ANDROID MALWARE DETECTION

PURVA TARANEKAR

BITS Pilani K K Birla Goa Campus

Zuarinagar, Goa

Guided by- Prof. Hemant Rathore

Assistant Professor at BITS Pilani K K Birla Goa Campus,

Zuarinagar, Goa

Table of Contents

INTRODUCTION

Mobile phones or smartphones have become a crucial part of almost every aspect of our life. Smartphones have app stores that contain thousand to millions of applications. These applications assist at all times, say cooking, travel, finance management, entertainment etc.

However, not all applications tend to be benign. These websites or app stores may contain malicious applications. Such applications are said to contain malware.
Malware is a collective term for viruses, spyware and ransomware. These applications contain code that may create extensive damage to user's data and system or may gain unauthorised access to the network. They are delivered in the form of file or android application or over email. It gets activated once user clicks on it and malware starts executing.

Such applications are a big threat as they look absolutely similar to any other benign application but internally have potential to cause great damage and play with privacy of user. At this point, it is important that we become vigilant before downloading any application onto our phones.

Being a general user, it is difficult to determine whether an application is good or bad. Hence, we require smart mechanism that could warn us before downloading that the application we are willing to download may contain malicious content and protect our phones and data from attackers.

PROBLEM STATEMENT

Keeping in the mind the aforementioned problems, we define our problem Statement as follows

We aim to build a Machine Learning Model that could detect if an android application available on app websites is malicious with an accuracy higher than state-of-the-art. The algorithm is built based on data.

We also aim to check the sanity of the websites from which the applications are downloaded.

BACKGROUND STUDY

Malware Detection can be done in three ways.

- Static Analysis – In this technique, the static or unchanging features such as permissions, intents, APIs are extracted. This analysis happens without executing any code. In this way, all possible paths get revealed. Algorithms such as Naïve Bayes sum, RS, DT are used to build model to detect malware.

  For example – if camera app asks for permissions to access contact app, the camera app becomes a suspect and its download can be avoided. Or, if an app requests a lot more permissions than usual, it could be a malicious app.

  However, this method is not completely reliable as it doesn't consider any of the properties to detect a malware. There are high chances of occurrences of false positives.

- Dynamic Analysis – This technique executes a program and observes the results. It considers only one path at a time but this can be improved with stimulations. Dynamic properties such as intent filters are analysed.

  There are two types of dynamic analysis.
  - In-the-box analysis – If the analysis resides on the same permission level, or architectural layer, as the malicious software, then malware can detect and tamper with the analysis. This is called in-the-box analysis.
  - Out-of-the-box analysis - If the analysis was to reside in a lower layer, say, the kernel, then it would increase security but make it more difficult to intercept app data and communications. This is called out-of-the-box analysis.

- Hybrid Analysis - Hybrid malware detectors use static analysis to find all possible activity paths before dynamically logging its system calls with kernel-level sandboxing. Hybrid methods can increase robustness, monitor edited apps, increase code coverage, and find vulnerabilities.

ENVIRONMENTAL SETUP

To carry out the experiment, following setup was done before and during the project

- Python 3.7.1 has to be installed and environment variables to be populated accordingly
- Anaconda installed to access Anaconda Prompt to start Jupyter Notebook
- Google Chrome Version 81.0.4044.113
- Install chromedriver application compatible with google chrome version

APPROACH AND SOLUTION

The approach on the development of an efficient model has been divided into following broad categories.

- Data Collection
- Feature Extraction
- Model Building
- Sanity Check

On the websites, the applications are stored categorywise into finance, entertainment, sports etc. We have downloaded data preserving the categorial information of each application

1. DATA COLLECTION

As we are aware that to any build any kind of Machine Learning model, we need huge dataset to train our model in order to get high accuracy results. Data Collection contributes to 80% of the total time required in building a model. In order to collect such a large dataset, we divided our data collection workplan into following parts

1.1 Manual Application Download: The websites contain lots of applications but sometimes the applications are not downloadable or they are paid. Hence, first we manually tried downloading applications from each category

1.2 Deciding Websites: After manual check, we listed down few websites which were allowing downloads without any restrictions. Following apps were shortlisted.

- Anzhi Market
- AppChina
- Baidu
- F-droid
- Huwaei
- Lenovo
- LeTV
- MI
- Tencent
- PP Assisstant
- Wandoujia

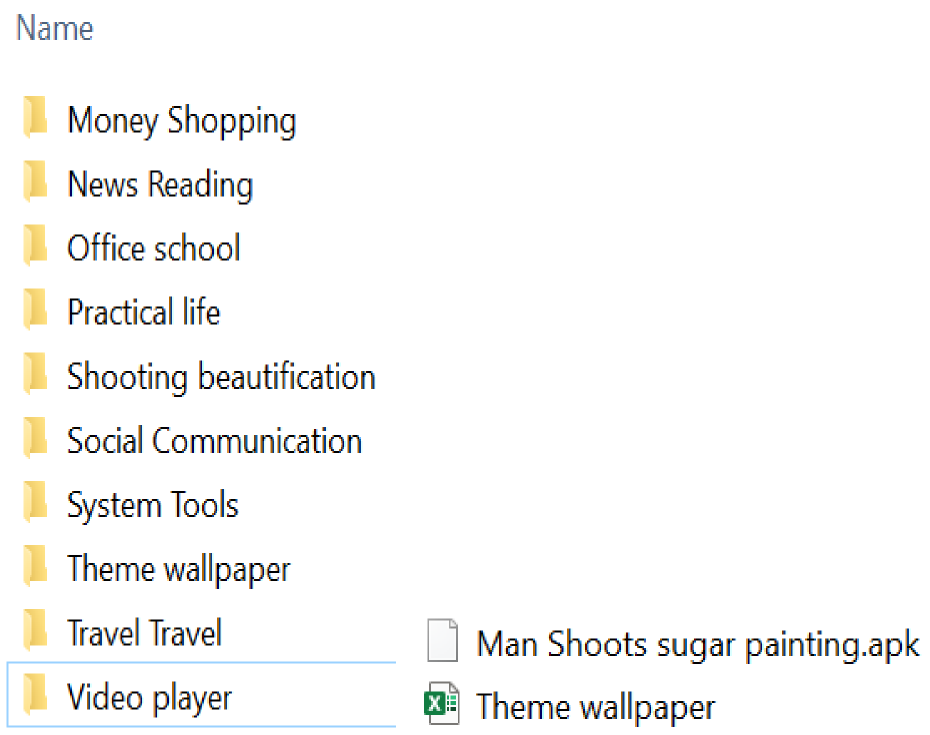
These websites are Chinese based websites. They have been chosen intentionally because these applications have high possibilities of having malicious application which will be useful for building our model.

The app distribution from each of these websites have been kept more or less same because the app count parameter during the sanity check must be same.
There were no categories found in F-droid website. Apps were arranged all together lexicographically.

1.3 Downloading App Links: Application Links of each application per category per website is stored into files, that would later be used for direct link download. Link downloading doesn’t take much time. Advantages of segregating this step from downloading apps step are

- App Link download doesn’t require high RAM and CPU. So, it can be done on any system
- It is easy to transfer app links rather than apps itself to another computer
- It helps us get count of available applications and modify the count to be downloaded based on available counts.

1.4 Downloading Apps: The app links stored into text files are extracted one by one from each category and links are downloaded maintaining each category. Log files have been populated in case of errors and exception for easy resolution

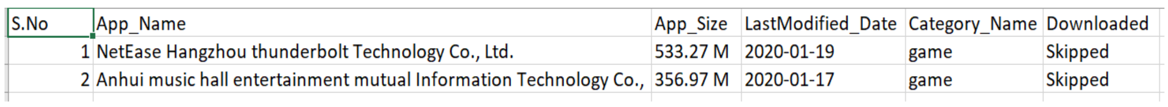


Name

Money Shopping
News Reading
Office school
Practical life
Shooting beautification
Social Communication
System Tools
Theme wallpaper
Travel Travel
Video player

Man Shoots sugar painting.apk
Theme wallpaper

Under each category, an Excel sheet is maintained to keep track of download status along with basic app info.

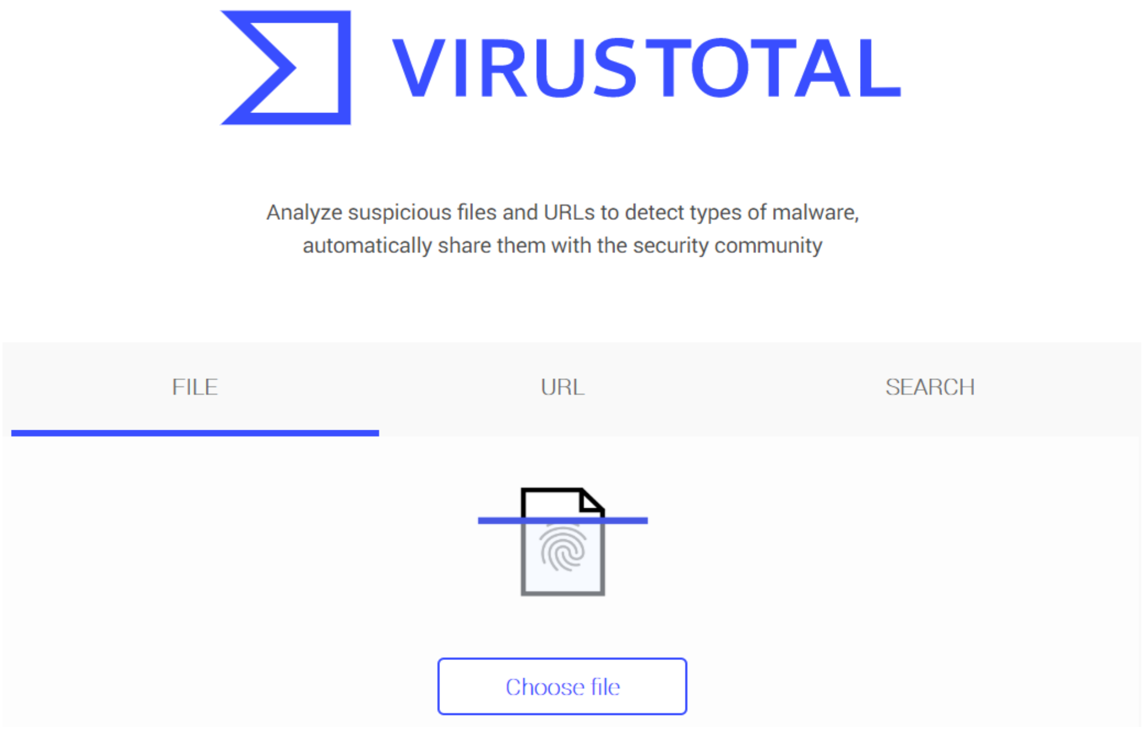


S.No	App_Name	App_Size	LastModified_Date	Category_Name	Downloaded
1	NetEase Hangzhou thunderbolt Technology Co., Ltd.	533.27 M	2020-01-19	game	Skipped
2	Anhui music hall entertainment mutual Information Technology Co.,	356.97 M	2020-01-17	game	Skipped

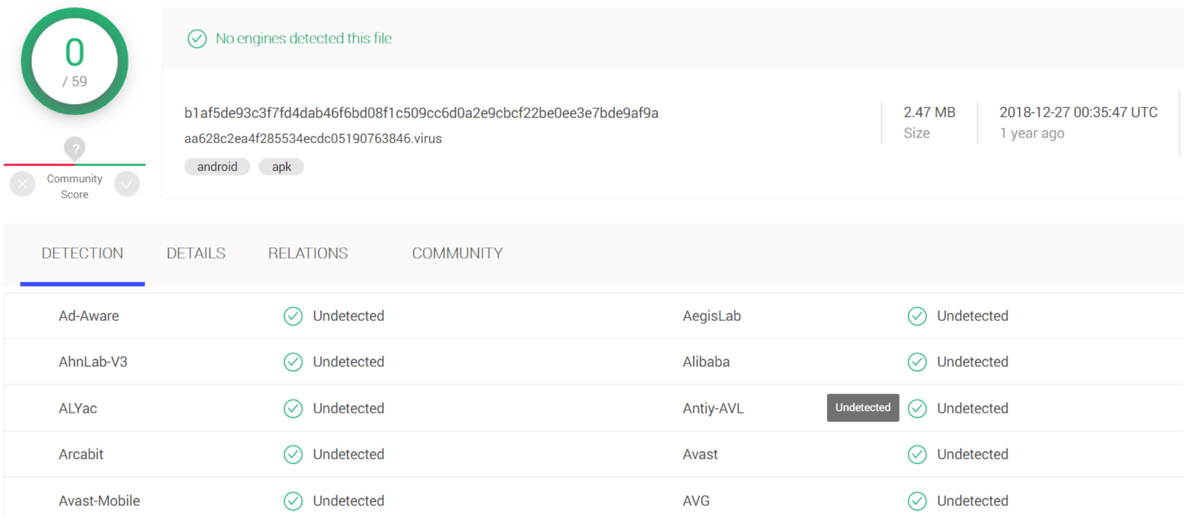
2. FEATURE EXTRACTION

The feature extraction includes fetching the internal details of the applications. Website named Virus Total is used to fetch details such as hash value, history, names, android info, exit tools, bundle info which would be later used in building model.
Feature extraction process has been divided into five parts

2.1 Uploading Apps: One by one, the apps are uploaded onto Virus Total website. Logic is implemented keeping in mind different download speeds at different times. The upload page of Virus Total looks like this.


VIRUSTOTAL
Analyze suspicious files and URLs to detect types of malware, automatically share them with the security community
FILE
URL
SEARCH
Choose file

After Uploading a website, the home page looks like the following.



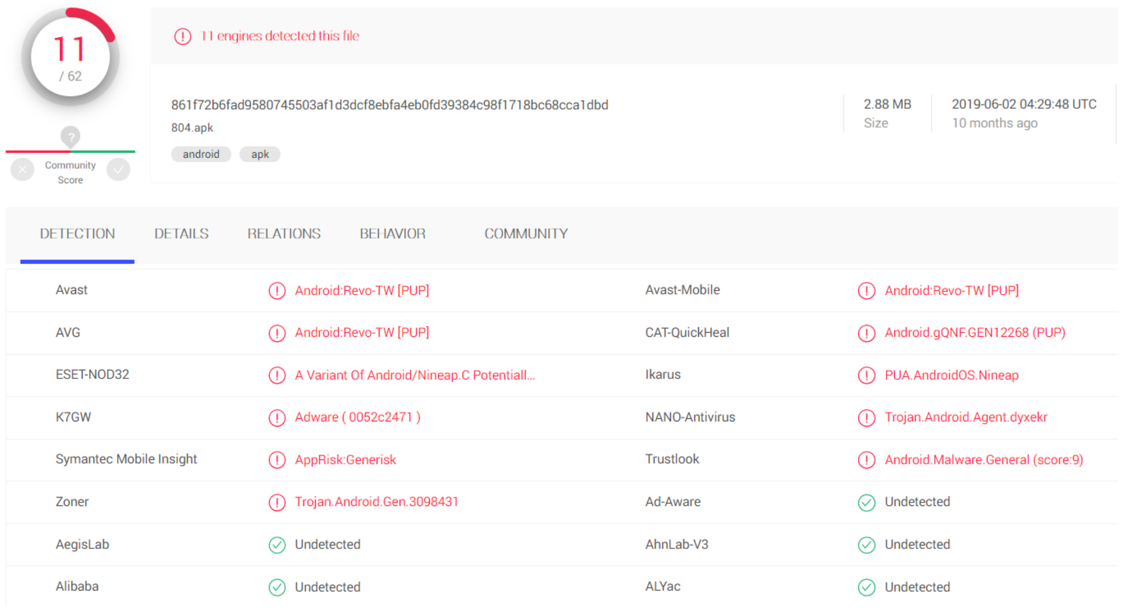
No engines detected this file

b1af5de93c3f7fd4dab46f6bd08f1c509cc6d0a2e9cbcf22be0ee3e7bde9af9a
aa628c2ea4f285534ecdc05190763846.virus
android apk

2.47 MB Size
2018-12-27 00:35:47 UTC 1 year ago

Ad-Aware	Undetected	AegisLab	Undetected
AhnLab-V3	Undetected	Alibaba	Undetected
ALYac	Undetected	Antiy-AVL	Undetected
Arcabit	Undetected	Avast	Undetected
Avast-Mobile	Undetected	AVG	Undetected

2.2 Extract Detection Page: The detection page shown above is extracted. Script is written to do this task automatically. Each antivirus gives their opinion about the uploaded app. If the antivirus finds it to be benign, undetected is written against it. The count of antivirus denoting the app to be malicious divided by total antivirus against which the app is checked is denoted in left top corner circle as shown in below diagram.



11 engines detected this file

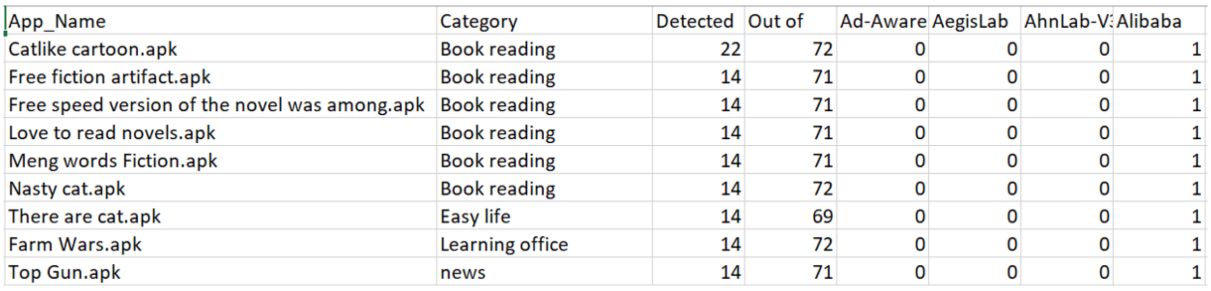
861f72b6fad9580745503af1d3dcf8ebfa4eb0fd39384c98f1718bc68cca1dbd
804.apk
android apk

2.88 MB
Size

2019-06-02 04:29:48 UTC
10 months ago

Avast	Android:Revo-TW [PUP]	Avast-Mobile	Android:Revo-TW [PUP]
AVG	Android:Revo-TW [PUP]	CAT-QuickHeal	Android.gQNF.GEN12268 (PUP)
ESET-NOD32	A Variant Of Android/Nineap.C Potentiall...	Ikarus	PUA.AndroidOS.Nineap
K7GW	Adware (0052c2471)	NANO-Antivirus	Trojan.Android.Agent.dyxekr
Symantec Mobile Insight	AppRisk:Generisk	Trustlook	Android.Malware.General (score:9)
Zoner	Trojan.Android.Gen.3098431	Ad-Aware	Undetected
AegisLab	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected

2.3 Save Detection Page: The details of antivirus and their opinion about the app is stored into an Excel sheet. Each column corresponds to an antivirus. A snapshot depicting how the information is stored have been attached

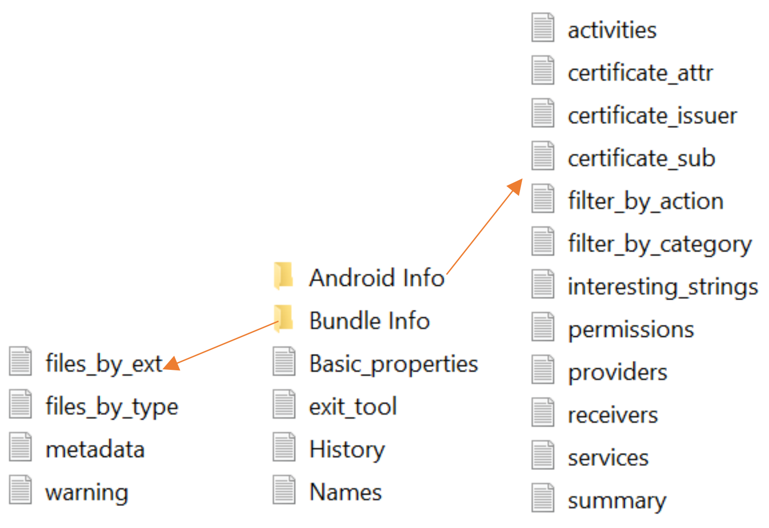


App_Name	Category	Detected	Out of	Ad-Aware	AegisLab	AhnLab-V3	Alibaba
Catlike cartoon.apk	Book reading	22	72	0	0	0	1
Free fiction artifact.apk	Book reading	14	71	0	0	0	1
Free speed version of the novel was among.apk	Book reading	14	71	0	0	0	1
Love to read novels.apk	Book reading	14	71	0	0	0	1
Meng words Fiction.apk	Book reading	14	71	0	0	0	1
Nasty cat.apk	Book reading	14	72	0	0	0	1
There are cat.apk	Easy life	14	69	0	0	0	1
Farm Wars.apk	Learning office	14	72	0	0	0	1
Top Gun.apk	news	14	71	0	0	0	1

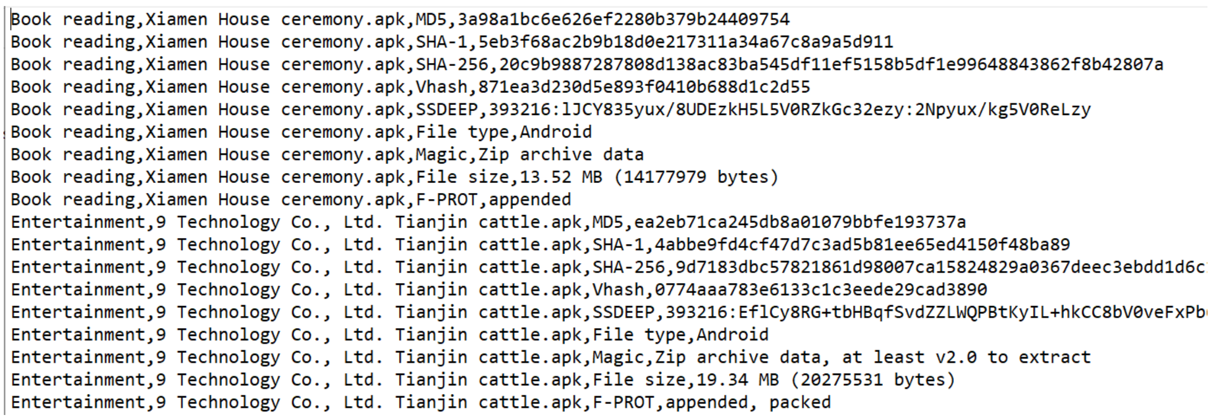
2.4 Extract Details Page: We move to the tab next to Detection tab i.e., Details tab. This tab shows the internal details of the application such as Android Info, Name etc. The code also prints download status of each page on jupyter notebook for quick lookup as.

```
uploading ......  12  ->  3.apk
app details saved in excel sheet !!!
Basic Properties written
history written
Names written
summary written
certificate_attr written
certificate_sub written
certificate_issuer written
permissions written
activities written
services not available for this App
receivers written
providers not available for this App
filter by action written
filter by category written
interesting_strings written
Contents Metadata  written
Contained Files By Type  written
Contained Files By Extension  written
exit_tool written
```

2.5 Save Details Page: These details have been stored in text file. The directory structure is as follows



A snapshot of how each detail is stored into text file maintaining the name and category of the application has been attached.



```
Book reading,Xiamen House ceremony.apk,MD5,3a98a1bc6e626ef2280b379b24409754
Book reading,Xiamen House ceremony.apk,SHA-1,5eb3f68ac2b9b18d0e217311a34a67c8a9a5d911
Book reading,Xiamen House ceremony.apk,SHA-256,20c9b9887287808d138ac83ba545df11ef5158b5df1e99648843862f8b42807a
Book reading,Xiamen House ceremony.apk,Vhash,871ea3d230d5e893f0410b688d1c2d55
Book reading,Xiamen House ceremony.apk,SSDEEP,393216:lJCY835yux/8UDEzkH5L5V0RZkGc32ezy:2Npyux/kg5V0ReLzy
Book reading,Xiamen House ceremony.apk,File type,Android
Book reading,Xiamen House ceremony.apk,Magic,Zip archive data
Book reading,Xiamen House ceremony.apk,File size,13.52 MB (14177979 bytes)
Book reading,Xiamen House ceremony.apk,F-PROT,appended
Entertainment,9 Technology Co., Ltd. Tianjin cattle.apk,MD5,ea2eb71ca245db8a01079bbfe193737a
Entertainment,9 Technology Co., Ltd. Tianjin cattle.apk,SHA-1,4abbe9fd4cf47d7c3ad5b81ee65ed4150f48ba89
Entertainment,9 Technology Co., Ltd. Tianjin cattle.apk,SHA-256,9d7183dbc57821861d98007ca15824829a0367deec3ebdd1d6c
Entertainment,9 Technology Co., Ltd. Tianjin cattle.apk,Vhash,0774aaa783e6133c1c3eede29cad3890
Entertainment,9 Technology Co., Ltd. Tianjin cattle.apk,SSDEEP,393216:EflCy8RG+tbHBqfSvdZZLWQPBtKyIL+hkCC8bV0veFxPb
Entertainment,9 Technology Co., Ltd. Tianjin cattle.apk,File type,Android
Entertainment,9 Technology Co., Ltd. Tianjin cattle.apk,Magic,Zip archive data, at least v2.0 to extract
Entertainment,9 Technology Co., Ltd. Tianjin cattle.apk,File size,19.34 MB (20275531 bytes)
Entertainment,9 Technology Co., Ltd. Tianjin cattle.apk,F-PROT,appended, packed
```

3. MODEL BUILDING

After collection of all the data, we need to organise it according to be given as input to train the model. For this, we need to vectorise all the details. Later, model will be built to detect new malicious application.

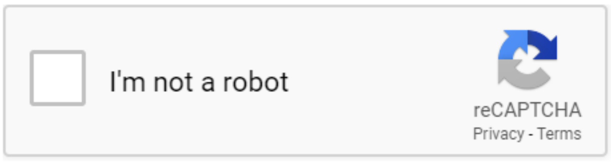
4. SANITY CHECK

Once the model is ready, we need to check how many of the websites contain a lot of malicious application and red marks those websites.

However, the last 2 point are still under progress.

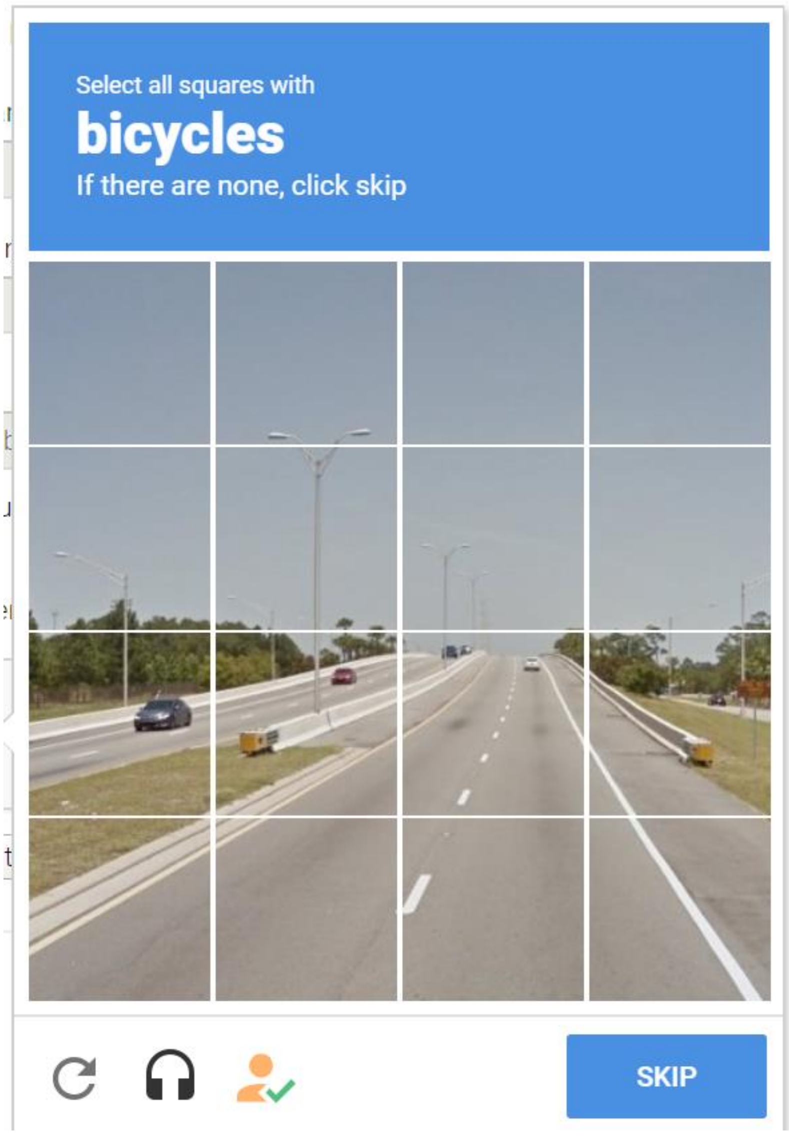
ISSUES

1. One of the most difficult issues faced during extracting information from Virus Total is the appearance of reCaptcha page. The difficulty is that it may appear at any point of time.



To solve this problem, following approaches have been tried

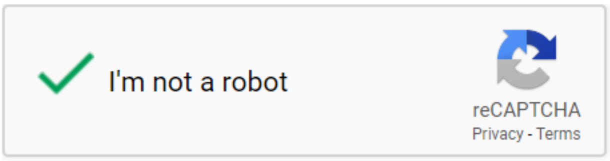
- Uploaded the required page again and again will it gets uploaded. The page gets uploaded after some time which is not predefined. But, this process is time consuming as we cannot leverage the upload limit of 4 requests per minute of Virus Total.
- We can solve reCaptcha manually to prove human. But as appearance of captcha page is not known apriori, this method is not sufficient.
- Servers detect that automation in being performed through a certain IP address. So they tend to block requests from that IP address. Hence, tried to change IP using proxy server. But this method failed as internet stops working on IP change.
- Used application such as psiphon that changes IP address automatically on selecting another country from the list. But it faces same issue of internet disconnect.
- We tried to automate the captcha solving code. It requires complex Machine Learning logic which is not required for simple data extraction technique and it is also time consuming as it would randomly select tiles for mentioned object.



- Tried to use Auto Captcha solver Extension like Buster and automate chrome to bypass captcha solving. The buster icon is visible in above diagram with a orange man with a green tick. When it is used multiple times, it sends heavy network traffic due to which server blocks any further requests.
- There are various Captcha Solving APIs such as Anti-Captcha, 2Captcha, DeathByCaptcha that are available online. When this API gets called, the requests sent on their server is allotted to an idle employee which in turn manually solves the captcha. These APIs are paid starting from $0.5 / 1000 requests.

After having researched and tried all the above mentioned methods, it was observed that Google's reCaptcha is one of the toughest captchas to be bypassed by robot. The reason is because to differentiate between human and robot, it uses many visible and invisible parameters such as speed of click, cursor movement pattern, number of wrong clicks vs number of correct clicks and many more that have not been disclosed.

We solved this issue by continuous checking of captcha page appearance and solved it manually. The script resumes running as normal after captcha handling. Solved Captcha looks like the following



2. There were elements like called shadow roots present in the webpage. Shadow DOM is a new DOM feature that helps you build components. It is similar to scoped subtree inside an element.
   However, it is not visible to normal selenium selectors like xpath. With the help of articles over Internet, this issue was solved using following code.

   ```
   def expand_shadow_element(element,driver):
     shadow_root = driver.execute_script('return arguments[0].shadowRoot', element)
     return shadow_root
   ```

   The function expand_shadow_element takes the element as input, expands it and returns the expanded tree as output

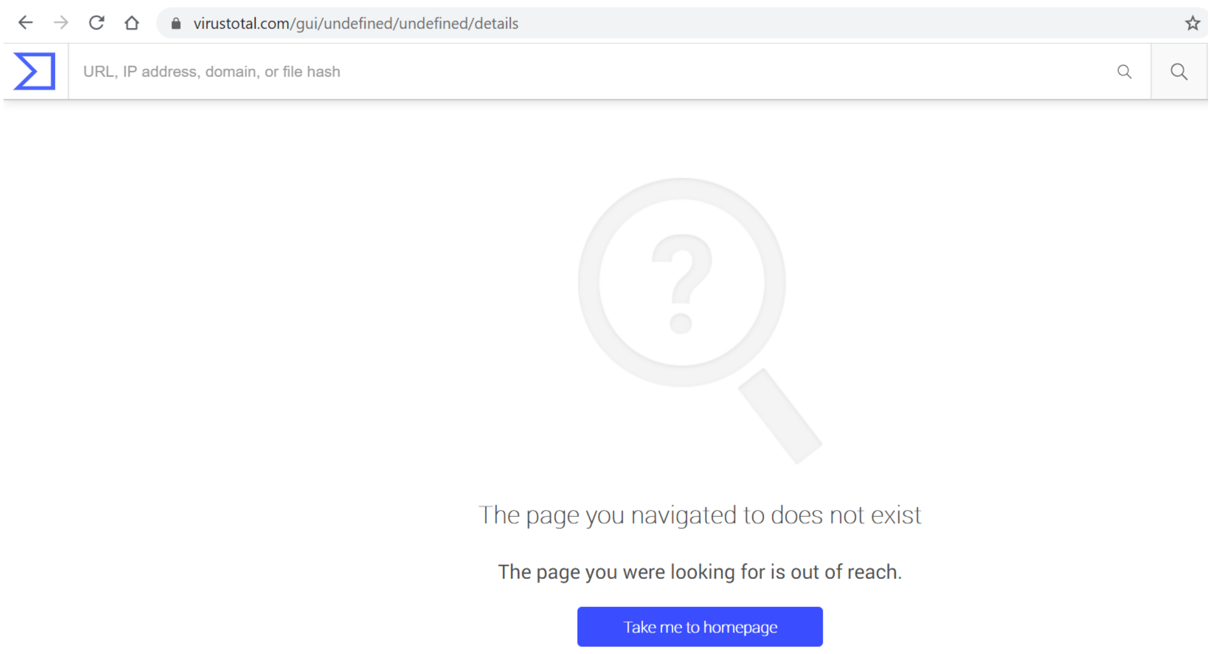
3. Some button were designed under svg tag that made it difficult to click through code because svg and span tags are not clickable. Hence, it threw ElementNotClickable Exceptions.
   It was observed that such tags could be located using combinations of selectors like name, class and viewbox.

   ```
   root9 = root7.find_element_by_xpath(".//span[@class='wrapper'][1]/*[name()='svg'][@viewBox='0 0 24 24']").click(
   ```

4. Each app uploaded on Virus Total was checked against different set of antiviruses, which makes it difficult to store in Excel sheet.
   With the use of dictionary data structure, each antivirus encountered was assigned a unique Id if not already in the dictionary. This unique Id also acted as column number while storing into Excel sheet. With this approach, a common Excel sheet to store detection information of all the applications together was possible.
5. Some of the properties were missing during extraction from Details tab. Hence, text files were kept blank indicating that details were missing. Also, notified on notebook while running the script.

   ```
   services not available for this App
   receivers not available for this App
   providers not available for this App
   ```

6. Due to different design patterns of each website, a separate script was needed for each website to be automated.
7. Some unusual pages like “Item not found” and “Undefined page” keeps appearing dynamically in Virus Total Application.
   Undefined page contains “Take me to Homepage” button. On clicking this button, it directs to the upload page. Script has been employed to do this procedure.

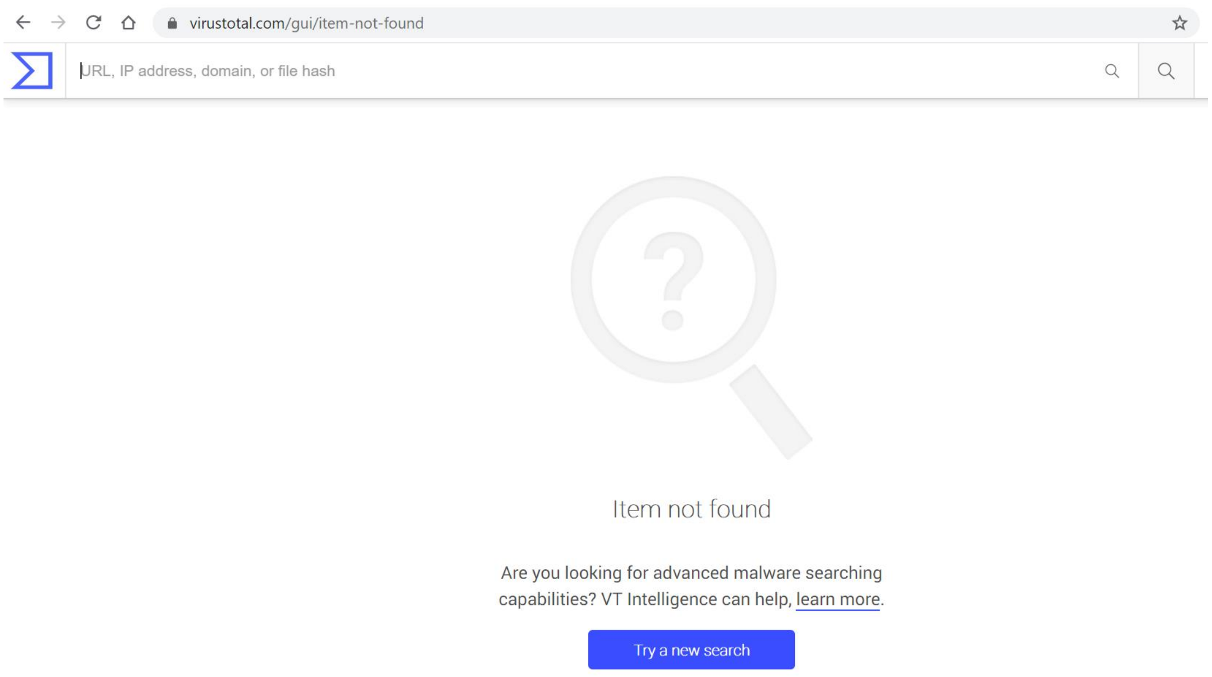

virustotal.com/gui/undefined/undefined/details

URL, IP address, domain, or file hash

The page you navigated to does not exist

The page you were looking for is out of reach.

The “item not found” page can be handled by reloading the required page again.


virustotal.com/gui/item-not-found

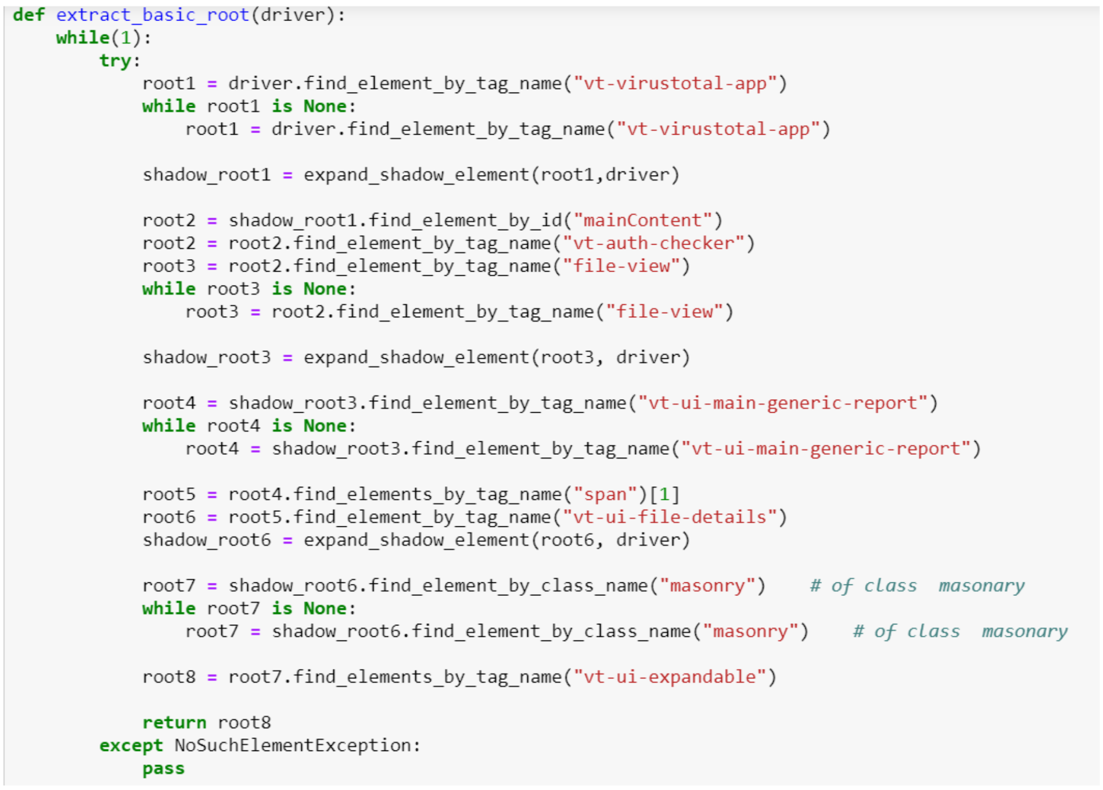
URL, IP address, domain, or file hash

Item not found

Are you looking for advanced malware searching capabilities? VT Intelligence can help, learn more.

Although these pages are handled, there are chances that more unseen pages may appear.

8. Modularization of code was a difficult task. The html orientation was observed and common layout was noted. Developed a common code for those portions and reused then multiple times in the code. With this, changing the code was easy as it needs to be changed at one place only. Also, same named variables can now be used in every function.
One such example has been noted below.



```python
def extract_basic_root(driver):
    while(1):
        try:
            root1 = driver.find_element_by_tag_name("vt-virustotal-app")
            while root1 is None:
                root1 = driver.find_element_by_tag_name("vt-virustotal-app")

            shadow_root1 = expand_shadow_element(root1,driver)

            root2 = shadow_root1.find_element_by_id("mainContent")
            root2 = root2.find_element_by_tag_name("vt-auth-checker")
            root3 = root2.find_element_by_tag_name("file-view")
            while root3 is None:
                root3 = root2.find_element_by_tag_name("file-view")

            shadow_root3 = expand_shadow_element(root3, driver)

            root4 = shadow_root3.find_element_by_tag_name("vt-ui-main-generic-report")
            while root4 is None:
                root4 = shadow_root3.find_element_by_tag_name("vt-ui-main-generic-report")

            root5 = root4.find_elements_by_tag_name("span")[1]
            root6 = root5.find_element_by_tag_name("vt-ui-file-details")
            shadow_root6 = expand_shadow_element(root6, driver)

            root7 = shadow_root6.find_element_by_class_name("masonry")    # of class  masonary
            while root7 is None:
                root7 = shadow_root6.find_element_by_class_name("masonry")    # of class  masonary

            root8 = root7.find_elements_by_tag_name("vt-ui-expandable")

            return root8
        except NoSuchElementException:
            pass
```

The function extract_basic_root navigates to the top level structure which was common to almost every property of Details tab. The code was reused during extraction of every property.

REFERENCES AND BIBLIOGRAPHY

- Kimberly Tam, Ali Feizollah, Nor Badrul Anuar, Rosli Salleh, and Lorenzo Cavallaro. 2017. “The evolution of android malware and android analysis techniques”. ACM Comput. Surv. 49, 4, Article 76 (January 2017), 41 pages. DOI: http://dx.doi.org/10.1145/3017427
- https://www.quora.com/Is-it-possible-to-bypass-a-CAPTCHA
- https://stackoverflow.com/questions/43930579/unable-to-click-on-svg-element-using-selenium
- https://medium.com/rate-engineering/a-guide-to-working-with-shadow-dom-using-selenium-b124992559f
- https://anti-captcha.com/mainpage
- https://www.seleniumeasy.com/selenium-tutorials/element-is-not-clickable-at-point-selenium-webdriver-exception
- https://medium.com/rate-engineering/a-guide-to-working-with-shadow-dom-using-selenium-b124992559f
- https://github.com/2captcha/2captcha-api-examples/blob/master/ReCaptcha%20v2%20API%20Examples/Python%20Example/2captcha_python_api_example.py
- https://blog.deathbycaptcha.com/tutorials-guides/solving-recaptcha-v2-via-api