**Creating a python script to scrape IOCs**

**Problem Statement**

Currently the process of retrieval of Indicators of Compromise (IOCs) is a manual process that involves a person skimming through a website and collecting (IOCs) including: domain names, URLs, hashes, malicious IP addresses, file paths, file names, process names, registry keys, to name a few. This process can be very tedious and time consuming when looked in the long run.

Looking at the fact that these IOCs follow a general pattern, it allows us to dive deep into it using pattern matching to make this tedious and time consuming task to be automated. This is where our project comes in. Our project is essentially a python script that makes use of regular expressions to match a certain criteria to detect these IOCs. IOCs are integral within Trustwave because IOCs are added on a regular basis to the main threat intelligence platform, called as Intel Fusion Platform (IFP). Hence, automating this process can help to substantially accelerate the otherwise slow and manual process of collecting IOCs.

**Technical Aspect**

*Python Libraries used*

1. Requests: used to make HTTP requests to access the web page
2. Re: provides regular expressions matching operations
3. Validators: to validate the type of contents
4. BeautifulSoup: used to scrape information from web pages
5. Nltk: a python package for natural language processing

*Forms of IOCs detected and the method used for detection*

1. Domain Names: Used an inbuilt *validators* library in python to check for domain names and simultaneously checked with a regular expression to check against commonly used domain names like: .com, .net, etc. (line 104, line 184)
2. URLs: Similar method as for detecting domain names. However, checks for words like ‘http’, ‘hxxp’, ‘hxxps’ and ‘https’ is also made in the string, to qualify it as a URL.
3. Hashes (MD5, SHA256, SHA512): A very interesting approach has been adopted here to detect hashes. As we know that there are multiple kinds of hashes, however, there are only some that are used to indicate IOCs. Some of these include: SHA256, SHA512 and MD5. These named hashes have a very specific characteristic, i.e., they are of known lengths. Described below is the flow of logic used:
4. Detect if the word has an English connotation
5. Look for words including ‘http’ and ‘https’
6. Check if it is an alphanumeric expression
7. Check for the length of the word and whether it is of 32, 64, 128 characters

If the word does not fulfill conditions (a) and (b) and satisfies (c) and (d), then it is categorized as a hash.

1. IP Addresses: checked against a regular expression to detect IP Address
2. File Paths: checks for symbols like ‘\\’, ‘.ps1’, ‘.doc’, ‘,js’, ‘.vbs’, ‘.Vbs’, ‘.py’, ‘.exe’, ‘.dll’ and alphabet characters were made to extract the file path
3. File Names: File names were extracted from the file paths themselves, and the approach was merged with File paths.
4. Registry Keys: checks were made against terms that are commonly found in registry keys.

*The flow for the logic of code*

1. The code asks the user to input the file path containing URLs separated by new line characters
2. A check is made for each URL to see whether the website is alive
3. Once the status is known, the contents of the website is parsed
4. A folder is created for each url to save the IOCs
5. Next step is to remove embedded JavaScript and CSS in the parsed website contents
6. Each word in the website is scanned for IOCs, using specially crafted functions, as discussed above
7. IOCs are categorised and are then stored in a dictionary data structure
8. The list of IOCs is then checked for duplication and duplicates are removed
9. The format of IOCs is standardised
10. Eventually, IOCs are written to text files, depending on its category

**Flow of program**

Running the program



Enter 1 for single URL or 2 for a text file containing URLs to be crawled.

For option 1: 

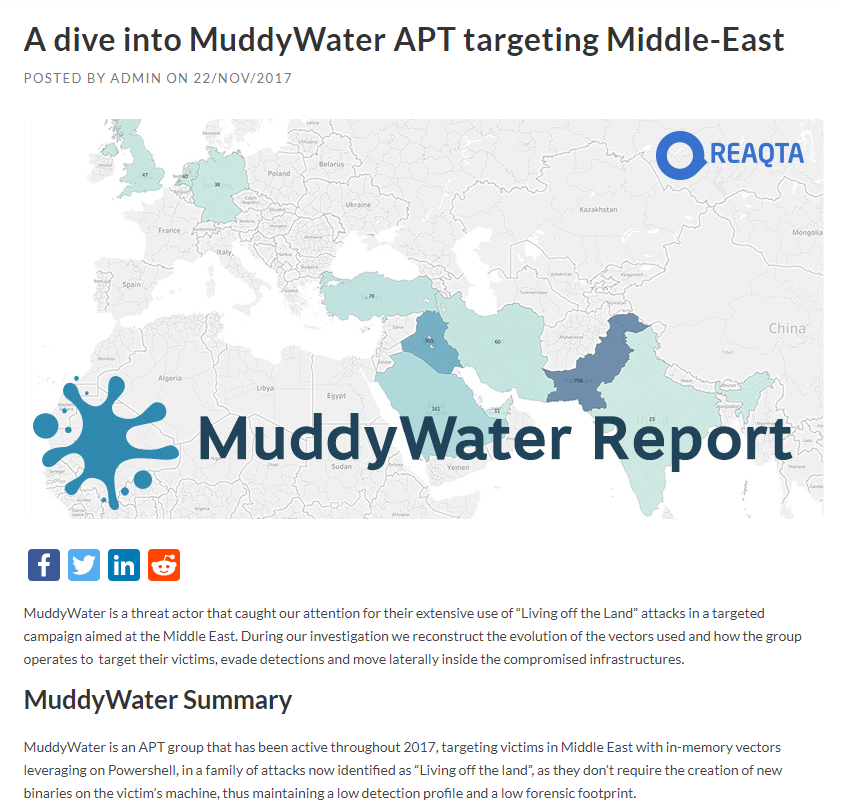
For option 2: 

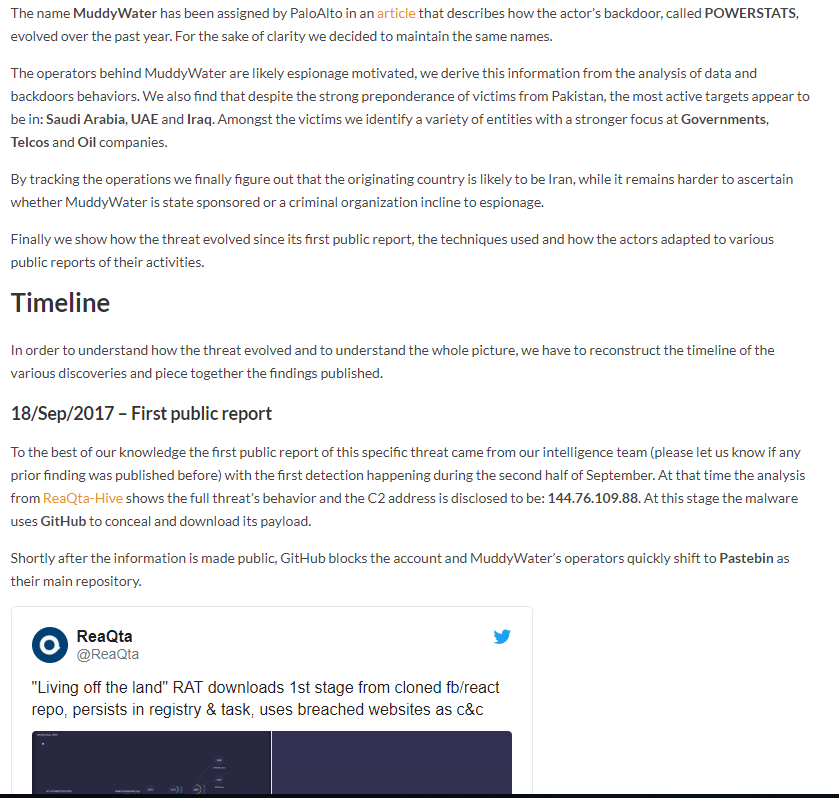
Program will crawl the URL line by line for option 2 or crawl through the single URL from user input

**Examples of Url Tested in text file**

<https://reaqta.com/2017/11/muddywater-apt-targeting-middle-east/>

*Report format*



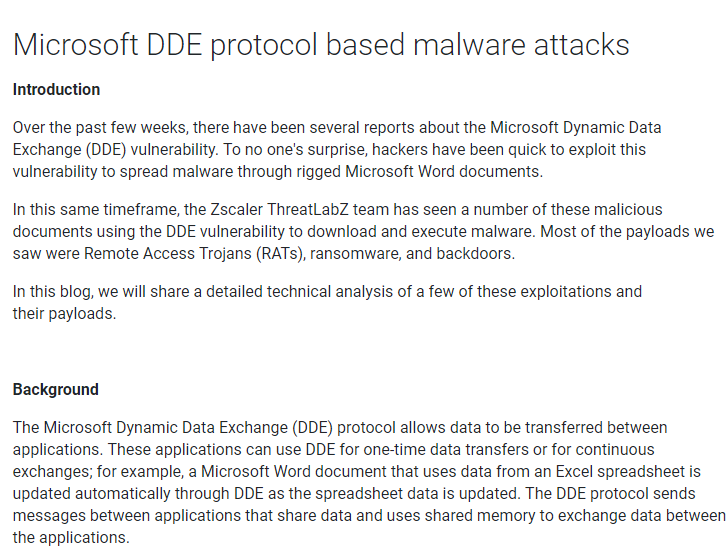


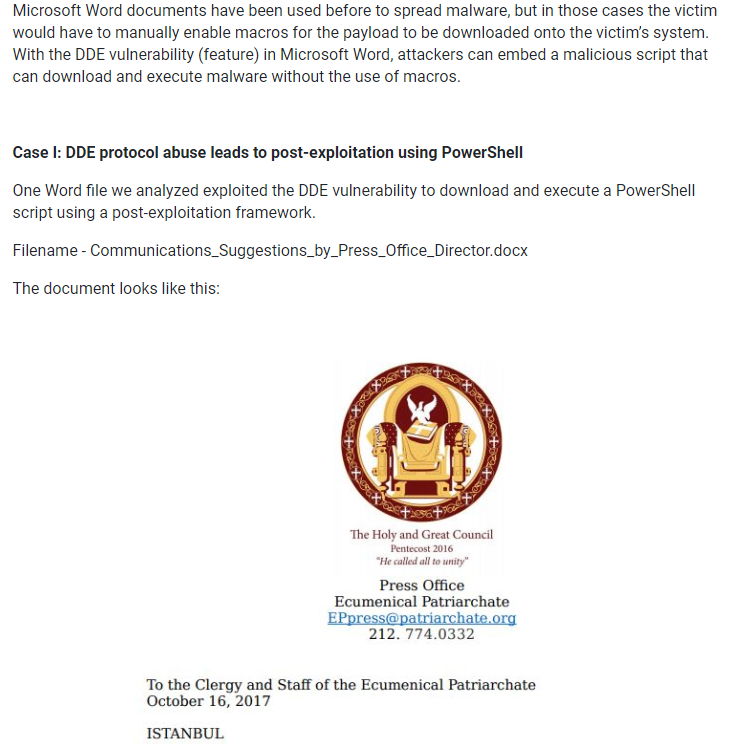


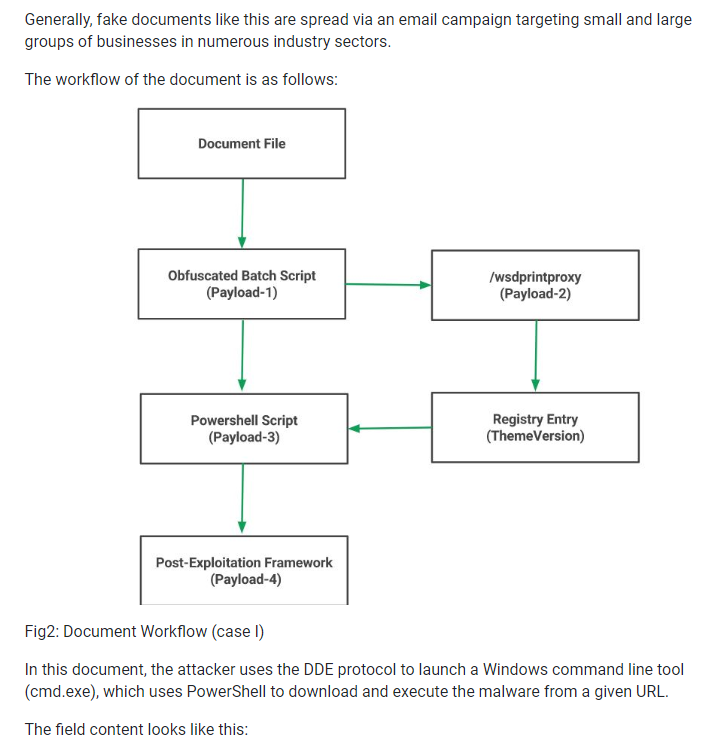
**Examples of Url Tested in text file**

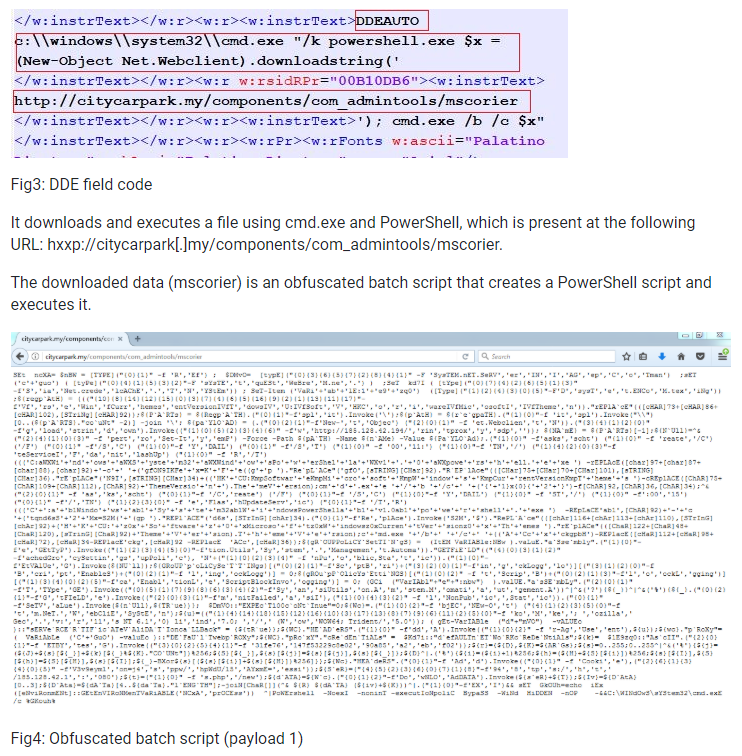
<https://www.zscaler.com/blogs/research/microsoft-dde-protocol-based-malware-attacks>

*Report Format*

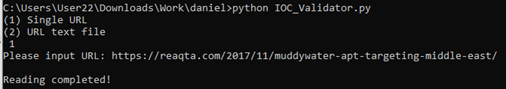


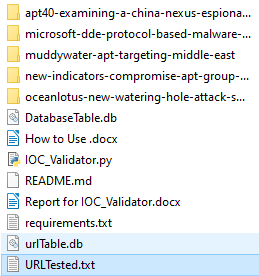






**Final Output from program**

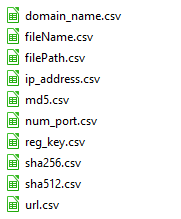




Inside each folder is a IOCs and URL folder:



Inside the IOCs folder contain CSV file containing each different types of IOCs:



Inside the URL folder is a CSV file containing the original URL that was crawled



**Limitations of Program**

1. Not able to follow through links in reports