a824f056
70a17ce04f4103804cdcb27e36235a89ba45561e5a477da5dc1a5e6f604d0ea7
710d719e4e99bb8b57db7b490b41a77043d0588c577cd645e81ceb0d17d95f7b
7193f9d4c0891fc4c0f9de21d97c6f77ec401f1636742969c8f402131fa38721
72903788ca82bf6ef459c9dc518bc1af0227dfa78f3800c982c7e00ced7ff6a7
72eb839bf81fa68735fd1680e975397a43d2413ecfd402a68db0f2f02eb3f104
73390b2819746a46caf0923f053b45d132bdd4fdaff52165e42c22a8d28543f2
7368e15e16845dd62299d43c9ecceed80ba8852a2f4ed36379337c1dd933d48a
73f9225cb459e80c3d6013f5f675f7b7ee481e819811f6ef69dbcb2308db72e2
73fb9642d596a55c8cc98409d3336c27158312cead76a0c39fd4e9de813893b8
740b8d98eca229f6873a327b046f509c7228ae9e7cf2acabfc43aa5c0e0d5d9d
747e7e2681dcb1663759b915ce6a656d40954127343aff9fade05be78d3d99df
752362021953d0160db0daf44f7266489e5f97d0c0893b074f4315dbdb9636b2
75838cea0738a0409d64c33454761fb1274d72c3673bc8462ab2734196704a15
75a1ab8cf9990ff406e8c0f6cb7dcb5aea1a45b5e6d5007b905c59b4e1d2fd3
75b803222917f13da307cd36a22b56e0192860ede2c368cab45d53cef5f4422f
75fe4b601dac47a21ea34b057b8c2ee8623db40d6fbb6e3398b77260ed38eabf
7774df1e5c6654bbf573b29464b57f3d7fa0a1b1eb9f6865b3bbfcd9f88d99f3
77dd82d858166a180a5de6d9da606500cd41fba6fe1fb7bc09c63162af78bb48
786aeabdb7664ab3c99105d7705946a0d3259b9096e5464db67eab31a2637194
788b816869fe860194e7288e7138e1efa96daea36480deb5fbb4ec5b894289b4
788c228ea56724227e6eb18a04d36fa4eda9dcfd5a90f1131a54f01ce7445a48
794ff32dcb5a26819bea2fa85f82ee36dae53a5d6a6d2e42790801bfb018cda7
7984714891615a64abddf226ae3538409f5554abcd6513ca58eab39c636e046
79fd22e1bb6fa5e88488288e2472fc4323948ae21b5ea4ad0a9692b0ebb8b835
7b4a943768c56d1e3080122f60e995e1b22de4dfbb8c806d583adaa21e0de66b
7c042716cacb46c8e1a105fd13ceb1093f20d28c869623c4d2236805876a9f1d
7c470477c32979be016bffc98f861acc089861a6a2ba015aed73d769774370bf
7d2679d585d5fd6b476830fea23e3d0ddc831476e40464cee74743a2c853b81e
7d51363f1fc990710be0a463a7123a5c5cbe532e81164b1c43d4a159a89dadeb
7d9b86175551c82242656b775033e8826604c5c534d241b4fbe0bc934a2bac26
7dc17f5f4a8da17857474af7083fad349e8e9799c16edad0f3753ec007126b08
7e88b42b5ddf91c441cdaecfb073a44b2aeae94f744df4d535250a1be07ec79d
7fbe5699e731ea97f492f07916e586f6c1232fe97ac2b9378f1e6372e0a06116
80700ebe75f2f735f21f5195c56597b15fb5c82945484406a42e85bd27b04d02
809ef51666d12a1b245eabed67f37b5834b1bdd6b90940b8730d10bf540d5f95
80a92f2ceb76a9e3f2a5405c1d2d26f838d54b5129d1ce97d60c4af88c07dc61
80e3e9bedf4677e6ae7ddb75c5008a2918f9201812cdccca07954ac3411b2f03
81278e61b365975c3ba6eb47c4734c4831e2488dfd3357cf8d0d018b4a57123f

81775364bd4784ed3f66a1bbe75d77d04caffd227351f872cd42f20047f84eaa
81adbb94cf5758852ad9d3e7ba4d958b1943715c3837074c7fcaeeeee22aadb7b
81f5eeeb34f22bd640cee738295303153d9afda2a35e365d061d058fbbfed498
8316cf1ff4d3efc0b0718d3f517fa9392bd68c456c3d0c96306793e210f48b15
8395137f5decba5ceed320e5d8e267cfcd4a5fe5dde47452018e098d32be3f8d
83aa41a7f164cf8e0d298b48a95f406560e3c49d42dfc7f0430eb54f61840add
8401c053dfd509441c73c2d06619fbfeff3794da4c86b59f4f7de80db539e062f
84d35dd9518f497edbc648709a6ba30b782e57c83c2e30f91476f094ac50681d
84d9875f941844ce35b7aac6e6e7442ea0c954025ab7578790854e923c099a70
8558b4d389b22ea8b2ec98c62d43646e2b733fb7c3209d6d7d2f0ae8744650fd
860ab31345d8320b749e8a6da716db3b74f4e2cf8bd6ac936dd0e99e6fffeec3
868c3436d8926091378d99a17eabdf20f18e58e7e48c1911ba93059f14df01c3
869b27dcb4475a95c0af50db49fad8c40730df5464a5c5482032ce5ffc578feb
86bff0f6efa102b2aad9d7dd04f9644fbc7a6c13992742de8f310bca8f39655d
8833ef255933b07f1ae298a7b06cc6b87b664eaa44f7c197a84a097f0dc0889b
88c35140b891ab68d50c39cba262f80e48038a90e92f46b685240b07494714a6
89ed1483ade890ada1d088ca1c76a378ed83043fe2dfc877b69788a5857b375c
8a1763ed91296e8515f7ed7311fbd5295fa9e8b7e3f831171a9d96bcc96a5cb8
8a530805d14a1d39f4016d711091acc26f7404a1dca050407f6d6ba4eeae0f5
8a805d0a7c36754612b2974443f36910ebe3b64777c7ed4dac31c7031b40c2c2
8ab395b226e1a1fe22ae66d900f307bd6215a5a0ade9770e81a4a31ee9ffbfdf
8ac1f7de055ec36e94804c99cf51741ec440b128e92f9a721749802e1204a0dc
8ad8fd02dbbc18609317abda87ad1b1edc03e1be75d2fd9537678249765bcf65
8cb3180f36991a5c91827582e8c7ed3cc8303dcec4362e285bef161585e86d84
8cce7e35531b0f8eaa5f67609588d2e0c0b72bdfe75c3e80d8058d74613bd964
8cdd2376c22a3f37faafe3a39f3730b7c03c9e641b729607ca2b083abbc3f05e
8cde8e6a8021c1d4d221179126d6fdd2a2a20a2b24f57fc20202f86640350f96
8ddd2ba01927bb631eda5dee78b63f8c9b91db44e9418fd4eb313b6b4847f6d7
8e603a99770c15419c36fd082b789d2285e7ddfd77da47cf9f9be899c350d913
8e6fe812e3f4a19a51a0978e9c47e2cdb891f1fecb0a7ae2c1eff744c971371
8f0027831799411e224efe5f6c740c3613c91c17f11edc16b54a8034c4226d21
9015858b495b167f238084816fde2c4017d0a991f7c60a75fdf6f6444ca43abd
90212e18e6b4b235bbb6f4083bd0a2c491e0be12edc600199eb0ed0839d9a554
90528ea50fbbca554dbe4d5f1b6a39b5731bb36a1e1efb79cbe906685b583215
90b786543a37b1566f5b8bca8fd8e7ba7915d8f97217f2b9f985248a8d0d9fed
9111218e89daf7e547af7777d2cde1fe35e029c1eff85f991349308a51af3d3d
916d30f29bbbbb9cd02ee4c66d611750d1a0aafa7b3f364b35e46216ae90a3
91ad5e288cc86d88bb3189716581fcd1bc5d0f13b2be3a471c8758fd97bfda69
9258cdc242c0e1759a9de5b0cc9382c8e49fc66d24310485badea6702aa51c5b
927bf1f7d51aacd7c7e504a0dd55f933b0cf845fa76dbe28740689c1aad7b9c1
929fde54449aa622ec9bb5e7e79f2c19d12f41f25cae22daa559db9fa9862046
933b7f5c98ba6f0c28c54a11d35c8ac0a36d825699df11a896767556ebce1603
93e14875bc031dbac8ba67009e2fe2f7c3e52e7f2bfafdd01ef929bcbd593851
93e9e99eb9d74a0cfa3cec7fea4a663b9ad5e8fe4e6ba38bf597a902ce592e1e

946fa8826403d58c6694d18b89af7bf80c078f792f9aae820ac9ced395450f63
949ce457953cd74abad14b375a79dd0d2d6f80d08868adc55b2ba40cd363cb80
94a6304c29b0c62b6a8a2748e9232e8822c8bd5695fdef506d836368fb63f61d
94b075b8f1e428bcd40fe4d9062f3449dcaed37cefa1c3ee8a599f28d73de631
94ebccd8a71740cab38549bb879d7c1a393c59e8b481e2e7bd37348e9724f39a
95f56fb182812795848d78ba412d42c76ee717945a57b0389b24e53e6fbbee04
968be89e91c52db13b473bf5894e1b11de82551176fff293d752e388f7e83175
9796b66297042ac5d8a1610afa233c9dd98ac5e6d514015a8a917a85a4474de0
97c28174a64eab003f2a1b2f4a742acbcbb8394249d136d176c19711908da21a
98e2349140954dcc6a9cf4cf4007adcb10687fe04cdbfa88235523168b0ebc0
9959cee30bba30b9021b1b209d7381d8859cb7fce7a7f4e147e9bef12b81579e
99b448a15910c00f26b19658c004806fbcf995d2669be8bdd74467bebaa18024
9a48156fe2c512c5ec1a1a7292e3ce4c590e826731678db73939346e1fa0fec9
9b297a8a63be5856dce916e4c52a0794387151485913b4cddb5cb9180e4e8b21
9c056a201bc60f2a7ccea833eb627295dcdad68e5d6f8782c44ac7fade74ba8d
9c0cab2fc38f52243eff7e3fa43e088d76b0a77d1253230c0d9b28c8f64f611d
9c9334c90a2e559eed3e8fc03ab85709ab00394cc4c0f12bd481d70f30d3171b
9d15ed9987b1497790a0c1a0a07039930a190d3ea55f6525f8cd3ed74f169955
9d172ab971a82ad60d39b4a9344f4b4c386c0f9b578b0d85f5596fe1d1cacbd7
9d617fa144dc0f4e194ba8b1651625f0725f89d2363072e3f6f4357262f9003c
9f53ee7ce6ca8ed9eabccfbcb84be47c1997373e81fla577bdf288967e0ccc05b
9f7982f678ad089efedf1ebbbe9fc890ac396061ca36db87288d0fa84a26b039
9f7cf45ec90f610f2082d73af4e88d1d4aa2ad1ba0ed8dbbf2b7c62750ea0913
9f9a67376f1c2f7076eff2eb971329895aa498cb000d0cec97ed8891cacd8851
a07ecf179de4bd36031ffd105bd6c8181b9a2fdcadeda398629e090a3438b6db
a1689d72306cf3dcdc8a3319008387ea66113ec85575ad4b0c905084e6aba306
a18ddbcb974723faec3f0bda9244216649786ca41471cc63049999d0408d7009
a1b9732cd03a120bc68fd88cc42ed1a138f0c3012eac4222b108ca1740b682fe
a2212018cea294d780ff1f6bee1ae2a71889e688fccccff09da95dcb9d228e3d
a2eaaaa66cd162d90cc6c68c0dcc379d89ba746ddbad8508c911c7a88ba86c9d
a4392c3be6ba337f370c0b2170bf46c6c65f0054304f355d40b0b7ffec718e4
a446d6dacbe5bac46382ed7f6302f21ba7b2a903cce90eab8b48387df8b0622f
a49555b7a7d09a2fdc27ce10a8a9a7f5a7c2ffd985a3c34723795c442b7a897d
a6251f77e4c34d7007b6ac5b9e76fb2905d90eb74cb3b5da23fc11ac437f44ad
a6785180a4da92c5c7ba441f18aa3f7214e14289cff670b39abc7f6ebd9fcd4
a69bc47f1d02de3a9887602ae00f5a1916eded288b44e696e66ff5bca2bb8f50
a76eaabcf06b86917aa8bcc5eab58db5db7cc90ca5682e64ab585d86d662cda7
a78731da2617fe258f08a30bb02775a66e2b8363e5b84f97aa61a0baf178a927
a7c01a1573cc4592ae5d66c36fe7a35fc44c87ea5c9497e9cb23d7271d7e75c7
a7cfc4cd3bc4c1c1069e9db79a026f0fa4e362fc3adc086a996a5cc355a64f6c
a7dc1d7849adc86bc322f33ae8316be94e572df4442da18826330c2bdc124152
a7ef23c55b79011e898e2b7734e2742890865ebfdc309dbc7bd6e2f073639a7c
a80b04632efa4771237718d8850fd0c77e5af11d11f1d468f2cff2fffe82eee9
a88ee1a9027be5f82df034c1367c54c7b3d925eb17802a77badfe8423fc54f83

a93b70eb6d8b21e04163be6ebc444ff76a557040e8d97021ec04a5a002cd5c8a
a97f182e8e7da0854b932b946352626e4c94c6f1319ea6ddf5cefa854af93bd7
a99ad6c50cd5ec1409151d618216ea2616d10d8e3959f8f3067b257b172efa4b
a9cf955162a9164b63c70530a2ed72b02ab53f7b39a3a9ece842cd2bebf117c
aaf9e62823296f07f2056a5c60cd24437985a71bcf5fa7c12d7dc9a7810c240f
ad5af6dd7b293e650c8e5b740aa48de75ae6aec0360489a5306b8cab6482a99f
ae91d68a41e56272d651d94ac67588c7c7fd558b766af81036d141068c560587
aea7a35212e49f49012cdfffd1439eb1ad9e6e761345b17ebcfbc5a8dd9dd7a5
af740e72fdc536b935138fe6974e3c4a7bd9956ae6f47bb86273cf49365c62d2
b077c634471ee290e38f9e3ced5375888f93890e6b0840b8db2c1691175019ee
b08185ae6e6de33ced47ab140ac3e4b71198be932999848ead61f4adde8544da
b0b430c82cc574323d38d65365540472f3f0e6133dcb36e20ee9fcf5483769fa
b0e5e12bea8386e6e06c82e4e25257b22649a608b2ef2a599332879983a000b0
b13700f6b5654760b7462c832b65592d6cccd3498f4094293fff020fb2b40970
b188a13a9f8d13e388089ecbe4725f5c0e2a17c2f1036e0a7ab0cf5aab878549
b27f7a44bbe68d1ceb1d8d2b0a81fe4dcb9dc8047080e6a79aaca37b409cf240
b33fed1a67ba44a36910b8a91b4c5478262f7b5cd0cc0621a72be29fc19a1c89
b3735e53d6bd12bc0dad5af1f6e67a6edf728f45cddb1d3ea0caf09c16fe04
b3ff87ae7ae92b3763bb9f88943b84fd58ba847a42812222b4193638cad09bad
b51a2616cda21a22be90fe6587e67a44675ee5daf65dcc18bb1516a09090bc43
b547804ebc3fd98bd35ffea608b152871d8387a473ac22fb547836df0085670f
b66fc3c8f01ea488838e57cda4157b5ec8d3398b321fcc68bdb9d7ce397486f5
b67c389fc71d512caba10e28fb950648a2971e42581698e1191f7583da2b8309
b785f053a5f8425a451fe36e0650b0353793e57cfaa25c191f871fbb82b2ae49
b8216cdc194ab1f82820fc2a29ca63af3a1f3e6b80102ab658c1ee33b7dd68da
b83277e8c7164257c1077c18eb894209d211d79b0032ad50230553c5b7a1e411
b87c55a95b3733d513090964b501d39432d77884a8a6991cf1e72730d60d67f6
b8aa87c8e614e00306b46db6b98507c5ec4b8a81cad30bef832eec8ecb4d619b
b921e7134e8ad218e90ad88491c0135ab7c00d5557c9781aea69de52c5d99207
ba226fb52801754547873418da0f884d05c0adf410f4aa21bf2d111c0bd84dce
bb01ebad742a61a5aee09777d88a01e627d29171513c335f52c57c41c7e41ef4
bb43a5754a893cb89704ada5e6ab99172fcaba378a12b890b6baae58fd6d35ba
bb6a92b2c43488bf8d0310090aa4036b5e292c9ab5030c8ebdd1864cf015f9ba
bc1cf6ade2e07cc46dd254933985c39647afef5408b8e55ef054f438683a5843
bd078e66ca3a624f316e932b32b088b22fbae4042de5446b1ba7f0c485977803
bd9e8e98b57be42915462ea8282987ebe17d779ead3d4c6461ec9e4d59150b3a
bde8b9cb1b5b006d1088bdb37f82de7e7c8ae5e3d3e02cb143a87d70ef774c4
be34ef57ebf4886266df46631de2a251e95761e916b7964ff57973d954f77c10
be6eddee7716fd87a1ff8c8b407bc3895b6acb165ecbaec087f1ebc4914d438c
be7ae51f020f20a576174c9d068e79981fecfa2f8f7fd229899b0998f156174c
bee4fde264ae5169a4cd306394075c9542f3059e8fdd6390347cfbbe52e77ff9
bf305d7a93949463c9410b53f1c874a42e1b0b6c1966c4e82520dfcf352402ca
bf7e149d1f9261676dcfd400ee235372b01f64302efc5be2eb053308e1203d73
c018bc22c700a8f6e4100b843d3a96b2409d6c215e1062608fd73714236764fd

c09e4ce0e0baa34a9c9043f4116639a4e0513d953135b4408fc5a5f02a9d2b58
c0a4d4aa0938886f16c9a805b66ea21acbd96c9092a56a7591459e3d3f3a20cc
c2baa049d73f9e491429c2f19d88408d4048465b7242ec6d6b0f1c594646f366
c36de6d07a8ce4407cb59a275dbf8c04d05844903bb6d566f295ccd13a2d4ce6
c38a1d6e03b28f30dcbf1b8dfb3edf2d184c10e32c556e1f7c5359d4c35d74ce
c3da35766206bfb026dc3cbe07ee301327a2def8f353ffd3a8099a3e7561f18e
c4219d6731e844fe6583905d041a827bad32cfe74fabcd6bf544a52b6358320d
c422ece156e7a265d2d5959272da3720365ce75a02ea6bfd83af31341638eed
c423a2cd001b0aea90ff3ac2e191d3acde70e7511fa84dd57bd576d729144752
c47f1c11eb874ce92afbe8eb3e24f44609f7d7bc154f705116975d5f735a1d4e
c531fabadc6804c5ec1a9d2741e8e3a1366baca83b70acac740b9996489df4bd
c5606addf13515a76575a8c76f0c4a5ea704324eb840e525c5dadaa655af124d
c5beba5301ba17a0429f95ee160c51fe2a06b7b208865b524556aa09aaec1e7e
c6070d16218077441b9faa6ed769b764e27d7a506c2eea9ee98527d88f2c8eae
c613749fa5ddfbcb29b303f99f48fde9a00ae31c5d560e04e92922df1f966be94
c6deef7825e9fc588ee77c25398896ac695e0c02c44276fc4382218807b01e17
c71036603d8d0b77c51cd2c013d22c137ea00eb44ad0553de3019dd9b6e0451e
c7a456f2d97f0c34cc5fd56f6825d60998a5d730e7d1e9157867257dfbaec0e7
c7b659e1222e6424a20002cd30cf3e3492535418a4e91dfcf3e39d901712d7ff
c7c4fe957dcaaa05d68315cb79441eb9159d4b4f224c4cd84d8ed8fb5d9ddfb0
c8517f6c638a7d458d5dc46ba0c8b62c22165996339338788f9632cba03570c5
c8676f5a35ecf028d928368d10e861e4c6e803438861ebf02e7256a6f9bb497b
c99832c103472ab073329d432d32d981fdb4c7c79ebcd7352d37d9fb4f1212b8
ca36938bca50bc95f123c86a7c886b2da3add6b082b31146bbfae46ddccc473c
ca644f4092d3fc0c4a9c5fc48eaad8e1a8c397f79e6fcafc0762912c9775b07
ccc7ec3b61ede4fdaf44b5b85c6175bc617650155e2913a076b1738dab059487
cd12b967014c7f18359b95e1ec52cd81d0528bbc2ce15d540d461bc431eaa71b
cdf5c57b42c0d59cfl6d83dd2e1999a2bc874fda7c521b25af910f80fd012691
ce1378ad4a7a072294468f548a2d323e7e5fa074a9ae0f65238a6bf1377f4c63
d00755892b27ea561c12ebf3cf369c405177152a586e75614cffa92ce0eca10b
d079abc9b57395804589c2583f98855b02d8aab8f850df5c56a9ce0aa1114508
d0b6d352462a66f0ed92bad1e87f7cfb7051018d2611f31c8677d928f6bebc9e
d0bdc69ed2bcd5332c1b8f871c78a80b4dedf2196d4151c71d32e14fd977050d
d0d5ba676a9697878e8009e8b0a04bea6441a4d5b33c6a256cd125aa1a35a913
d0e3b6ae7c5eeeb3d7a830619b3b11fb9463a1dc5808aad5926eea6fa49c7893
d0e7776bac7c4f0d6a2ba3314ffc6f430130cd3f6f3ffc4b8496b31ecc9043d
d1ca3f8fb1acd28ee6ce1227880e74fca8aa908ca373931a5180f3cb76bb8fe1
d1e9780a620ddf149c2aed319388bca7ed690c2a58c9ffc8f60b1c4515115dc9
d254745ca2edd62c5e9d3231b3131ae065b2e1759fe9916df96e6c14af59a99e
d2ca24f7ef22dfa8be6dc45ba0a64350d668bd0b9fb5a7be7d82305313d404bf
d2dd5814fcd0acd5202091e9b0392ec2e96029315580930bcb9ab38deed14d0
d4033853aaf8943cc42925d689bf1693c629faa668cdf79469d64a04a2c28250
d413e1247e22a975e7aaf93ecc4ed8affde08b7e2b72d152132c8829ab9ea3bf
d442cd3002178041faa5d9d233820b84031a3432f5384ce9e8ec8990a2e15b70

d4fc26a3fdaa990c81597fdcc4a194e600beac247d5fc5a2eb5bf5774ee2aabc
d5345c816fe716b0e2fc8f7b15eb10347859aa1e47c8e3c10f002f4809c69aef
d59072f67b73ae7663813e206bf001dbf69d86f73478879df4fee510ecc72e41
d63889a88b1546c0e457c123d946f27ffcc64105931b47f805f196eada498880
d794c27fbcf5de096750f2be32587d19cab1364b6fdd86318685574dedeb5af4
d7f4f24a8b33d5a2fab054e912827112a1f35178f1168ea1b5594d4a833389e6
d80650ed37463b35238a439658309270ab12dd0b360f1d6dbe9b3e27fa298929
d9fa47042f446bce08cddb14d6d08220016133a63d6d7a5313ba9d7a584c27ac
da26f30555398dc8af04b9e093f14fea0ad9e9a71c299a1bf5f447dd2886a5f8
da48897315da67554f90b847fcfe027d12d2c75f42ad1c413bf87d23bbd07b09
db32b295a5b8d07a8bf93cee250587b68d12e45141859c1f9a73552c572db41f
dc03ab08a186dd2b781faa3af0d4c7a8a66501a6ba6e54f5f46f42b5cff32574
dc3f64f01b34de10c06e4f6f9a00e0d6d3213e5d7d3b13df5219287d4b0ec1a8
dc7a7370db72d2ec4260c9b450e1194cb1f3979d0d21340dde9bcc123392134a
dd7311a0428c78cec1db4fbca409094ab6694db68c03aa878f9f1872ecc6e1db
dd7c962afea2752944885fdd3551a0a50fc3a58f676c1466f5fb71eda72d5a24
dd9d1598455d3782174967b6ce233018871170a80119238a96e748cb735e70fc
de080bda9659d18ec12bc86a769bc01efe1dd69dab0bfbef7e696aa56f66450f
df4eed6b429eba0c7fe96cce25b317efe39b46a443574f1d0aa9445da22f0ac2
df6e110c293a985a2138e25f3dcceb1449aa245b014a9b9b88c5490a0589a99f
e01bcad48f2c618bf090cb6e7a893ca593916f67e6497f89e92f61336a270ce9
e2eee60bbbb923cdc1916d93b9a6a2c496399f5e0e6d818d466019990370c17b
e35ec3f8d2384a5cbe85713b4230c45287036f1df924288a2ad970cae5a2391f
e3fff4738bd69ef5de3c2be90f820bd4af361ec9f3150c15a37e95fb866b4310
e44711216aa35c3dd0be03ce3787c95a6c52388f216cf4d1f2ae6089c6a9e39d
e47e898923208e15cedda56de05b9896e2e3df46628a74692d91484ab4fcecd
e6c4c251d5481c065e23d202620aa2d2d0ebc7e4d159a70c6ac9e377804da5bf
e7529a2d209e8f8405bfc92ae8f79486335989c0dcb472b678335db442fc6c04
e7af05d6e7ec4ea68d42fdcd19bbfabbb3c3e7dd2609582d87d5a9f5b387ac792
e7d9eaa267cb89c37fb9fbbb5dfd61e5b2fbc278f01dc49abb174ce7041f430a
e82920cb9b7cf1077cdddabb9b56e84c70653f836ef85ea4d4a700501ccd12a3
e8ae9d7ab270771b0aa2fbc378d11920d6804b065daaf25b5be207986b1c85c4
e8ed49650b42ebb3a251a7353809309e94f885316bf1b0153def1c180628cee4
e93bf7b3691b3cb5b9758c1ee2cf8c0537de9f921a4703c1cd6fa4df2a1aab5e
e9517186d0d7e16e56588c2aaab6db78d9dd597fea0456c49c64da82f19b17eb
e96dc80063ff83db7a421f7fc8b4ead494280a0de70d91ff27f92172ce6ef6d5
e9eea68df104e363249749346768848c4a1f08f91bed2ad48ac8ca5e1f962f86
ea43d517a6f297bacd8199f6df1f0cfaa00acac3c3c2eb125093df80c73a87a6
ea7c276788e37a7566cff0711b6a84be63f9b0878d3802881e039441b3d88a37
ecfe7056494cbc6dc0d572e3d2a7ead974f20cf5ad98bd237ad2d81a36f07cb7
ee511bade3822feb689576b65cb3e490e9459c11c5b6583b36920f742c1bd99d
eeb4e124303b10a370d6c20f70c17303dfe1cd6de4b255d85061804ad6902db0
efe299573660e349b6169b099708c6794d5b593095b9a81ac5000a9b18936252
eff390745ac05246fb469a01d7cb05409dee8f0275c809b3a7b6dfc0c58ddc7a

f02203dbc984119217b3443be3b2e1c1ae8cace333e5927acb8539d09d9c365b
f04bcd88e4fc673e3c6a50fe97b68d93a35ee6a0029b4d1082aac6751c0b751d
f0c54f3f1717c1039dacdf7c4b63200d6f027e5a622c6eb65636d2089aa4b1c0
f0da229cd56486cb27d1465410147676261d663a62aa9e95f27fda1b2ee5a662
f0ffe30cd228800ef89e93b87315c547fde5ec6e3dc8e09485b9004726bbe822
f11ff5bb01c4ad6c270b86f63b4209127d3521c98496f6bb2f021c99eaa881ba
f135ac8542776ed7b2ac2f4a95bc34ae2fc59d2710cf3d6964b57f2fa20151ad
f35f916d4ed59f3ab01d38ab67cfc4ed65ac79ef82c72298ce43083e430e8bc8
f3e1de591d92fcd9d64d221505e83d93f5639a8f154323e2d82923251d7e57b6
f424d6523a8d57a8c923f6578fc55b3be5e42d49d2f4ebf1145107effa28beb9
f42ff1b3995ca789686b11f28cd181c4a17f5afce76bcb6b6ee4c97339f5ec75
f4716cf29a2fa3ff5650ff6a4d35a26a5a534658fe7518fdf2c08554158db841
f47fae36937dabb1c97098581deb76302b10f5abdf77971626813dd158af341c
f568ee7f9b0e311bd10c08b8fc2f3ff69e603ff5a5e2c6af2d050b6de0ab1ba5
f5d4ebc19dcb1a8676ba1459a04606b5b94e3e1a02bfl1393c18e0980e8da2f8
f77a4f26ae75cf875a1e22cbab32ba379832dafc70be712dd462b5f33d129e3b
f805d4a8a759df5d85a6d3dd4a958886d7b4770c31965f8c3a76947def1dcec4
f875be79be10a88a9a5c815b0676cbfc58f48e7524f2e4d383b2d7ef63d2e306
f92ef33594e2dea236dd194b6d1c78ba5ace8702e5fa02fc48462073e8d8c010
fc41cb9d33a84a20e437091bf78e00938e4b5adb003f6c86370e975912d74dba
fc6ea5f247109ed480e04f24687519690c47a826516b64fabad31bdd190420b2
fcaf490e6ee9254f89eecbce2dd1dfa355e450c30a8648e7db43db4c5e9e434d
fc607571a9c811b0de454a772235b6d2575b9995aed6f6f772aea0b93932b070
fd8bdf9d7b2f9e99038262182f919005102ba4a513411db2d00563ee933c78b
fe0c73fc668f2c3e1c80c4085ef4599f16da5c8f5204a2c210542cd992166703
fe106c74f3eeef6d52fd27e83aea992b37896aaceb6ac5b969a9cfaf096c9d54
fe199e437de02d997427dfea4e864f386e12887c5ae16d2acc465417cd539521
fe5734f6621e30a2686219f31e5ebcca7e7851e8a572baf1463551de0d72d4ea
fec217c4b7172868e7681677436ea2b5629aa6d2c73f91b0f2a02fb357ef75e6
fed3b772542834aba36e33587e360877e9b76c186a937e39454119c757014763
ff66313758e6501562445ad51ad00a14390b5493b248e44d315d7a7041c0fc6b

Screenshots MISP Events

Events									
Source org	Member org	ID	Tags	Index	Date	Info	Distribution	Actions	
13438			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	24	2022-12-28	BitLocker Spam Run (2022-12-18 - PROFORMA TALEP)	Community		
18518			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	24	2022-12-22	BitLocker Spam Run (2022-12-12 - New Inquiry)	Community		
18523			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	24	2022-12-22	BitLocker Spam Run (2022-12-12 - Hesap hareketleri)	Community		
18778			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	24	2022-12-09	BitLocker Spam Run (2022-12-09 - Hesap hareketleri)	Community		
18827			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	18	2022-12-12	BitLocker Spam Run (2022-12-12 - Hesap hareketleri)	Community		
18812			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	26	2022-12-12	WacomRAT Spam Run (2022-12-12 - Re: New Order)	Community		
18503			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	30	2022-12-12	WacomRAT Spam Run (2022-12-12 - Re: Order confirmation reference no: FAKP562022)	Community		
18549			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	25	2022-12-14	WacomRAT Spam Run (2022-12-14 - Re: Request for offer price)	Community		
18862			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	18	2022-12-14	AgentTests Spam Run (2022-12-14 - RE: NEW PROFORMA INVOICE)	Community		
18861			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	18	2022-12-14	SkateKrygger Spam Run (2022-12-14 - Re: Question...)	Community		
18869			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	15	2022-12-14	AgentTests Spam Run (2022-12-14 - NEW SHIPMENT)	Community		
18859			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	18	2022-12-14	SkateKrygger Spam Run (2022-12-14 - RETURN PAYMENT TT)	Community		
18857			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	14	2022-12-14	AgentTests Spam Run (2022-12-14 - Notice of Arrival for ASB # 183521408)	Community		
18856			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	16	2022-12-14	AgentTests Spam Run (2022-12-14 - RE: Shipping documents attached ISO Certificate)	Community		
18855			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	18	2022-12-14	AgentTests Spam Run (2022-12-14 - RV: OFFERTA POW 9008862)	Community		
18853			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	14	2022-12-14	AgentTests Spam Run (2022-12-14 - SOA Nov 2022 (52512,28,23))	Community		
18851			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	14	2022-12-14	AgentTests Spam Run (2022-12-14 - RE: DRAFT MAIL KISAKAZI2389 F ETOS 12 DEC (SHIPMENT DOCUMENTS)	Community		
18842			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	14	2022-12-13	AgentTests Spam Run (2022-12-13 - Re: inquiry/advance product)	Community		
18841			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	17	2022-12-13	NanoCore Spam Run (2022-12-13 - Payment Advice - Advice Ref[242036]nd - ACH credits / Customer Ref[105817130] / Second Party Ref[2304303088])	Community		
18840			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	18	2022-12-13	SkateKrygger Spam Run (2022-12-13 - PINE-ALERT[2022]CORRECTION[SHIPING] KTN - CSK / M- 888-8388 3422 / H- 05022212851)	Community		
18839			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	14	2022-12-13	AgentTests Spam Run (2022-12-13 - BSB - PO 180720228 - NAKULAB)	Community		
18838			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	18	2022-12-13	SkateKrygger Spam Run (2022-12-13 - DALCANDRO SHIPPING CO., LTD.)	Community		
18837			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	15	2022-12-13	AgentTests Spam Run (2022-12-13 - NEW ORDER)	Community		
18836			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	15	2022-12-13	AgentTests Spam Run (2022-12-13 - RE: SHIPPING DOC (51,COO,PL,BL))	Community		
18835			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	18	2022-12-13	SkateKrygger Spam Run (2022-12-13 - INWIDE BLUE KOREA[5] SOA OCT 2022)	Community		
18834			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	18	2022-12-13	SkateKrygger Spam Run (2022-12-13 - RE: [URGENT] invoices last shipment)	Community		
18833			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	18	2022-12-12	SkateKrygger Spam Run (2022-12-12 - RE: PI & Packing list)	Community		
18829			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	17	2022-12-12	NanoCore Spam Run (2022-12-12 - RE: Transfer Confirmation)	Community		
18828			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	15	2022-12-12	AgentTests Spam Run (2022-12-12 - 05 05 05 P22210-PRC228 BOULE SCALLOP MEAT CHN - 2278520141 / P22210-PRC228 BOULE SCALLOP MEAT CHN - 2278520141)	Community		
18826			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	14	2022-12-12	BitLocker Spam Run (2022-12-12 - Request for quotation)	Community		
18825			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	15	2022-12-12	AgentTests Spam Run (2022-12-12 - RE: SHIPPING DOC (51,COO,PL,BL))	Community		
18824			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	14	2022-12-12	AgentTests Spam Run (2022-12-12 - NEW ORDER #308078610)	Community		
18823			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	23	2022-12-12	AgentTests Spam Run (2022-12-12 - Pro Forma Invoice / BL & NAW 28767 / Certified Processors / 88817455 / DASM 3002)	Community		
18822			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	14	2022-12-12	AgentTests Spam Run (2022-12-12 - PO 3785)	Community		
18821			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	15	2022-12-12	AgentTests Spam Run (2022-12-12 - RE: SHIPPING DOC (51,COO,PL,BL))	Community		
18820			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	22	2022-12-12	AgentTests Spam Run (2022-12-12 - Order description...)	Community		
18819			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	31	2022-12-12	Libble Spam Run (2022-12-12 - FVW: Estimation for discharging crude ETA December 2022 (possible CTM Delivery)	Community		
18818			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	24	2022-12-12	NanoCore Spam Run (2022-12-12 - Contract Agreement Draft Copy)	Community		
18817			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	22	2022-12-12	DarkCloud Spam Run (2022-12-12 - Updated Payment List)	Community		
18816			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	23	2022-12-12	AgentTests Spam Run (2022-12-12 - Re: LSA Agreement)	Community		
18815			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	31	2022-12-12	Libble Spam Run (2022-12-12 - SOA 08.12.22)	Community		
18814			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	31	2022-12-12	Libble Spam Run (2022-12-12 - RE: SWIFT_COPY(175486558))	Community		
18813			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	22	2022-12-12	AgentTests Spam Run (2022-12-12 - RE: STATEMENT OF ACCOUNT)	Community		
18812			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	31	2022-12-12	Libble Spam Run (2022-12-12 - RE: Purchase Inquiry / KPCPU / 2204203088 / 05-12)	Community		
18810			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	23	2022-12-12	AgentTests Spam Run (2022-12-12 - RE: SHIPPING DOC (51,COO,PL,BL))	Community		
18808			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	23	2022-12-12	AgentTests Spam Run (2022-12-12 - RE: SHIPMENT DOCUMENT)	Community		
18807			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	22	2022-12-12	AgentTests Spam Run (2022-12-12 - RE: HOLIDAY BOOKING)	Community		
18806			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	27	2022-12-12	SkateKrygger Spam Run (2022-12-12 - RE: NEW ENQUIRY)	Community		
18797			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	24	2022-12-09	AgentTests Spam Run (2022-12-09 - PO 74389-8554)	Community		
18796			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	13	2022-12-09	DarkCloud Spam Run (2022-12-09 - Para Transfer Swift Mensaj)	Community		
18795			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	31	2022-12-09	Libble Spam Run (2022-12-09 - RE: Remittance Copy)	Community		
18794			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	25	2022-12-09	SkateKrygger Spam Run (2022-12-09 - Payment Advice - Advice Ref[242036]nd - Priority payment / Customer Ref[4F0001188575])	Community		
18793			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	22	2022-12-09	AgentTests Spam Run (2022-12-09 - RE: ELEMENT ORDER (8208 0204081 P))	Community		
18792			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	23	2022-12-09	AgentTests Spam Run (2022-12-09 - RE: PAYMENT OF USD323,517)	Community		
18791			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	22	2022-12-09	AgentTests Spam Run (2022-12-09 - RE: SWIFT CONFIRMATION COPY (1882 TT450227145022))	Community		
18790			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	31	2022-12-09	Libble Spam Run (2022-12-09 - SOA 08.12.22)	Community		
18788			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	23	2022-12-09	AgentTests Spam Run (2022-12-09 - FVW: payment swift)	Community		
18787			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	22	2022-12-09	BitLocker Spam Run (2022-12-09 - Demande de cotation (A-price quote))	Community		
18786			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	23	2022-12-09	AgentTests Spam Run (2022-12-09 - For Confirm as Booking)	Community		
18785			sp:grain, sp:malware, sp:crypto, sp:malware, sp:crypto	23	2022-12-09	AgentTests Spam Run (2022-12-09 - Re: INVENTORY)	Community		

Author Name, email@address

Home	Events	Dashboard	Searches	Alerts	Alerts Filter	Global Alerts	System Alerts	Administration	Logout	Help
Events										
x previous 1 2 3 4 5 6 next x										
Filter: Tag: none No Events Org: Events										
☐	☐	Source	Tag	Info	Alerts	Date	Info	Details	Actions	
☐	☒	W 180106	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-05	AgentTecla Span Run (2022-12-05 - Product Inquiry 1-ATLAS SHIPPING SUPPLY)	Community	☒ ☒ ☒		
☐	☒	W 180105	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-05	AgentTecla Span Run (2022-12-05 - MY HONG KONG SPIRIT pre-arrival Documents)	Community	☒ ☒ ☒		
☐	☒	W 180104	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	24	2022-12-05	SnokeKeylogger Span Run (2022-12-05 - RE [URGENT] DHL Express)	Community	☒ ☒ ☒		
☐	☒	W 180103	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-05	AgentTecla Span Run (2022-12-05 - Statement of the account)	Community	☒ ☒ ☒		
☐	☒	W 180102	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	24	2022-12-05	RemcoRAT Span Run (2022-12-05 - Purchase Order A030315)	Community	☒ ☒ ☒		
☐	☒	W 180101	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-05	AgentTecla Span Run (2022-12-05 - New question 28131002)	Community	☒ ☒ ☒		
☐	☒	W 180100	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-05	AgentTecla Span Run (2022-12-05 - RE SHIPMENT DOCUMENT)	Community	☒ ☒ ☒		
☐	☒	W 180109	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-05	AgentTecla Span Run (2022-12-05 - RE FDA HHS)	Community	☒ ☒ ☒		
☐	☒	W 180108	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	24	2022-12-05	AgentTecla Span Run (2022-12-05 - payment swift details)	Community	☒ ☒ ☒		
☐	☒	W 180107	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-05	AgentTecla Span Run (2022-12-05 - RPQ_BP0400 QWQC)	Community	☒ ☒ ☒		
☐	☒	W 180106	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-05	AgentTecla Span Run (2022-12-05 - RE SHIPPING DOC (COCOCPL BL))	Community	☒ ☒ ☒		
☐	☒	W 180105	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-05	AgentTecla Span Run (2022-12-05 - FW Your latest DHL invoice - K10900071498)	Community	☒ ☒ ☒		
☐	☒	W 180104	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-05	AgentTecla Span Run (2022-12-05 - Message - Annual Notice ready for Bill of Lading 209500072)	Community	☒ ☒ ☒		
☐	☒	W 180103	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	24	2022-12-05	SnokeKeylogger Span Run (2022-12-05 - RE [URGENT] Payment Confirmation SwiftRef000000010564)	Community	☒ ☒ ☒		
☐	☒	W 180102	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	24	2022-12-05	SnokeKeylogger Span Run (2022-12-05 - AMOSTRAS PICOTACAO - OT03AL)	Community	☒ ☒ ☒		
☐	☒	W 180101	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	31	2022-12-05	Lolbox Span Run (2022-12-05 - RE RE RE Shipping Documents)	Community	☒ ☒ ☒		
☐	☒	W 180100	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	7	2022-12-05	AgentTecla Span Run (2022-12-05 - RE NEW ORDER NEEDED)	Community	☒ ☒ ☒		
☐	☒	W 180179	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-05	AgentTecla Span Run (2022-12-05 - 2ND REMITTANCE COPY)	Community	☒ ☒ ☒		
☐	☒	W 180178	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-05	AgentTecla Span Run (2022-12-05 - FW NEW ORDER FOR PLUM 211213-01AQR20070)	Community	☒ ☒ ☒		
☐	☒	W 180177	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-05	AgentTecla Span Run (2022-12-05 - RE PAYMENT IS COMPLETED)	Community	☒ ☒ ☒		
☐	☒	W 180168	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-02	AgentTecla Span Run (2022-12-02 - [URGENT] Invoices test shipment)	Community	☒ ☒ ☒		
☐	☒	W 180167	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-02	AgentTecla Span Run (2022-12-02 - RE SHIPMENT DOCUMENT)	Community	☒ ☒ ☒		
☐	☒	W 180166	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	24	2022-12-02	SnokeKeylogger Span Run (2022-12-02 - DOCUMENT)	Community	☒ ☒ ☒		
☐	☒	W 180165	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-02	AgentTecla Span Run (2022-12-02 - Shipping Documents - Commercial Invoice & Packing List)	Community	☒ ☒ ☒		
☐	☒	W 180164	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-02	AgentTecla Span Run (2022-12-02 - 12.1 Beijing BOE display technology customer declaration information - B4220800)	Community	☒ ☒ ☒		
☐	☒	W 180162	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-02	AgentTecla Span Run (2022-12-02 - Enclosed Swift Details)	Community	☒ ☒ ☒		
☐	☒	W 180159	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-01	AgentTecla Span Run (2022-12-01 - PURCHASE ORDER # 12076008 & 12076005)	Community	☒ ☒ ☒		
☐	☒	W 180158	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-01	AgentTecla Span Run (2022-12-01 - PRE-ALERT)	Community	☒ ☒ ☒		
☐	☒	W 180156	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	24	2022-12-01	AgentTecla Span Run (2022-12-01 - Notification: DOW VESSEL - Chenbrans Uranus - 20-224 Nephtha - loading on Dec 9-10, 2022)	Community	☒ ☒ ☒		
☐	☒	W 180155	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-01	AgentTecla Span Run (2022-12-01 - RE: HBL and shipper's documents attached)	Community	☒ ☒ ☒		
☐	☒	W 180154	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-01	AgentTecla Span Run (2022-12-01 - SORNDZ1200004 / LCL / SAVOUR & SPICE / FROM NZ TO SR)	Community	☒ ☒ ☒		
☐	☒	W 180153	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	25	2022-12-01	AgentTecla Span Run (2022-12-01 - Request For Machine Quotation)	Community	☒ ☒ ☒		
☐	☒	W 180152	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-01	AgentTecla Span Run (2022-12-01 - RE: Agriculture Centre of Excellence (Shipped by Air to Singapore Airport, Singapore) Ref. PO No.01_QT)	Community	☒ ☒ ☒		
☐	☒	W 180151	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-01	AgentTecla Span Run (2022-12-01 - Payment Advice - Advice Ref(147430454) / ACH credits / Customer Ref(1090017230) / Second Party Ref(120441001806)	Community	☒ ☒ ☒		
☐	☒	W 180150	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-12-01	WIG Order	Community	☒ ☒ ☒		
☐	☒	W 180149	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-01	AgentTecla Span Run (2022-12-01 - Thruage)	Community	☒ ☒ ☒		
☐	☒	W 180148	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-01	AgentTecla Span Run (2022-12-01 - DHL Shipping Document/Invoice Receipt)	Community	☒ ☒ ☒		
☐	☒	W 180147	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-12-01	AgentTecla Span Run (2022-12-01 - Pre-Payment)	Community	☒ ☒ ☒		
☐	☒	W 180110	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - quotation Request)	Community	☒ ☒ ☒		
☐	☒	W 180109	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	24	2022-11-25	SnokeKeylogger Span Run (2022-11-25 - SR1210022 TARIKH MESABA GELLEN FAST BILGILENDIRME FORMU)	Community	☒ ☒ ☒		
☐	☒	W 180108	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	24	2022-11-25	SnokeKeylogger Span Run (2022-11-25 - DHL Shipping Notice: AHB 687862684)	Community	☒ ☒ ☒		
☐	☒	W 180106	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - Amended Purchase Order)	Community	☒ ☒ ☒		
☐	☒	W 180105	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - RE SHIPPING DOCUMENT & PACKING LIST)	Community	☒ ☒ ☒		
☐	☒	W 180104	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - RE: WRONG SHIPMENT RETURNED)	Community	☒ ☒ ☒		
☐	☒	W 180103	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - RE: RPQ Machine Quotation)	Community	☒ ☒ ☒		
☐	☒	W 180102	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - RE: Request for Quotation - Urgent 650202)	Community	☒ ☒ ☒		
☐	☒	W 180101	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - PO 3642_PRR 10782_HYDROLIC CYL REPAIRING) GLOBAL CLEARING)	Community	☒ ☒ ☒		
☐	☒	W 180100	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - Re PURCHASE ORDER 102087 - INQUIRY FOR PT & RT SEAL NOV 2022 (DSP & GSD DEF SEAL))	Community	☒ ☒ ☒		
☐	☒	W 180109	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-11-25	AgentTecla Span Run (2022-11-25 - RE RPQ_FOB China port (NOVIDEC))	Community	☒ ☒ ☒		
☐	☒	W 180108	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - Due Payment Follow UP)	Community	☒ ☒ ☒		
☐	☒	W 180107	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - NEW SALES ORDER FOR YOUR REFERENCE)	Community	☒ ☒ ☒		
☐	☒	W 180106	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-11-25	AgentTecla Span Run (2022-11-25 - New Order #987)	Community	☒ ☒ ☒		
☐	☒	W 180105	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - PRE ALERT 160-4839836/4502788)	Community	☒ ☒ ☒		
☐	☒	W 180104	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-11-25	AgentTecla Span Run (2022-11-25 - RE SHIPPING DOCUMENT & PACKING LIST)	Community	☒ ☒ ☒		
☐	☒	W 180103	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - RPQ1103022 SOBEC)	Community	☒ ☒ ☒		
☐	☒	W 180102	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	24	2022-11-25	AgentTecla Span Run (2022-11-25 - Re: Revised Document For Payment)	Community	☒ ☒ ☒		
☐	☒	W 180101	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - RE: CI & PL Confirmation for the Order)	Community	☒ ☒ ☒		
☐	☒	W 180100	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	22	2022-11-25	AgentTecla Span Run (2022-11-25 - RE: AW Enclosed our payment details)	Community	☒ ☒ ☒		
☐	☒	W 180109	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	31	2022-11-25	Lolbox Span Run (2022-11-25 - RE: Payment SwiftRef000000010564)	Community	☒ ☒ ☒		
☐	☒	W 180108	☒ AgentTecla ☒ AgentTecla ☒ AgentTecla	23	2022-11-25	AgentTecla Span Run (2022-11-25 - RE URGENT PURCHASE ORDER)	Community	☒ ☒ ☒		
Page 3 of 6, showing 60 records out of 330 total, starting on record 121, ending on 180										
x previous 1 2 3 4 5 6 next x										

line | 71

© 2023 The SANS Institute
<https://t.me/learningnets>

Author retains full rights.

Suricata Enabled Ruleset

Rule name	Reference
3coresec.rules	(ET Rules, 2022)
botcc.portgrouped.rules	(ET Rules, 2022)
botcc.rules	(ET Rules, 2022)
ciarmy.rules	(ET Rules, 2022)
compromised.rules	(ET Rules, 2022)
drop.rules	(ET Rules, 2022)
dshield.rules	(ET Rules, 2022)
emerging-activex.rules	(ET Rules, 2022)
emerging-adware_pup.rules	(ET Rules, 2022)
emerging-attack_response.rules	(ET Rules, 2022)
emerging-chat.rules	(ET Rules, 2022)
emerging-coinminer.rules	(ET Rules, 2022)
emerging-current_events.rules	(ET Rules, 2022)
emerging-deleted.rules	(ET Rules, 2022)
emerging-dns.rules	(ET Rules, 2022)
emerging-dos.rules	(ET Rules, 2022)
emerging-exploit_kit.rules	(ET Rules, 2022)
emerging-exploit.rules	(ET Rules, 2022)
emerging-ftp.rules	(ET Rules, 2022)
emerging-games.rules	(ET Rules, 2022)
emerging-hunting.rules	(ET Rules, 2022)
emerging-icmp_info.rules	(ET Rules, 2022)
emerging-icmp.rules	(ET Rules, 2022)
emerging-imap.rules	(ET Rules, 2022)
emerging-inappropriate.rules	(ET Rules, 2022)
emerging-info.rules	(ET Rules, 2022)
emerging-ja3.rules	(ET Rules, 2022)
emerging-malware.rules	(ET Rules, 2022)
emerging-misc.rules	(ET Rules, 2022)
emerging-mobile_malware.rules	(ET Rules, 2022)
emerging-netbios.rules	(ET Rules, 2022)
emerging-p2p.rules	(ET Rules, 2022)
emerging-phishing.rules	(ET Rules, 2022)
emerging-policy.rules	(ET Rules, 2022)
emerging-pop3.rules	(ET Rules, 2022)
emerging-rpc.rules	(ET Rules, 2022)
emerging-scada.rules	(ET Rules, 2022)
emerging-scan.rules	(ET Rules, 2022)
emerging-shellcode.rules	(ET Rules, 2022)
emerging-smtp.rules	(ET Rules, 2022)
emerging-snmp.rules	(ET Rules, 2022)

emerging-sql.rules	(ET Rules, 2022)
emerging-telnet.rules	(ET Rules, 2022)
emerging-tftp.rules	(ET Rules, 2022)
emerging-user_agents.rules	(ET Rules, 2022)
emerging-voip.rules	(ET Rules, 2022)
emerging-web_client.rules	(ET Rules, 2022)
emerging-web_server.rules	(ET Rules, 2022)
emerging-web_specific_apps.rules	(ET Rules, 2022)
emerging-worm.rules	(ET Rules, 2022)
feodotracker.rules	(Feodo Tracker, 2022)
sslblacklist.rules	(SSL blacklist, 2022)
sslipblacklist.rules	(SSL blacklist, 2022)
threatview_CS_c2.rules	(ET Rules, 2022)
tor.rules	(ET Rules, 2022)
urlhaus.rules	(URLhaus, 2022)

Python Code

main.py

```
#!/usr/bin/python3

from plugins.abstract_plugin import AbstractPlugin
from utils.state import State
from utils.task import Task

class Pipeline():
    def __init__(self, ):
        print('Init Main')
        self.plugins = {}
        self.redis = None
        self.task_id = 0
        self._load_environment()

    def _load_environment(self) -> None:
        self._load_plugins()

    def _load_plugins(self) -> None:
        self.plugins = AbstractPlugin.load_plugins()

    def run_plugins(self) -> None:
        self.task_id += 1
        State.attachment_name = 'Unknown'
        task = Task(self.task_id, len(self.plugins))
        for _, plugin in self.plugins.items():
            try:
                task.adavance_progress()
                plugin.run({})
            except Exception as e:
                print(e)
                task.set_error()
                break

if __name__ == '__main__':
    app = Pipeline()
    while True:
        app.run_plugins()
```

state.py

```
import os
import pathlib

from typing import Optional
from dataclasses import dataclass
from configparser import ConfigParser
from pymisp import MISPEvent, ExpandedPyMISP, MISPTag
from typing import List

class State:
    config: ConfigParser
    path: Optional[pathlib.Path]
    attachment_name: Optional[str]
    misp: ExpandedPyMISP
    tags: List[MISPTag]
    event: MISPEvent
```

```

@staticmethod
def init() -> None:
    root_path = os.getcwd()
    cfg_file = os.path.join(root_path, 'config.cfg')
    State._init_config(cfg_file)
    State._init_misp()
    State._init_tags()

@staticmethod
def _init_config(file) -> None:

    if not os.path.isfile(file):
        exit(1)

    State.config = ConfigParser()
    State.config.read(file)
    State.path = None

@staticmethod
def _init_misp() -> None:
    misp_config = State.config['misp']
    verifycert = misp_config['verifycert']
    if verifycert and verifycert.lower() == 'false':
        verifycert = False
    if verifycert and verifycert.lower() == 'true':
        verifycert = True
    State.misp = ExpandedPyMISP(misp_config["url"],
misp_config["key"], verifycert)

@staticmethod
def _init_tags() -> None:
    tags = State.misp.tags(pythonify=True)
    State.tags = {tag["name"]: tag["id"] for tag in tags}

State.init()

```

task.py

```

import json
import redis

from utils.state import State

class Task:

    def __init__(self, task_id, plugins_count):
        self.task_id = task_id
        self.progress = 0
        self.plugins_count = plugins_count
        self.error = False
        self._update_task()

    @staticmethod
    def _load_redis() -> None:
        Task.redis = redis.Redis(host=State.config['redis']['host'],
port=State.config['redis']['port'], db=0)
        Task.redis.flushall()

    def advance_progress(self):
        self.progress += 1

```

```

        self._update_task()

    def set_error(self):
        self.error = True
        self._update_task()

    def _update_task(self):
        data = json.dumps({"task_id": self.task_id,
                           "sample_name": State.attachment_name,
                           "progress": self.progress,
                           "total_progress": self.plugins_count,
                           "error": self.error})
        Task.redis.set(str(self.task_id), data)

Task._load_redis()

```

abstract_plugin.py

```

from __future__ import annotations
import pkgutil
import importlib
import plugins

from argparse import ArgumentParser
from ast import Or
from itertools import chain
from utils.state import State
from collections import OrderedDict

class AbstractPlugin:

    def __init__(self):
        pass

    def run(self, args: dict) -> None:
        raise NotImplementedError("Each plugin needs to implement this")

    def load_argparser(self, parser: ArgumentParser) -> None:
        raise NotImplementedError("Each plugin needs to implement this")

    @staticmethod
    def load_plugins() -> dict:

        discovered_plugins = {
            name: importlib.import_module(name)
            for finder, name, ispkg,
            in pkgutil.iter_modules(plugins.__path__,
                                    plugins.__name__ + ".")
        }

        plugins_to_load = State.config['plugins']['run_order']
        plugins_to_load = plugins_to_load.split(',')
        plugins_to_load = [plugin.strip() for plugin in
                           plugins_to_load]
        plugins_dict = OrderedDict().fromkeys(plugins_to_load)

        for plugin_name, plugin in discovered_plugins.items():
            __, call_name = plugin_name.rsplit('.', 1)

```

```

        if call_name not in plugins_to_load:
            continue
        importlib.import_module(plugin_name)
        instance = getattr(plugin, plugin.class_name)()
        plugins_dict[call_name] = instance
    return plugins_dict

```

email.py

```

import datetime
import os
import json
import re
import eml_parser
import shutil
import pathlib

from hashlib import new
from base64 import b64decode
from plugins.abstract_plugin import AbstractPlugin
from utils.state import State

class_name = 'Email'

class Email(AbstractPlugin):

    def __init__(self):
        super().__init__()
        self.data = {}

    # Run Email plugin
    def run(self, args):
        self.new_emails_dir = State.config['emails']['new_emails_dir']
        self.analysis_dir = State.config['emails']['analysis_dir']
        self.already_analyzed_samples = State.config['collection']['analyzed_samples']

        self.oldest_email_full_path = self.parse_new_email()
        print(self.oldest_email_full_path)
        if self.oldest_email_full_path:
            new_analysis_dir = self.create_analysis_dir()
            self.extract_attachments(new_analysis_dir)
            State.path = pathlib.Path(new_analysis_dir)
            with open(new_analysis_dir + 'mail.json', 'w') as f:
                json.dump(self.data, f)

    # Extract attachments from new email
    def extract_attachments(self, new_analysis_dir):
        ep = eml_parser.EmlParser(include_attachment_data=True)
        attachments = ep.decode_email(new_analysis_dir + 'mail.eml')
        for attachment in attachments['attachment']:
            with open(new_analysis_dir + attachment['filename'], 'wb')
as f:
                f.write(b64decode(attachment['raw']))
                State.attachment_name = attachment['filename']
        return

    # Create new analysis dir
    def create_analysis_dir(self):
        dir_today = datetime.datetime.now().strftime("%Y") + '/' + \
            datetime.datetime.now().strftime("%m") + '/' + \

```

```

        datetime.datetime.now().strftime("%d") + '/'
    if not os.path.exists(self.analysis_dir + dir_today):
        os.makedirs(self.analysis_dir + dir_today)
    list_dir_today = os.listdir(self.analysis_dir + dir_today)
    if list_dir_today:
        sorted_list_dir_today = sorted(list_dir_today, key=int)
        last_dir = sorted_list_dir_today[-1]
        new_dir = int(last_dir) + 1
        new_analysis_dir = self.analysis_dir + dir_today +
str(new_dir) + '/'
        os.mkdir(new_analysis_dir)
        shutil.move(self.oldest_email_full_path, new_analysis_dir +
'mail.eml')
        return new_analysis_dir
    else:
        new_analysis_dir = self.analysis_dir + dir_today + '1' +
        '/'
        os.mkdir(new_analysis_dir)
        shutil.move(self.oldest_email_full_path, new_analysis_dir +
'mail.eml')
        return new_analysis_dir

# Parse new email
def parse_new_email(self):
    check_if_new_emails = os.listdir(self.new_emails_dir)
    if not check_if_new_emails:
        return
    os.chdir(self.new_emails_dir)
    oldest_email = min(check_if_new_emails, key=os.path.getctime)
    oldest_email_full_path = self.new_emails_dir + oldest_email
    with open(oldest_email_full_path, 'rb') as fhdl:
        print(oldest_email_full_path)
        raw_email = fhdl.read()
        ep = eml_parser.EmlParser()
        parsed_eml = ep.decode_email_bytes(raw_email)

# Extract email header information to use it in the MISP
plugin
        self.data['email-subject'] =
parsed_eml['header']['subject']
        self.data['email-src-display-name'] =
parsed_eml['header']['from']
        ipv4_regex = r"\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"
        ip_match = list(filter(lambda x: re.match(ipv4_regex, x),
parsed_eml['header']['received'][-1]['from']))
        self.data['ip-src'] = ip_match[0]
        print(self.data)

# check if there is an attachment
if parsed_eml.get('attachment'):
    for filename in parsed_eml['attachment']:
        self.data['email-attachment'] =
filename['filename']
        print(filename['filename'])
        print(filename['hash'])
        print(filename['hash']['md5'])
        print(filename['hash']['sha256'])
        if not
os.path.isfile(self.already_analyzed_samples):
            open(self.already_analyzed_samples, 'w')

```

```

        with open (self.already_analyzed_samples, 'r') as
collection:
    if
os.stat(self.already_analyzed_samples).st_size == 0:
        data = []
    else:
        data = json.load(collection)

    if any(d['md5'] == filename['hash']['md5'] for
d in data):
        print('Already analyzed')
        State.attachment_name =
filename['filename']
        os.remove(oldest_email_full_path)
        raise Exception('Already analyzed')
        collection.close()
        return

    else:
        print('not analyzed yet')
        new_file_entry = {
            'md5':
filename['hash']['md5'],
            'sha1':
filename['hash']['sha1'],
            'sha256':
filename['hash']['sha256'],
            'sha512':
filename['hash']['sha512']
        }

        data.append(new_file_entry)
        with open(self.already_analyzed_samples,
'w') as new_collection:
            json.dump(data, new_collection,
indent=4)

        collection.close()
        new_collection.close()
        return oldest_email_full_path

    else:
        print('no attachment')
        os.remove(oldest_email_full_path)
        return

```

unpack.py

```

import os
import subprocess

from plugins.abstract_plugin import AbstractPlugin
from utils.state import State
from pathlib import Path

class_name = 'Unpack'

class Unpack(AbstractPlugin):

    def __init__(self):
        super().__init__()

```

```

        self.archive_list_7z = ['application/zip', 'application/x-tar',
                                'application/x-xz', 'application/x-iso9660-image', 'application/x-7z-
                                compressed']
        self.archive_list_unrar = ['application/x-rar']
        self.todo_path = ''

    def run(self, args):
        print('Run Unpack Plugin')
        self.todo_path = str(State.path)
        if not State.path:
            print('No path to unpack a file')
            raise Exception('No path to unpack a file')

        for file in os.listdir(State.path):
            archive_tool = self.check_archive(file)
            State.attachment_name = file
            if archive_tool:
                self.unpack_archive(file, archive_tool)
            return

        # Check if file is an archive
        def check_archive(self, file):
            archive_file_mime = subprocess.run(['file', '--mime-type',
            self.todo_path + '/' + file], stdout=subprocess.PIPE)
            print(archive_file_mime)
            for archive_type in self.archive_list_7z:
                if archive_type in archive_file_mime.stdout.decode('utf-
8'):
                    return '7z'
            for archive_type in self.archive_list_unrar:
                if archive_type in archive_file_mime.stdout.decode('utf-
8'):
                    return 'unrar'

            return False

        def unpack_archive(self, file, archive_tool):
            if archive_tool == '7z':
                subprocess.run(['7z', 'x', '-y', self.todo_path + '/' +
            file, '-o' + self.todo_path + '/'])
            elif archive_tool == 'unrar':
                subprocess.run(['unrar', 'e', '-y', self.todo_path + '/' +
            file, self.todo_path + '/'])
            else:
                print('No archive tool found')
                raise Exception('No archive tool found')

```

trriage.py

```

import json
import os
import time
import re
import subprocess
import requests

from plugins.abstract_plugin import AbstractPlugin
from utils.state import State
from pathlib import Path

class_name = 'Triage'

```

```

class Triage(AbstractPlugin):

    def __init__(self):
        super().__init__()
        self.data = None
        self.triage_path = ''
        self.triage_url = State.config['triage']['url']
        self.type_map = {
            'domain': 'domain',
            'hostname': 'hostname',
            'dst': 'ip-dst|port',
            'dumped_file': 'filename',
            'family': 'tag',
            'md5': 'md5',
            'sha1': 'sha1',
            'sha256': 'sha256',
            'sha512': 'sha512',
            'target': 'email-attachment',
            'url': 'url',
            'tls_ja3': 'ja3-fingerprint-md5',
            'user-agent': 'user-agent',
            'username': 'email-dst',
            'email_to': 'email-dst'
        }

    # Run Triage
    def run(self, args):
        print('Run Triage')
        todo_path = str(State.path)
        if todo_path:
            print(todo_path)
            # create directory for triage output
            if not os.path.exists(todo_path + '/triage'):
                os.makedirs(todo_path + '/triage')
                self.triage_path = todo_path + '/triage/'

        else:
            print('No samples')
            return

        for file in os.listdir(todo_path):
            if self.check_executable(todo_path, file):
                State.attachment_name = file
                self.run_triage(todo_path, file)

        triage_report = self.triage_path + 'triage_report.json'
        if os.path.exists(triage_report):
            with open(triage_report, "r", encoding='utf-8') as f:
                triage_report_data = f.read()

            self.data = json.loads(triage_report_data)
            self.get_signature_info()
            self.get_c2_info()

        else:
            print('No triage report')
            return

    # Check if file is executable
    def check_executable(self, todo_path, file):

```

```

mime_type_executable = subprocess.run(['file', '--mime-type',
todo_path + '/' + file], stdout=subprocess.PIPE)
print(mime_type_executable)
if 'application/x-dosexec' in
mime_type_executable.stdout.decode('utf-8'):
    return True
return False

# Upload file to Triage and get report/dump/pcap
def run_triage(self, todo_path, file):
    auth_bearer = 'Bearer ' + State.config['triage']['api_key']
    headers = {
        'Authorization': auth_bearer
    }

    files = {
        'file': open(todo_path + '/' + file, 'rb'),
        '_json': (None,
'{"kind": "file", "interactive": false, "profiles": []}'),
    }

    # Upload file to Triage
    task_id = requests.post(self.triage_url, headers=headers,
files=files)
    task_id = task_id.json()
    task_id = task_id['id']
    print(task_id)
    triage_url = self.triage_url + '/' + task_id + '/behavioral2/'

    # Check if Triage is done
    time.sleep(150)
    check_status = requests.get(triage_url + 'report_triage.json',
headers=headers)
    while check_status.status_code != 200:
        time.sleep(10)
        check_status = requests.get(triage_url +
'report_triage.json', headers=headers)
        print (check_status.status_code)

    # Get Triage JSON report
    report_triage = requests.get(triage_url + 'report_triage.json',
headers=headers)
    save_report = open(self.triage_path + 'triage_report.json',
'w')
    save_report.write(report_triage.text)
    save_report.close()

    # Get Triage PCAP
    pcap_triage = requests.get(triage_url + 'dump.pcapng',
headers=headers)
    save_pcap = open(self.triage_path + 'suricata.pcap', 'wb')
    save_pcap.write(pcap_triage.content)
    save_pcap.close()

    # Load Triage JSON Report
    with open(self.triage_path + 'triage_report.json') as f:
        data = json.load(f)
        if not os.path.exists(self.triage_path + 'files'):
            os.makedirs(self.triage_path + 'files')
        if not os.path.exists(self.triage_path + 'memory'):
            os.makedirs(self.triage_path + 'memory')
        # get all memory dumps and files

```

```

        for i in data['dumped']:
            # get memory dump and files
            dump = requests.get(triage_url + i['name'],
                                headers=headers)

            # save memory dump or files
            save_dump = open(self.triage_path + i['name'], 'wb')
            save_dump.write(dump.content)
            save_dump.close()

# Add C2 info to JSON file
def add_attr(self, field, field_val):
    print('add_attr')
    if not field_val:
        return
    misp_type = self.type_map.get(field)
    # Add attribute to C2 JSON file
    c2_file_name = self.triage_path + 'c2.json'
    if os.path.exists(c2_file_name):
        with open(c2_file_name, "r", encoding='utf-8') as f:
            c2_data = f.read()
            c2_data = json.load(c2_data)
    else:
        c2_data = {}

    if misp_type == 'domain' and field_val.count('.') > 1:
        misp_type = 'hostname'

    with open(c2_file_name, "w", encoding='utf-8') as f:
        print(c2_data)
        if not c2_data:
            print('c2_data is empty')
            c2_data = {misp_type: field_val}
        else:
            c2_data.update({misp_type: field_val})
        json.dump(c2_data, f)

# Write signature to JSON file
def add_signature(self, field, field_val):
    signature_file_name = self.triage_path + 'signature.json'

    if os.path.exists(signature_file_name):
        with open(signature_file_name, "r") as f:
            signature_data = f.read()
            signature_data = json.loads(signature_data)
    else:
        signature_data = []

    with open(signature_file_name, "w") as f:
        misp_type = self.type_map.get(field)
        if misp_type == 'tag':
            tag = self.search_tag(field_val)

            if tag:
                signature_data.append(tag)
                json.dump(signature_data, f)

# Get MISP Tags
def search_tag(self, tag):
    # Ignore the misp-galaxy tags
    available_tags = filter(lambda x: not x.startswith('misp-'),
                             State.tags)

```

```

        for misp_tag in available_tags:
            if tag.lower() in misp_tag.lower():
                return misp_tag

    # Get Malware Signature from Triage JSON report
    def get_signature_info(self):
        analysis = self.data.get('analysis', {})
        tags = analysis.get('tags', [])
        for tag in tags:
            if tag.startswith('family'):
                self.add_signature('family', tag.split(':')[1])

    # Get C2 credentials from Triage JSON report
    def get_c2_credentials(self, data):
        print('get_c2_credentials')
        credentials = data.get('credentials', [])
        if not isinstance(credentials, list):
            credentials = [credentials]
        for cred in credentials:
            proto = cred.get('protocol', '')
            proto_comment = 'FTP' if proto == 'ftp' else 'Email'

            username = cred.get('username')
            if username:
                self.add_attr('username', username)

            email_to = cred.get('email_to')
            if email_to:
                self.add_attr('email_to', email_to)

            host = cred.get('host')
            if host:
                attr_type = 'url' if re.match(r'(http|https|ftp)://',
host) else 'hostname'
                self.add_attr(attr_type, host)

    # Get C2 info from Triage JSON report
    def get_c2_info(self):
        extracted = self.data.get('extracted', [])
        for ext in extracted:
            dropper = ext.get('dropper', {})

            urls = dropper.get('urls', [])
            for url in urls:
                self.add_attr('url', url['url'], url['type'])

            config = ext.get('config', {})

            c2_config = config.get('c2', [])
            for url in c2_config:
                ip_port = re.fullmatch(r'[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}:[0-9]+', url)
                if ip_port:
                    self.add_attr('dst', url)
                else:
                    self.add_attr('url', url)

            self.get_c2_credentials(config)
            self.get_c2_credentials(ext)

```

joebox.py

```

import json
import os
import time
import subprocess
import requests

from plugins.abstract_plugin import AbstractPlugin
from utils.state import State
from pathlib import Path

class_name = 'Joebox'

class Joebox(AbstractPlugin):
    pass
    def __init__(self):
        super().__init__()
        self.data = None
        self.joebox_path = ''
        self.joebox_url = State.config['joebox']['url']
        self.type_map = {
            'domain': 'domain',
            'hostname': 'hostname',
            'dst': 'ip-dst|port',
            'dumped_file': 'filename',
            'family': 'tag',
            'md5': 'md5',
            'sha1': 'sha1',
            'sha256': 'sha256',
            'sha512': 'sha512',
            'target': 'email-attachment',
            'url': 'url',
            'tls_ja3': 'ja3-fingerprint-md5',
            'user-agent': 'user-agent',
            'username': 'email-dst',
            'email_to': 'email-dst'
        }

    # Run Joebox
    def run(self, args):
        print('Run Joebox')
        todo_path = str(State.path)
        if todo_path:
            print(todo_path)
            # create directory for joebox output
            if not os.path.exists(todo_path + '/joebox'):
                os.makedirs(todo_path + '/joebox')
                self.joebox_path = todo_path + '/joebox/'

        else:
            print('No samples')
            return

        for file in os.listdir(todo_path):
            if self.check_executable(todo_path, file):
                State.attachment_name = file
                self.run_joebox(todo_path, file)

        joebox_report = self.joebox_path + 'joebox_report.json'
        if os.path.exists(joebox_report):
            with open(joebox_report, "r", encoding='utf-8') as f:

```

```

        joebox_report_data = f.read()

        self.data = json.loads(joebox_report_data)
        self.get_signature_info()
        self.get_c2_info()

    else:
        print('No joebox report')
        return

    # Check if file is executable
    def check_executable(self, todo_path, file):
        mime_type_executable = subprocess.run(['file', '--mime-type',
        todo_path + '/' + file], stdout=subprocess.PIPE)
        print(mime_type_executable)
        if 'application/x-dosexec' in
mime_type_executable.stdout.decode('utf-8'):
            return True
        return False

    # Upload file to Joebox and get report/dump/pcap
    def run_joebox(self, todo_path, file):

        # Upload file to Joebox
        url = State.config['joebox']['url']
        apikey = State.config['joebox']['api_key']

        headers = {
            'apikey': apikey,
            'accept-tac': 1
        }

        files = {
            'sample': open(todo_path + '/' + file, 'rb')
        }

        submission_id = requests.post(url + 'submission/new',
data=headers, files=files)
        submission_id = submission_id.json()
        submission_id = submission_id['data']['submission_id']
        print(submission_id)

        headers = {
            'apikey': apikey,
            'submission_id': submission_id
        }

        # Check if Joebox is done
        time.sleep(1000)
        response = requests.post(url + 'submission/info', data=headers)
        check_status = response.json()
        print(check_status['data']['status'])
        while check_status['data']['status'] != "finished":
            time.sleep(60)
            response = requests.post(url + 'submission/info',
data=headers)
            check_status = response.json()
            print (check_status['data']['status'])

        web_id =
check_status['data']['most_relevant_analysis']['webid']
        print(web_id)

```

```

# Get Joebox JSON report
headers = {
    'apikey': apikey,
    'web_id': web_id,
    'type': 'json'
}
report_joebox = requests.post(url + 'analysis/download',
data=headers)
save_report = open(self.joebox_path + 'joebox_report.json',
'w')
save_report.write(report_joebox.text)
save_report.close()

# Get Joebox PCAP
headers = {
    'apikey': apikey,
    'web_id': web_id,
    'type': 'pcap'
}
pcap_joebox = requests.post(url + 'analysis/download',
data=headers)
save_pcap = open(self.joebox_path + 'suricata.pcap', 'wb')
save_pcap.write(pcap_joebox.content)
save_pcap.close()

# Get Joebox Memory Dump
headers = {
    'apikey': apikey,
    'web_id': web_id,
    'type': 'memdumps'
}
memdumps_joebox = requests.post(url + 'analysis/download',
data=headers)
save_memdumps = open(self.joebox_path + 'memdumps.zip', 'wb')
save_memdumps.write(memdumps_joebox.content)
save_memdumps.close()

# Get Joebox Files
headers = {
    'apikey': apikey,
    'web_id': web_id,
    'type': 'unpackpe'
}
files_joebox = requests.post(url + 'analysis/download',
data=headers)
save_files = open(self.joebox_path + 'files.zip', 'wb')
save_files.write(files_joebox.content)
save_files.close()

if not os.path.exists(self.joebox_path + 'files'):
    os.makedirs(self.joebox_path + 'files')
if not os.path.exists(self.joebox_path + 'memory'):
    os.makedirs(self.joebox_path + 'memory')

# Unzip Joebox Files/Memdumps
subprocess.run(['7z', 'x', self.joebox_path + '/memdumps.zip',
'-o' + self.joebox_path + '/memory/'])
subprocess.run(['7z', 'x', self.joebox_path + '/files.zip', '-
o' + self.joebox_path + '/files/'])

# Add C2 info to JSON file

```

```

def add_attr(self, field, field_val):
    print('add_attr')
    if not field_val:
        return
    misp_type = self.type_map.get(field)
    # Add attribute to C2 JSON file
    c2_file_name = self.joebox_path + 'c2.json'
    if os.path.exists(c2_file_name):
        with open(c2_file_name, "r", encoding='utf-8') as f:
            c2_data = json.load(f)
    else:
        c2_data = {}

    if misp_type == 'domain' and field_val.count('.') > 1:
        misp_type = 'hostname'

    with open(c2_file_name, "w", encoding='utf-8') as f:
        print(c2_data)
        if not c2_data:
            print('c2_data is empty')
            c2_data = {misp_type: field_val}
        else:
            c2_data.update({misp_type: field_val})
        json.dump(c2_data, f)

# Write signature to JSON file
def add_signature(self, field, field_val):
    signature_file_name = self.joebox_path + 'signature.json'

    if os.path.exists(signature_file_name):
        with open(signature_file_name, "r") as f:
            signature_data = json.loads(f)
    else:
        signature_data = []

    with open(signature_file_name, "w") as f:
        misp_type = self.type_map.get(field)
        if misp_type == 'tag':
            tag = self.search_tag(field_val)

            if tag:
                signature_data.append(tag)
                json.dump(signature_data, f)

# Get MISP Tags
def search_tag(self, tag):
    # Ignore the misp-galaxy tags
    available_tags = filter(lambda x: not x.startswith('misp-'),
State.tags)
    for misp_tag in available_tags:
        if tag.lower() in misp_tag.lower():
            return misp_tag

# Get Malware Signature from Joebox JSON report
def get_signature_info(self):
    configs = self.data.get('config', [])
    for config in self.make_list(configs):
        threatname = config.get('@threatname')
        if threatname:
            self.add_signature(threatname)

# Get C2 info from Joebox JSON report

```

```

def get_c2_info(self):
    mal_config = self.data.get('malwareconfigs')
    if not mal_config:
        return

    configs = mal_config.get('config', [])
    for config in self.make_list(configs):

        for key, val in config.items():
            if key == '@threatname':
                continue

            cfg = json.loads(val)
            c2_url = cfg.get('C2 url')
            if c2_url:
                self.add_attr('url', c2_url)

            c2_list = cfg.get('C2 list')
            if c2_list:
                for url in c2_list:
                    self.add_attr("url", url, comment="C&C")
            mail_from = cfg.get('From: ')
            if mail_from:
                self.add_attr('email-dst', mail_from)

            for key, value in cfg.items():
                if key in ['C2 url', 'C2 list', 'From: '] or
not value:
                    continue

```

yara.py

```

import subprocess
import os
import json

from plugins.abstract_plugin import AbstractPlugin
from collections import OrderedDict
from utils.state import State
from pathlib import Path

class_name = 'Yara'

class Yara(AbstractPlugin):

    def __init__(self):
        super().__init__()
        self.data = {}

    def run(self, args):
        print('Yara run')

        self.scan_path = str(State.path)
        self.yara_path = State.config['yara']['rules_dir']
        sandboxes_to_scan = State.config['sandbox']['sandbox_list']

        # Loop through the sandboxes and scan them
        sandboxes = sandboxes_to_scan.split(',')

        # Iterate over the items in the list and process each one
        for sandbox in sandboxes:

```

```
sandbox = sandbox.strip()
self.scan(sandbox)

def scan(self, sandbox):
    # Scan the directory for yara rules
    yara_output_path_txt = self.scan_path + '/' + sandbox +
'/yara.txt'
    yara_output_path_json = self.scan_path + '/' + sandbox +
'/yara.json'
    yara_memory_path = self.scan_path + '/' + sandbox + '/memory/'
    yara_files_path = self.scan_path + '/' + sandbox + '/files/'
    for file in os.listdir(self.yara_path):
        with open(yara_output_path_txt, 'a') as f:
            yara_output = subprocess.run(['yara', '-w', '-r',
self.yara_path + file, yara_memory_path], stdout=f)

        for file in os.listdir(self.yara_path):
            with open(yara_output_path_txt, 'a') as f:
                yara_output = subprocess.run(['yara', '-w', '-r',
self.yara_path + file, yara_files_path], stdout=f)

    with open(yara_output_path_txt, 'r') as f:
        # Read the lines of the file
        lines = f.readlines()

    # Write the signatures to a JSON file
    for line in lines:
        signature = line.split()[0]
        self.data[signature] = self.data.get(signature, 0) + 1
    with open(yara_output_path_json, 'w') as f:
        json.dump(self.data, f)

    # Remove the tmp txt file
    os.remove(yara_output_path_txt)
```

pcap.py

```
import subprocess

from plugins.abstract_plugin import AbstractPlugin
from utils.state import State
from pathlib import Path

class_name = 'PCAP'

class PCAP(AbstractPlugin):

    def __init__(self):
        super().__init__()

    def run(self, args):
        print('Suricata run')

        self.scan_path = str(State.path)
        self.suricata_path = State.config['suricata']['suricata_dir']
        sandboxes_to_scan = State.config['sandbox']['sandbox_list']
        self.suricata_config =
State.config['suricata']['suricata_config']
```

```
# Loop through the sandboxes and scan them
sandboxes = sandboxes_to_scan.split(',')

# Iterate over the items in the list and process each one
for sandbox in sandboxes:
    sandbox = sandbox.strip()
    self.scan(sandbox)

def scan(self, sandbox):
    # Suricata for network traffic
    suricata_pcap_path = self.scan_path + '/' + sandbox +
    '/suricata.pcap'
    suricata_output_path = self.scan_path + '/' + sandbox
    suricata_output = subprocess.run(['suricata', '-c',
self.suricata_config, '-l', suricata_output_path, '-r',
suricata_pcap_path], stdout=subprocess.PIPE)
```

malwarebazaar.py

```
import requests
import json
import os
import subprocess

from plugins.abstract_plugin import AbstractPlugin
from utils.state import State
from pathlib import Path

class_name = 'MalwareBazaar'

class MalwareBazaar(AbstractPlugin):

    def __init__(self, ):
        super().__init__()
        self.api_key = State.config['malwarebazaar']['api_key']
        self.url = State.config['malwarebazaar']['url']
        self.sandboxes_to_scan =
        State.config['sandbox']['sandbox_list']
        self.signature = []

    def run(self, args):
        print('MalwareBazaar Plugin')
        self.todo_path = str(State.path)
        for file in os.listdir(State.path):
            print('Checking file: ' + file)
            if self.check_executable(file):
                print('Found executable: ' + file)
                if self.check_sandbox_signature():
                    self.upload(file)

    def check_executable(self, file):
        mime_type_executable = subprocess.run(['file', '--mime-type',
self.todo_path + '/' + file], stdout=subprocess.PIPE)
        if 'application/x-dosexec' in
mime_type_executable.stdout.decode('utf-8'):
            return True
        else:
            return False
```

```
def check_sandbox_signature(self):
    # Loop through the sandboxes and check for malware signature
    sandboxes = self.sandboxes_to_scan.split(',')

    for sandbox in sandboxes:
        print(sandbox)
        sandbox = sandbox.strip()
        signature_file = self.todo_path + '/' + sandbox +
        '/signature.json'
        if os.path.exists(signature_file):
            with open(signature_file, 'r') as f:
                signature = json.load(f)
                self.signature = self.signature + signature
    if self.signature:
        return True
    else:
        return False

def upload(self, file):
    headers = {
        'API-KEY': State.config['malwarebazaar']['api_key']
    }

    data = {
        'anonymous': 1,
        'delivery_method': 'email_attachment',
        'tags': self.signature,
        'references': {
            'any_run': [
            ],
            'joe_sandbox': [
            ],
            'malpedia': [
            ],
            'twitter': [
            ],
            'links': [
            ],
        },
        'context': {
            'dropped_by_md5': [
            ],
            'dropped_by_sha256': [
            ],
            'dropped_by_malware': [
            ],
        }
    }

    file_path = self.todo_path + '/' + file
    files = {
        'json_data': (None, json.dumps(data),
        'application/json'),
        'file': (open(file_path, 'rb'))
    }

    print(data)

    requests.post(self.url, files=files, verify=True,
    headers=headers)
```

statistics.py

```
import json
```

```

import os

from plugins.abstract_plugin import AbstractPlugin
from utils.state import State
from pathlib import Path

class_name = 'Statistics'

class Statistics(AbstractPlugin):

    def __init__(self):
        super().__init__()

    def run(self, args):
        print('Statistics run')

        sandboxes_to_scan = State.config['sandbox']['sandbox_list']

        # Loop through the sandboxes
        sandboxes = sandboxes_to_scan.split(',')

        # Iterate over the items in the list and process each one
        for sandbox in sandboxes:
            sandbox = sandbox.strip()
            # Create the stats file if it doesn't exist
            stats_file = State.config['stats']['stats_dir'] + sandbox +
'_stats.json'
            if not Path(stats_file).is_file():
                with open(stats_file, 'w') as f:
                    json.dump({}, f)

            with open(stats_file, "r", encoding='utf-8') as f:
                stats_data = f.read()
                self.data = json.loads(stats_data)
                self.update_stats(stats_file, sandbox)

        def update_stats(self, stats_file, sandbox):
            signature_file = str(State.path) + '/' + sandbox +
'/signature.json'
            print(signature_file)

            # Update the stats file
            if os.path.exists(signature_file):
                with open(signature_file, "r", encoding='utf-8') as f:
                    signatures = f.read()
                    signatures = json.loads(signatures)
                    for signature in signatures:
                        print(signature)
                        self.data[signature] = self.data.get(signature, 0) + 1
                        print(self.data)
                    with open(stats_file, 'w') as f:
                        json.dump(self.data, f)
            else:
                print('No signature file')
                return

```

misp.py

```

import os
import json
import subprocess

```

```

from plugins.abstract_plugin import AbstractPlugin
from utils.state import State
from pathlib import Path
from pymisp import ExpandedPyMISP, MISPEvent, MISPAttribute
from datetime import datetime

class_name = 'MISP'

class MISP(AbstractPlugin):
    pass
    def __init__(self):
        super().__init__()
        self.data = {}
        self.misp_distribution_id = 1 # This community only
        self.misp_threat_level_id = 3 # Low
        self.misp_analysis = 2 # Completed
        self.count_events = 0

    def run(self, args):
        print('MISP run')

        self.todo_path = str(State.path)
        print(self.todo_path)
        sandboxes_to_scan = State.config['sandbox']['sandbox_list']

        # Check if mail.json exists
        if not Path(self.todo_path + '/mail.json').is_file():
            print('No mail.json file')
            return

        # Loop through the sandboxes
        sandboxes = sandboxes_to_scan.split(',')

        # Iterate over the items in the list and process each one
        for sandbox in sandboxes:
            sandbox = sandbox.strip()
            self.check_signature_and_c2(sandbox)
            # Count the number of signatures
            #count_signature = self.count_signature(sandbox)
            #count_c2 = self.check_c2(sandbox)
            #if count_signature > 0 and count_c2 > 0:
            #    break
            #else:
            #    return

            #self.create_misp_event(sandbox)

        def check_signature_and_c2(self, sandbox):
            if os.path.exists(self.todo_path + '/' + sandbox +
                '/signature.json') and \
                os.path.exists(self.todo_path + '/' + sandbox + '/c2.json')
            and \
                os.path.exists(self.todo_path + '/' + 'mail.json'):

                # Read the signature.json file
                with open(self.todo_path + '/' + sandbox +
                    '/signature.json', "r", encoding='utf-8') as f:
                    signatures = f.read()
                    signatures = json.loads(signatures)

```

```

        # Read the c2.json file
        with open(self.todo_path + '/' + sandbox + '/c2.json', "r",
encoding='utf-8') as f:
            c2 = f.read()
            c2 = json.loads(c2)

        # Read the mail.json file
        with open(self.todo_path + '/' + 'mail.json', "r",
encoding='utf-8') as f:
            mail = f.read()
            mail = json.loads(mail)

        # Create the misp event
        if self.count_events > 2:
            self.count_events += 1
            self.create_misp_event(sandbox, signatures, c2, mail)

    def create_misp_event(self, sandbox, signatures, c2, mail):
        # Create the MISP event
        misp = State.misp
        #misp = ExpandedPyMISP(State.config['misp']['url'],
State.config['misp']['key'], True)
        event = MISPEvent()
        signature_slash = '/'.join(signatures)
        event.info = signature_slash + ' Spam Run (' +
datetime.now().strftime("%Y-%m-%d") + ' - ' + mail['email-subject'] +
')'

        event.distribution = self.misp_distribution_id
        event.threat_level_id = self.misp_threat_level_id
        event.analysis = self.misp_analysis
        event.add_tag('tlp:green')
        event.add_tag('Crimeware')
        event.add_tag('exe')

        for signature in signatures:
            event.add_tag(signature)

        misp.add_event(event)

        # Add the attributes
        attributes = []
        for type, value in mail.items():
            a = MISPAtribute()
            a.category = 'Payload delivery'
            a.type = type
            a.value = value
            a.to_ids = False
            attributes.append(a)

        for type, value in c2.items():
            a = MISPAtribute()
            a.category = 'Network activity'
            a.type = type
            a.value = value
            a.to_ids = True
            a.comment = 'C&C'
            attributes.append(a)

        misp.add_attribute(event, attributes)

```

```
# Upload malware samples
for file in os.listdir(self.todo_path):
    print(file)
    file_path = self.todo_path + '/' + file
    mime_type_executable = subprocess.run(['file', '--mime-
type', file_path], stdout=subprocess.PIPE)
    if 'application/x-dosexec' in
mime_type_executable.stdout.decode('utf-8'):
        a = MISPAtribute()
        a.category = 'Payload delivery'
        a.type = 'malware-sample'
        a.value = file
        a.data = Path(file_path)
        a.expand = 'binary'
        a.to_ids = True
        a.comment = 'Attachment'
        event.attributes.append(a)
event.run_expansions()

misp.update_event(event)
```

prmapf.py

```
import redis
import time
import json
import time

from rich.console import Console
from rich.table import Table
from rich.columns import Columns
from rich.live import Live
from rich.panel import Panel
from rich.progress_bar import ProgressBar

console = Console()

r = redis.Redis(host='localhost', port=6379, db=0)

def load_redis_data() -> list:
    keys = sorted([int(x) for x in r.keys()], reverse=True)
    return [json.loads(r.get(str(key))) for key in keys[:10]]

def load_data(filename) -> list:
    with open(filename, "r", encoding="utf-8") as datafile:
        data = json.load(datafile)
        sorted_data = sorted(data.items(), key=lambda x: x[1],
reverse=True)
        return [f"{cnt: 5d} | {name}" for name, cnt in
sorted_data[:10]]

def get_malware_table() -> Table:
    table = Table(show_header=True, header_style="bold bright_yellow")
    table.add_column("Nr", width=8)
    table.add_column("Sample")
    table.add_column("Progress", max_width=17)

    data = load_redis_data()
    for item in data:
```

```

        pgb = ProgressBar(total=item["total_progress"],
                           completed=int(item["progress"]), width=10,
                           complete_style="bright_yellow")

        if item.get('error'):
            caption = "Error"
        elif item["progress"] > item["total_progress"]:
            caption = "Done"
        else:
            caption = f"{item['progress']}/{item['total_progress']}"

        progress_panel = Columns([pgb, caption])
        table.add_row(
            str(item["task_id"]),
            item["sample_name"],
            progress_panel
            # item["progress"],
        )
    return table

def get_stats_table() -> Table:
    jb_data = load_data("joebox_stats.json")
    tr_data = load_data("triage_stats.json")
    data = zip(jb_data, tr_data)
    table = Table(show_header=True, header_style="bold bright_yellow")
    table.add_column("Joebox-Stats")
    table.add_column("Triage-Stats")
    for item in data:
        table.add_row(*item)
    return table

def get_panel() -> Panel:
    return Panel.fit(Columns([get_stats_table(), get_malware_table()]),
                     title="Malware Analysis Pipeline Framework", title_align="left")

with Live(get_panel(), refresh_per_second=1) as live:
    while True:
        time.sleep(1)
        live.update(get_panel())

```