**I - Pre-requisites for setting up a programing environment**

1) A laptop with Mac OS High Sierra or higher:

- Install Appium (<http://appium.io/>)

- Install Anaconda Python 2.7 (<https://www.anaconda.com/distribution/>)

- Install Android Studio (<https://developer.android.com/studio>)

- Install Appium Python Client (<https://github.com/appium/python-client>)

- Install Selenium (<https://selenium-python.readthedocs.io/installation.html>)

- Install Chrome Driver (<https://sites.google.com/a/chromium.org/chromedriver/downloads>)

- Install Iterm2 terminal (<https://iterm2.com/>)

- Install Homebrew (<https://brew.sh/>)

- Install Microsoft Office (<https://www.office.com>)

- Install Text Editor (e.g., vim, emacs, vscode, sublime, notepad, etc.)

- Install Chrome web browser (<https://www.google.com/chrome/>)

- Install Latex (<http://www.tug.org/mactex/>)

- Install FileZilla (<https://filezilla-project.org/>)

- Install Eclipse IDE (<https://projects.eclipse.org/releases/kepler>)

- Install ipdb (<https://pypi.org/project/ipdb/>)

- Install sshpass (<https://gist.github.com/arunoda/7790979>)

- Install pyautogui (<https://pyautogui.readthedocs.io/en/latest/>)

- Install Cisco Anyconnect client (<https://www.wichita.edu/services/its/userservices/documents/Remote_Desktop_Instructions.pdf>)

- Install Screaming Frog (<https://www.screamingfrog.co.uk/seo-spider/>)

- Install lxml (<https://lxml.de/parsing.html>)

2) A server with Ubuntu 16.04:

- Install Anaconda Python 2.7 (<https://www.anaconda.com/distribution/>)

- Install Apktool (<https://ibotpeaches.github.io/Apktool/install/>)

- Install tesseract (<https://github.com/tesseract-ocr/tesseract/wiki>)

- Install graphviz (<https://www.howtoinstall.co/en/ubuntu/xenial/graphviz>)

- Install openssh server (<https://help.ubuntu.com/lts/serverguide/openssh-server.html>)

- Install html2text (<http://manpages.ubuntu.com/manpages/trusty/man1/html2text.1.html>)

- Install Text Editor (e.g., vim, emacs, vscode, sublime, notepad, etc.)

- Install ipdb (<https://pypi.org/project/ipdb/>)

- Install Apache webserver (<https://httpd.apache.org/>)

- Install httrack (<https://www.httrack.com/>)

3) A mobile phone with Android OS 6.0:

- Make sure to turn on Developer Mode

- Have a Google Play Store account with gmail and password

**Notes**

- Make sure Bash version is at least 4.0

- Make sure the following GNU commands are available from the Bash command prompt:

*find, grep, awk, sed, dos2unix, dot, curl, wget, adb, aapt, scp, apktool, split, sshpass, tesseract, html2text, uiautomatorviewer…*

- Any text editor is sufficient to develop in Bash and Python

- Eclipse IDE (Keepler) is preferred if developing in Java with Soot framework

- gcc and GNU build system are preferred if developing in C

- Software version control (e.g., git, svn) is highly recommended for serious programers

- Openssh is recommended if working in a team of at least 2 people

- Database management system can be used but not required

- Graphical softwares (e.g., Adobe products) can be used but not required

- Latex is recommended for serious programers

- FileZilla can be used but not required

- Server secondary-storage should be large if running with thousands of apks

- Apache webserver can be replaced with a more light-weight webserver

- alias server='python -m SimpleHTTPServer' is a very useful snippet for launching a builtin python webserver

- Cisco Anyconnect client can be used for remote access from home to school if necessary

- Screaming Frog and httrack can be used but not required

- lxml can be used but not required

**II – Project structure**

The project consists of the following packages:

lai

contains all the tools for automated testing android apps via appium

lai\_web

contains all the tools for automated testing websites via selenium

analyzer

contains all the tools related to account deletion functionality for android apps

analyzer\_web

contains all the tools related to account deletion functionality for the web

analyzer\_combiner

contains all the tools which combine results from the two packages analyzer and analyzer\_web

pystatparser

contains a parser for natural language processing

nlp

contains all the tools which use pystatparser for analyzing the semantics in natural language processing

duc

contains all the tools to be used with Soot framework

calc

contains all the templates to be used for data calculation

latex

contains all the templates to be used for latex (please ask professor Shan)

clang

contains all the tools implemented in C language

**III – Notations**

**./** run an executable bash script

< > some user input

[ X | Y ] choose either X or Y

Z software tool Z is developed or customized in-house

\_\_\_ optional

**IV - Software tools and usages**

**apkdl.sh**

|  |  |
| --- | --- |
| synopsis | **./apkdl.sh** |
| description | scrape all the android package names from a website |
| options | none  the input website is already hardcoded as apk-dl.com  the script will return a list of android package names |
| dependencies | curl  awk |
| packages | lai\_web |
| examples | **A screen shot of a social media post  Description automatically generated** |

**apk2web.sh**

|  |  |
| --- | --- |
| synopsis | **./apk2web.sh <apkfile>** |
| description | find the equivalent website address for an apk file by looking into the manifest |
| options | <apkfile>  an apk file  given an apk file, the script will return the equivalent website address |
| dependencies | aapt  dos2unix  grep |
| packages | lai\_web |
| examples | **A picture containing device, object  Description automatically generated** |

**apk2url.sh**

|  |  |
| --- | --- |
| synopsis | **./apk2url.sh <packagelist> auto** |
| description | find the equivalent website addresses for a list of android package names via google |
| options | <packagelist>  a text file containing android package names  the auto flag is optional:  if present, the script will automatically pick the first website address from google candidate search results  if not present, the script will let testers pick a website manually from the google candidate search results |
| dependencies | url.py |
| packages | lai\_web |
| examples | **A screenshot of a cell phone  Description automatically generated**  **A picture containing object  Description automatically generated** |

**auto.sh**

|  |  |
| --- | --- |
| synopsis | **./auto.sh [ <apklist-directory> <server-username> <server-password> <wait-time-in-secs> collect**  **| <apkdir> apk2web ]** |
| description | an automatic script to collect apks and websites without any manual intervention |
| options | collect  automatically go to google play store, download apks, and upload them to a dedicated server  <apklist-directory>  a directory which contains files whose contents are android package names  <server-username>  ssh username  <server-password>  ssh password  <wait-time-in-secs>  time to wait between each batch of apk files finish uploading to the server  apk2web  automatically find equivalent websites from a directory containing all apk files pulled from the mobile phone  <apkdir>  a directory which contains all apk files pulled from the android mobile phone |
| dependencies | lac\_batch.sh  upload.sh  apk2web.sh |
| packages | lai\_web |
| notes | appium is required to run the script |
| examples | **A picture containing clock, object, indoor  Description automatically generated**  **A picture containing object  Description automatically generated** |

**batch.sh**

|  |  |
| --- | --- |
| synospis | **./batch.sh <apk-directory> [ install | uninstall | remove | signup | login | delete | manual ]** |
| description | a tool for testers to work with a directory containing apk files (working in batch mode) |
| options | <apk-directory>  a directory which contains apk files  choose one of the following options:  install  install all the apk files in the directory on the android mobile phone  uninstall  uninstall all the apk files in the directory from the android mobile phone  remove  remove the <apk-directory> and all the output directories *\_out* and error directories *\_err*, ready for next batch  signup  automatically signup a user account for each of the apk file in <apk-directory>  login  automatically login the user account for each of the apk file in <apk-directory>  delete  automatically delete the user account for each of the apk file in <apk-directory>  manual  manual intervention from testers, break into *ipdb* console and finish any of the options above  testers can also use function *dump\_text* builtin the tool to do screen scraping on the android phone |
| dependencies | adb  aapt  grep  manual.py  delete.py  login.py  signup.py |
| packages | lai  lai\_web |
| notes | appium is required to run the script  depending on the chosen option, the tool will save log files into *\_out* and *\_err* directories, from which statistical analysis can be performed |
| examples | **A screen shot of a clock  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A picture containing indoor  Description automatically generated**  **A close up of a clock  Description automatically generated** |

**single.sh**

|  |  |
| --- | --- |
| synopsis | **./single.sh <apkfile> [ download | signup | install | uninstall | login | delete | launch | search | manual ]** |
| description | a tool for testers to work with one single apk file |
| options | <apkfile>  an apk file  choose one of the following options:  download  download the <apkfile> from the server  signup  automatically signup a user account for the <apkfile>  install  install the <apkfile> on the android mobile phone  uninstall  uninstall the <apkfile> from the android mobile phone  login  automatically login the user account for the <apkfile>  delete  automatically delete the user account for the <apkfile>  launch  launch the <apkfile> and fire random events with monkey tool  search  automatically search apk file for a given android package name by automating google web browser  manual  manually break into the ipdb console |
| dependencies | adb  monkey  scp  signup.py  manual.py  delete.py  login.py |
| packages | lai  lai\_web |
| notes | appium is required to run the script  this script is similar to batch.sh except it works on only 1 apk file at a time, can be used in tandem with batch.sh  it also provides some extra commands not in batch.sh |
| examples | **A close up of a clock  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated** |

**pm.sh**

|  |  |
| --- | --- |
| synopsis | **./pm.sh [ pull | uninstall | list | count | <package-file> search | <package-name> pull ]** |
| description | a package manager tool to help testers manage android packages efficiently |
| options | pull  pull all non-system packages from the android phone to your laptop  uninstall  uninstall all non-system packages currently residing in the android phone  list  list all non-system packages in the android phone  count  count the number of packages after pulling  search  automatically search for android packages via google browser in the phone  <package-file>  a text file which contain android packages  pull  pull a specific package from the android phone to your laptop  <package-name>  an android package |
| dependencies | dos2unix  sed  cut  adb |
| packages | lai |
| examples | **A screen shot of a computer  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated** |

**minus\_operation.sh**

|  |  |
| --- | --- |
| synopsis | **./minus\_operation.sh <gmail> <gpassword>** |
| description | auto login google play store website (*My apps* page) and calculate the difference between all the android packages listed there versus all the android packages listed in the mobile phone |
| options | <gmail>  google email address to login play store  <gpassword>  google password to login play store |
| dependencies | my\_apps.py  awk  dos2unix |
| packages | lai\_web |
| notes | a compatible *chromedriver* is required for using with Chrome web browser |
| examples | **A close up of a logo  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated** |

**lac\_batch.sh**

|  |  |
| --- | --- |
| synopsis | **./lac\_batch.sh [ <gmail> <gpassword> minus**  **| <./minus/laXX> install**  **| <package-name> <app-email> <app-password> clean**  **| pull**  **| uninstall**  **| <./minus/laXX> remove ]** |
| description | when users uninstall apps on their android phones, they may have forgotten to delete their user accounts for those apps,  the script can help to automatically cleanup those accounts, it also other functions to manage apk files |
| options | minus  see minus\_operation.sh  <gmail>  google email address to login play store  <gpassword>  google password to login play store  install  automatically searching the play store for packages provided in <./minus/laXX> and install them on android phone  <./minus/laXX>  a text file which contains a list of android package names  clean  given <app-email> and <app-password>, this will automatically delete the user account on a given <package-name>  <package-name>  the package-name of an app currently residing on android phone  <app-email>  the email that user has provided when signing up with an app on android phone  <app-password>  the password that user has provided when signing up with an app on android phone  pull  see pm.sh  uninstall  see pm.sh  remove  remove all log directories *\_out* and *\_err*, ready for the next batch  <./minus/laXX>  a text file which contains a list of android package names |
| dependencies | minus\_operation.sh  gsplit  store.py  dos2unix  adb  grep  lac\_adf.sh  cleanup.py |
| packages | lai\_web  lai |
| notes | appium is required to run the script |
| examples | **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a social media post  Description automatically generated** |

**upload.sh**

|  |  |
| --- | --- |
| synopsis | **./upload.sh <ssh-username> <ssh-password>** |
| description | this little snippet will upload all apk files in *pull* directory and upload to a specific directory on the dedicated server,  it can replace FileZilla |
| options | <ssh-username>  ssh username to login the dedicated server  <ssh-password>  ssh password to login the dedicated server |
| dependencies | sshpass |
| packages | lai\_web |
| notes | a pull directory which contains all apk files must exist before running this script (e.g., after running *pm.sh pull*) |
| examples | **A close up of a screen  Description automatically generated** |

**lac\_adf.sh**

|  |  |
| --- | --- |
| synopsis | **./lac\_adf <package-name>** |
| description | check on-the-fly whether an existing package on the android mobile device has user account deletion function or not (without using any natural language processing technique) |
| options | <package-name>  an android package on the mobile device  the script will return:  *AD* means account deletion  *Not AD* means not account deletion |
| dependencies | adb  dos2unix  grep  sed |
| packages | lai\_web |
| notes | to know which packages are currently installed on the device, see pm.sh |
| examples |  |

**stats2.sh**

|  |  |
| --- | --- |
| synopsis | **./stats2.sh <log-directory>** |
| description | gather statistics for each android app after running batch.sh script |
| options | <log-directory>  a directory which contains all the log files stored under *\_out* and *\_err* directories |
| dependencies | grep |
| packages | lai |
| notes | make sure to run batch.sh before running this script  the results can be redirected to a csv file and export to ms excel for further analyses |
| examples | **A screen shot of a computer  Description automatically generated**  **A screenshot of a computer  Description automatically generated** |

**stats2\_web.sh**

|  |  |
| --- | --- |
| synopsis | **./stats2.sh <weblog-directory>** |
| description | gather statistics for each website after running batch.sh script |
| options | <weblog-directory>  a directory which contains all the log files for websites after running adf.sh script |
| dependencies | grep |
| packages | lai |
| notes | make sure to run adf.sh for the web first before running this script  the results can be redirected to a csv file and export to ms excel for further analyses |
| examples | **A screen shot of a computer  Description automatically generated**  **A screenshot of a computer  Description automatically generated** |

**adf.sh**

|  |  |
| --- | --- |
| synopsis | **./adf.sh [ <apkfile> | <website> ]** |
| description | build a strings-to-screen mapping model to determine whether a given apk file has user account deletion functionality |
| options | <apkfile>  an apk file  <website>  a website which is downloaded as a webpage complete from google chrome  the script returns a *mapping\_model.dot* file for the corresponding apk or website in *output* directory for further analysis |
| dependencies | apktool  find  awk  xargs  tesseract  grep  sed  semantics.py |
| packages | analyzer  analyzer\_web |
| notes | options are available depending on which package the script is running from  the script depends on tesseract to convert images to text  the script depends on semantics.py which use natural language processing to check the semantics of a phrase  further analysis can be performed on *mapping\_model.dot* file by using stats.sh script |
| examples | **A close up of a logo  Description automatically generated**  **A screen shot of a computer  Description automatically generated**    **A screen shot of a computer  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A close up of a screen  Description automatically generated**  **A screen shot of a computer  Description automatically generated** |

**parallel.sh**

|  |  |
| --- | --- |
| synopsis | **./parallel.sh [ <apk-directory> | <web-directory> ] <max-num-of-cores>** |
| description | an efficient way to run adf.sh in a multiprocessing environment |
| options | <apk-directory>  a directory which contains all downloaded apk files  <web-directory>  a directory which contains all downloaded websites  <max-num-of-cores>  a maximum number of cpu cores assigned to run this task in parallel |
| dependencies | adf.sh  xargs |
| packages | analyzer  analyzer\_web |
| notes | <max-num-of-cores> must be less than or equal to the actual number of cpu cores the server has |
| examples | **A screen shot of a computer  Description automatically generated**  **A screen shot of a computer  Description automatically generated**    **A screen shot of a computer  Description automatically generated** |

**adf\_fp.sh**

|  |  |
| --- | --- |
| synopsis | **./adf\_fp.sh [ <apkfile> | <website> ]** |
| description | find not account deletion strings (not-ad) for an apk or a website using natural language processing |
| options | <apkfile>  an apk file  <website>  a website which is downloaded as a webpage complete from google chrome  if not-ad is found, the script will save it in *not\_ad\_strings.txt* for an equivalent apk or website in *output* directory  otherwises, it will output nothing |
| dependencies | grep  sed  xargs  semantics.py |
| packages | analyzer  analyzer\_web |
| notes | must run adf.sh or parallel.sh first before running this script |
| examples | **A picture containing clock, monitor, object, indoor  Description automatically generated**  **A close up of a screen  Description automatically generated**  **A screen shot of a computer  Description automatically generated**  **A picture containing object, indoor  Description automatically generated**  **A screen shot of a computer  Description automatically generated**  **A picture containing indoor  Description automatically generated** |

**parallel\_fp.sh**

|  |  |
| --- | --- |
| synopsis | **./parallel.sh [ <apk-directory> | <web-directory> ] <max-num-of-cores>** |
| description | an efficient way to run adf\_fp.sh in a multiprocessing environment |
| options | <apk-directory>  a directory which contains all downloaded apk files  <web-directory>  a directory which contains all downloaded websites  <max-num-of-cores>  a maximum number of cpu cores assigned to run this task in parallel |
| dependencies | adf.sh  xargs |
| packages | analyzer  analyzer\_web |
| notes | <max-num-of-cores> must be less than or equal to the actual number of cpu cores the server has  this tool is similar to parallel.sh |
| examples | **A picture containing indoor  Description automatically generated**  **A picture containing indoor  Description automatically generated** |

**stats.sh**

|  |  |
| --- | --- |
| synopsis | **./stats.sh [ time | dot | svg | fp ]** |
| description | a tool to extract statistical information from log files |
| options | time  output time in seconds for each app (apk or website)  dot  copy all valid *mapping\_models.dot* files to *analyses/dot* directory (see adf.sh for more information)  svg  convert those *mapping\_model.dot* files to *svg* files  fp  copy all *not\_ad\_strings.txt* files to *analyses/fp* directory (see adf\_fp.sh for more information) |
| dependencies | find  sed  tail  grep  dot  wc |
| packages | analyzer  analyzer\_web  lai **(deprecated)**  lai\_web **(deprecated)** |
| notes | must run adf.sh or parallel.sh or adf\_fp.sh or parallel\_fp.sh first before running this script  time outputs can be redirected to a csv file and export to ms excel for further analyses |
| examples | **A close up of a screen  Description automatically generated**    **A picture containing object, clock, indoor, monitor  Description automatically generated**  **A picture containing clock, showing, object, indoor  Description automatically generated**  **A close up of text on a white background  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a computer  Description automatically generated** |

**concat.sh**

|  |  |
| --- | --- |
| synopsis | **./concat.sh <apk-dot-directory> <website-dot-directory>** |
| description | combine adf results from apks and websites together |
| options | <apk-dot-directory>  the directory which contains all the dot files for apks after running adf.sh or parallel.sh tools  in package analyzer  <website-dot-directory>  the directory which contains all the dot files for websites after running adf.sh or parallel.sh tools  in package analyzer\_web  after combining dot files, the script will convert them into svg files, also it saves the common dot files into *both.txt* |
| dependencies | head  tail  sed  cp  dot |
| packages | analyzer\_combiner |
| notes | must run adf.sh or parallel.sh first before running this script (in both package analyzer and analyzer\_web)  in svg graph, dotted lines represent paths coming from the website, normal lines are from the apk |
| examples | **A picture containing indoor, object  Description automatically generated**  **A close up of a screen  Description automatically generated**  **A close up of a screen  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated** |

**semantics.py**

|  |  |
| --- | --- |
| synopsis | **\_\_DEV\_\_=1 python semantics.py <paragraph> .\*<verb>.\*<noun>.\*** |
| description | check the semantics of a paragraph against a special regular expression using natural language processing techniques |
| options | <paragraph>  a collection of sentences preferred to be in English  <verb>  an english verb  <noun>  an english noun  \_\_DEV\_\_=1 is optional:  if present, a constituent tree will be printed and vice versa  the script will return:  (‘TP’, ‘’) means the paragraph’s true meaning has been understood as intended by regex .\*<verb>.\*<noun>.\*  (‘FP’, some reason) means the paragraph’s true meaning has not been understood by regex .\*<verb>.\*<noun>.\* |
| dependencies | pystatparser  nltk |
| packages | nlp |
| notes | on linux or unix system, it is highly recommended to set PATH variable pointing to this package so that the command can be used through the system and as a dependency to other scripts  currently the lab server file /etc/bash.bashrc is set up as : export PATH=$PATH:/home/hoang/Documents/Achilles/py |
| examples | **A picture containing indoor, black  Description automatically generated**  **A picture containing object, indoor, clock  Description automatically generated** |

**context.py**

|  |  |
| --- | --- |
| synopsis | **python context.py <document>** |
| description | an attempt to classify a document content on the difficulty levels of the user account deletion process using natural language processing techniques |
| options | <document>  a textfile  the script should return one of the following values in increasing difficulty order:  -1: no context match  0: there is an account deletion function  1: the account deletion function asks for reasons  2: the account deletion function asks to enter some inputs  3: the account deletion function asks to confirm email address  4: the account deletion function asks to contact customer service |
| dependencies | semantics.py |
| packages | nlp |
| examples | **A picture containing indoor, object  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated** |

**To pull single apks manually from android device to computer:**

1. **Firstly find the path of packages:**

adb shell pm list packages -f -3

Output :

package:/data/app/com.foap.android-1/base.apk=com.foap.android

package:/data/app/com.adventure.skyticket-1/base.apk=com.adventure.skyticket

package:/data/app/com.getsomeheadspace.android-2/base.apk=com.getsomeheadspace.android

package:/data/app/com.mercariapp.mercari-1/base.apk=com.mercariapp.mercari

package:/data/app/com.multibrains.taxi.passenger.tirhal-1/base.apk=com.multibrains.taxi.passenger.tirhal

1. **To pull a specific package- Example: com.mercariapp.mercari-1:**

adb pull /data/app/com.mercariapp.mercari-1/base.apk /Users/preethis/Desktop

**apktool to extract retention period from apps**:

|  |  |
| --- | --- |
| synopsis | **Change corresponding app folder names in the source code and run** |
| description | A program to extract retention period patterns |
| options | None |
| dependencies | semantics.py, apktool |
| packages | None |
| notes | This program converts or reverse engineer’s apk files to decoded form and creates an output directory containing decoded folders and uses ‘strings.xml’ file to extract retention period patterns |

**Web tool to extract retention period from websites:**

|  |  |
| --- | --- |
| synopsis | **python testing.py <foldername.apk>** |
| description | A program to extract retention period patterns |
| options | None |
| dependencies | semantics.py |
| packages | None |
| notes | This program converts folder containing webpages to texts and creates an output directory containing converted text folders then uses these files to extract retention period patterns |

**url.py**

|  |  |
| --- | --- |
| synopsis | **python url.py <android-package-file>** |
| description | a little snippet to help testers pick the equivalent websites for android packages by listing the top ten results from google |
| options | <android-package-file >  a text file containing android package names |
| dependencies | urlparse  googlesearch |
| packages | lai\_web |
| notes | output results will be saved in an csv file and can export to excel for further analysis |
| examples | **A screen shot of a smart phone  Description automatically generated** |

**store.py**

|  |  |
| --- | --- |
| synopsis | **python store.py <android-package-names-via-stdin>** |
| description | automatically goes to google play store, search for apks and install them to the android phone |
| options | <android-package-names-via-stdin>  a list of android package names seperated by new line \n from stdin |
| dependencies | commons.py |
| packages | lai\_web |
| notes | <android-package-names-via-stdin> can be replaced by a redirection from cat command  appium is required to run this script |
| examples | **A close up of a logo  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated** |

**my\_apps.py**

|  |  |
| --- | --- |
| synopsis | **python my\_apps.py <gmail> <gpassword>** |
| description | automatically navigate and scrape html from *My apps* page on google play store website via chrome browser |
| options | <gmail>  google account email to login play store  <gpassword>  google account passowrd to login play store  the script will scrape all html content from *My apps* page of google play store |
| dependencies | common\_web.py  python selenium  chrome driver |
| notes | make sure to install python selenium, chrome driver, chrome browser before running this script |
| packages | lai\_web |
| examples | **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated** |

**manual.py**

|  |  |
| --- | --- |
| synopsis | **python manual.py [ <apkfile> | <website-address> ]** |
| description | automatically launch an apkfile or a website then break into its ipdb console for manual intervention tasks |
| options | <apkfile>  an apk file  <website-address>  a website address |
| dependencies | commons.py  commons\_web.py  ipdb |
| packages | lai  lai\_web |
| notes | appium is required if running the script from package lai  selenium, chrome driver, and chrome browser are required if running the script from package lai\_web |
| examples | **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated** |

**signup.py**

|  |  |
| --- | --- |
| synopsis | **python signup.py [ <apkfile> | <website-address> ]** |
| description | an attempt to automatically signup a user account via the android application or its equivalent website |
| options | <apkfile>  an apk file  <website-address>  a website address |
| dependencies | commons.py  commons\_web.py |
| packages | lai  lai\_web |
| notes | appium is required if running the script from package lai  selenium, chrome driver, and chrome browser are required if running the script from package lai\_web  user account information is hardcoded in commons.py or commons\_web.py |
| examples | **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a computer screen  Description automatically generated** |

**login.py**

|  |  |
| --- | --- |
| synopsis | **python login.py [ <apkfile> | <website-address> ]** |
| description | an attempt to automatically login a user account via the android application or its equivalent website |
| options | <apkfile>  an apk file  <website-address>  a website address |
| dependencies | commons.py  commons\_web.py |
| packages | lai  lai\_web |
| notes | appium is required if running the script from package lai  selenium, chrome driver, and chrome browser are required if running the script from package lai\_web  user account information is hardcoded in commons.py or commons\_web.py |
| examples | **A screenshot of a cell phone  Description automatically generated**  **A screen shot of a smart phone  Description automatically generated**  **A screenshot of a computer screen  Description automatically generated** |

**delete.py**

|  |  |
| --- | --- |
| synopsis | **python delete.py [ <apkfile> <dotfile> | <website-address> ]** |
| description | an attempt to automatically delete a user account via the android application or its equivalent website |
| options | <apkfile>  an apk file  <dotfile>  a dot file generated by adf.sh to guarantee that the account deletion functionlity exists on the apk  <website-address>  a website address |
| dependencies | commons.py  commons\_web.py |
| packages | lai  lai\_web |
| notes | first make sure that there exists a user account with an android application or its equivalent website, possibly by running signup.py  appium is required if running the script from package lai, also must run adf.sh or parallel.sh first to have dot files  selenium, chrome driver, and chrome browser are required if running the script from package lai\_web  user account information is hardcoded in commons.py or commons\_web.py |
| examples | **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated** |

**cleanup.py**

|  |  |
| --- | --- |
| synopsis | **python cleanup.py <package-name> <app-email> <app-password>** |
| description | an attempt to automatically cleanup a user account for a certain android app when the user uninstalls it but forgets to do so |
| options | <package-name>  the android package name which currently resides on the android device  <app-email>  the user email which the user uses to login the app  <app-password>  the user password which the users uses to login the app |
| dependencies | commons.py |
| packages | lai  lai\_web |
| notes | appium is required to run the script  make sure that the <package-name> exists on the device, this can be checked by using pm.sh |
| examples | **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a computer  Description automatically generated** |

**commons.py**

|  |  |
| --- | --- |
| synopsis | **from commons import <some-functionality>** |
| description | a python library to interact with the android mobile device via appium |
| options | <some-functionality> can be one of the followings:  external constants  \_UNINSTALL\_FLAG, EMAIL, PASSWORD, USERNAME, BIRTHDAY, FIRSTNAME, LASTNAME,  PHONENUMBER, HEIGHT, WEIGHT, AGE, GENDER, LOCATION, ZIPCODE, SSN, DLN, CREDITCARD  external variable  session  functions:  init  **deprecated**  init2  auto launch an android app using the apk file and get a session ready  init3  auto launch an android app using the package name and get a session ready  unintall  uninstall the android app by using the apk file  find  find a widget by its text  find\_edits  find all the editable widgets  find\_clickable\_textviews  find all the textview widgets which are clickable  scroll\_to  scroll to a specific widget identified matching its text  getClickableWidgets  get all clickable widgets on the screen  filterClass  **deprecated**  filterKeyword  **deprecated**  checkWidgetCount  **deprecated**  Init  **deprecated**  swipe\_up  auto swiping up the screen  swipe\_left  auto swiping to the left of the screen  dump\_text  dump all the textual information of current active screen |
| dependencies | appium  time  re  csv  os  selenium |
| packages | lai  lai\_web |
| notes | none |
| examples | none |

**commons\_web.py**

|  |  |
| --- | --- |
| synopsis | **from commons\_web import <some-functionality>** |
| description | a python library to interact with the chrome browser via selenium |
| options | <some-functionality> can be one of the followings:  external constants  \_UNINSTALL\_FLAG, EMAIL, PASSWORD, USERNAME, BIRTHDAY, FIRSTNAME, LASTNAME,  PHONENUMBER, HEIGHT, WEIGHT, AGE, GENDER, LOCATION, ZIPCODE, SSN, DLN, CREDITCARD  external variable  session  functions:  init  auto launch a website address (url) and get the session ready  dump\_text  dump all textual information of a webpage  dump\_html  dump all html information of a webpage  find\_clickable  find a clickable widget if its text matches a specific regular expression pattern  find\_editables  find all editable widgets on current webpage  fill\_edits  automatically fill all editable widgets with information provided via a dictionary data type  nicely\_quit  nicely quit the web session  signal\_handler  nicely handle external signals  click  click on a provided web element  shutdown  shutdown the session  save\_as  mimic the *save as* functionality of a chrome web browser  scroll\_infinite  automatically scroll the webpage until its end is reached |
| dependencies | selenium  time  html2text  re  sys  pyautogui |
| packages | lai\_web |
| notes | none |
| examples | none |

**random\_scraper.py**

|  |  |
| --- | --- |
| synopsis | **python random\_scraper.py <apkfile>** |
| description | a tool which triggers random events on the android app and scrape all the screen textual information |
| options | <apkfile>  an apk file |
| dependencies | commons.py  monkey |
| packages | lai |
| notes | appium is required to run the script, must set --relaxed-security option  the number of events and iteration to be fired are hardcoded in the script |
| examples | A picture containing object, clock, device  Description automatically generated  A screenshot of a cell phone  Description automatically generated |

**capture.py**

|  |  |
| --- | --- |
| synopsis | **python capture.py <website-address>** |
| description | a tool which mimics the save as function in google chrome web browser |
| options | <website-address>  a website address |
| dependencies | commons\_web.py |
| packages | lai\_web |
| notes | selenium, web driver, and chrome browser are required before running this script  make sure to have pyautoui library installed |
| examples | **A screenshot of a cell phone  Description automatically generated** |

**DefUseChain.java**

|  |  |
| --- | --- |
| synopsis | **see examples** |
| description | a tool for def-use-chain analysis using Soot Framework |
| options | see examples |
| dependencies | see examples |
| packages | see examples |
| notes | Eclipse (Keepler) is currently used for running this script |
| examples | **A screenshot of a computer  Description automatically generated**  **A screenshot of a social media post  Description automatically generated** |

**pystatparser**

|  |  |
| --- | --- |
| synopsis | **from stat\_parser import Parser** |
| description | a python package for parsing natural languages |
| options |  |
| dependencies | github |
| packages | pystatparser |
| notes | this open-source python package has been readjusted to work properly with python 2.7 |
| examples | **A picture containing indoor  Description automatically generated** |

**auth**

|  |  |
| --- | --- |
| synopsis | **#include “auth.h”** |
| description | a small c-library built with curl to bypass basic authorization |
| options | functions:  basic\_auth  bypass basic authorization  basic\_get  simulate a GET request bypassing basic authorization |
| dependencies | curl  Base64Encode  Base64Decode  crypto |
| packages | clang |
| notes | at the time of writing, httrack was a popular opensource website copier but it did not pass basic authorization  this library can be compiled as a dynamic library .so to be used in complement with httrack |
| examples | **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a social media post  Description automatically generated** |