目录

抓包代理利器：mitmproxy

- 最新版本：`v0.8`
- 更新时间：`20201231`

简介

介绍主要用于抓包领域的代理工具mitmproxy，尤其是常用的命令行版的mitmdump。先对mitmproxy概述，再介绍mitmdump的下载和安装。包括Mac和Win中如何安装和常见问题。接下来介绍如何使用mitmdump，包括核心的通用逻辑，即先电脑端启动mitmdump代理，再去移动端初始阿虎安装mitmproxy的根证书ssl代理证书文件，其中总结了各种iOS和安卓手机在安装根证书时候的坑和问题及解决办法。再去介绍如何给移动端中WiFi去设置代理。总结了常见的问题，比如No module named yaml、traffic is not passing through mitmproxy等等。还有其他方面，比如代码中调用控制mitmdump的运行、win中如何给mitmdump的python打包成exe。最后附上help语法供需要时查阅。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- crifan/crawler_proxy_tool_mimproxy: 抓包代理利器：mitmproxy

如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

在线浏览

- 抓包代理利器：mitmproxy book.crifan.com
- 抓包代理利器：mitmproxy crifan.github.io

离线下载阅读

- 抓包代理利器：mitmproxy PDF
- 抓包代理利器：mitmproxy ePub
- 抓包代理利器：mitmproxy Mobi

版权说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

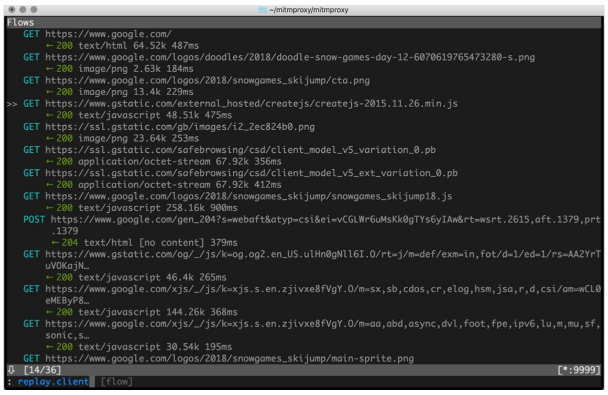
更多其他电子书

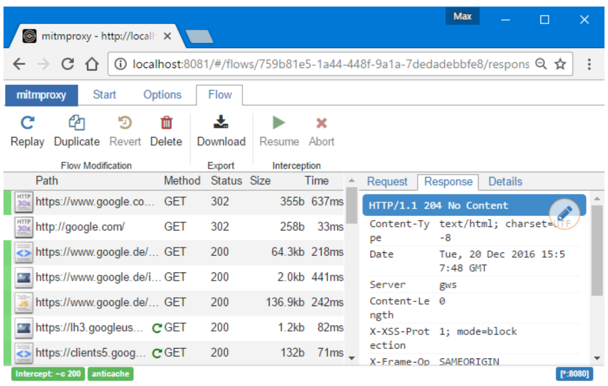
本人 `crifan` 还写了其他 `100+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

mitmproxy概述

- `mitmproxy`
  - 名词解析
    - `mitmproxy` = `mitm` 的 `proxy`
      - `mitm` = `MITM` = `Man-In-The-Middle`
        - 直译：`人在中间`
          - 在中间 ->
            - 首先要确保原先网络请求能继续
              - 所以就是代理的功能：**正常转发**原有网络请求
            - 但也可以干很多事情
              - 比如
                - **记录**、**保存**网络请求
                - **拦截**（符合特定规则的）网络请求
                - （甚至）**篡改**、**伪造**成新的合法的或不合法的网络请求
        - 相关：往往也被叫做
          - `Man-In-The-Middle attack` = `中间人攻击`
  - `mitmproxy` 是一套工具的总称，包含
    - `mitmproxy`：交互式命令行工具
      - 是什么=一句话概述
        - 英文
          - mitmproxy is an interactive, SSL/TLS-capable intercepting proxy with a console interface for HTTP/1, HTTP/2, and WebSockets
        - 中文
          - mitmproxy是一个代理工具
            - 功能和特点
              - 交互式的
              - 支持https拦截
              - 支持协议：`HTTP/1`、`HTTP/2`、`WebSockets`
                - 产品形态：控制台console中显示交互界面
      - 长什么样=截图

                - 

                  ```
                  ~/mitmproxy/mitmproxy
                  Flows
                     GET https://www.google.com/
                         ← 200 text/html 64.52k 487ms
                     GET https://www.google.com/logos/doodles/2018/doodle-snow-games-day-12-6070619765473280-s.png
                         ← 200 image/png 2.63k 184ms
                     GET https://www.google.com/logos/2018/snowgames_skijump/cta.png
                         ← 200 image/png 13.4k 229ms
                  >> GET https://www.gstatic.com/external_hosted/createjs/createjs-2015.11.26.min.js
                         ← 200 text/javascript 48.51k 475ms
                     GET https://ssl.gstatic.com/gb/images/i2_2ec824b0.png
                         ← 200 image/png 23.64k 253ms
                     GET https://ssl.gstatic.com/safebrowsing/csd/client_model_v5_variation_0.pb
                         ← 200 application/octet-stream 67.92k 356ms
                     GET https://ssl.gstatic.com/safebrowsing/csd/client_model_v5_ext_variation_0.pb
                         ← 200 application/octet-stream 67.92k 412ms
                     GET https://www.google.com/logos/2018/snowgames_skijump/snowgames_skijump18.js
                         ← 200 text/javascript 258.16k 900ms
                     POST https://www.google.com/gen_204?s=webaft&atyp=csi&ei=vCGLWr6uMsKk0gTYs6yIAw&rt=wsrt.2615,aft.1379,prt
                          .1379
                          ← 204 text/html [no content] 379ms
                     GET https://www.gstatic.com/og/_/js/k=og.og2.en_US.ulHn0gNll6I.O/rt=j/m=def/exm=in,fot/d=1/ed=1/rs=AA2YrT
                         uVOKajN…
                         ← 200 text/javascript 46.4k 265ms
                     GET https://www.google.com/xjs/_/js/k=xjs.s.en.zjivxe8fVgY.O/m=sx,sb,cdos,cr,elog,hsm,jsa,r,d,csi/am=wCL0
                         eMEByP8…
                         ← 200 text/javascript 144.26k 368ms
                     GET https://www.google.com/xjs/_/js/k=xjs.s.en.zjivxe8fVgY.O/m=aa,abd,async,dvl,foot,fpe,ipv6,lu,m,mu,sf,
                         sonic,s…
                         ← 200 text/javascript 30.54k 195ms
                     GET https://www.google.com/logos/2018/snowgames_skijump/main-sprite.png
                  ⇩ [14/36]                                                                                          [*:9999]
                  : replay.client  [flow]
                  ```
        - `mitmweb` ： 基于命令行的带UI界面
            - 可以理解为： **网页版的mitmproxy**
            - 是什么=一句话描述
                - mitmweb is a web-based interface for mitmproxy
            - 长什么样=截图
                - 

                  localhost:8081/#/flows/759b81e5-1a44-448f-9a1a-7dedadebbfe8/respons

Path	Method	Status	Size	Time
https://www.google.co…	GET	302	355b	637ms
http://google.com/	GET	302	258b	33ms
https://www.google.de/…	GET	200	64.3kb	218ms
https://www.google.de/i…	GET	200	2.0kb	441ms
https://www.google.de/…	GET	200	136.9kb	242ms
https://lh3.googleus…	GET	200	1.2kb	82ms
https://clients5.goog…	GET	200	132b	71ms

                  ```
                  HTTP/1.1 204 No Content
                  Content-Ty   text/html; charset=UTF
                  pe           -8
                  Date         Tue, 20 Dec 2016 15:5
                               7:48 GMT
                  Server       gws
                  Content-Le   0
                  ngth
                  X-XSS-Prot   1; mode=block
                  ection
                  X-Frame-Op   SAMEORIGIN
                  ```
        - `mitmdump` ： 命令行版本
            - 是什么=一句话描述
                - mitmdump is the command-line version of mitmproxy. Think tcpdump for HTTP
            - 类比
                - `mitmdump` 之于 `mitmproxy`
                    - 就像
                        - `tcpdump` 之于 `HTTP`
            - 可以理解为
                - 命令行版本的 `Charles` / `Fiddler`
- 主要用途：实现对网页、app的抓包
- 网址
    - 官网文档
        - Introduction
            - https://docs.mitmproxy.org/stable/
    - GitHub
        - mitmproxy/mitmproxy: An interactive TLS-capable intercepting HTTP proxy for penetration testers and software developers
            - https://github.com/mitmproxy/mitmproxy

mitmdump

`mitmdump` 是 `mitmproxy` 的命令行版本。

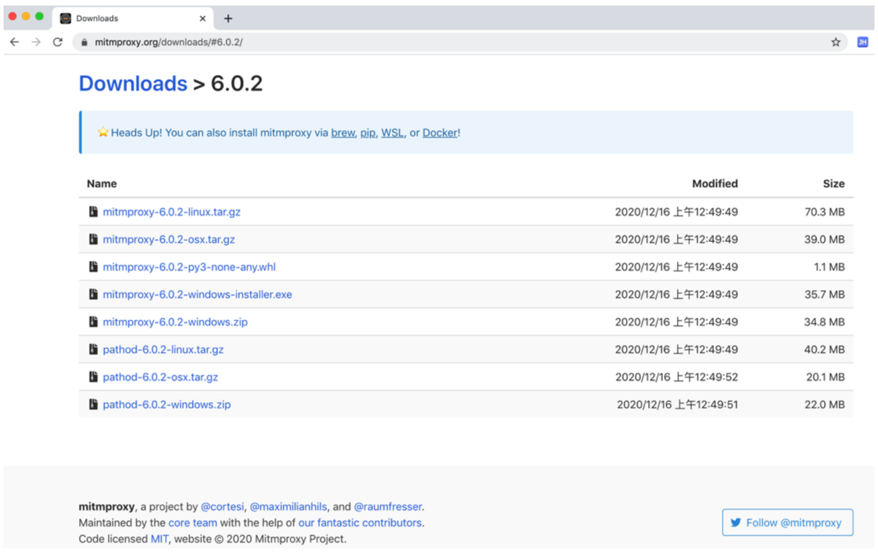
`mitmproxy` 和 `mitmdump` 的用法和逻辑基本一致。

此处主要介绍 `mitmdump` 的使用。

下载mitmproxy

官网下载地址：

- Downloads
    - https://mitmproxy.org/downloads/#6.0.2/



mitmproxy.org/downloads/#6.0.2/

Downloads > 6.0.2

⭐ Heads Up! You can also install mitmproxy via brew, pip, WSL, or Docker!

Name	Modified	Size
mitmproxy-6.0.2-linux.tar.gz	2020/12/16 上午12:49:49	70.3 MB
mitmproxy-6.0.2-osx.tar.gz	2020/12/16 上午12:49:49	39.0 MB
mitmproxy-6.0.2-py3-none-any.whl	2020/12/16 上午12:49:49	1.1 MB
mitmproxy-6.0.2-windows-installer.exe	2020/12/16 上午12:49:49	35.7 MB
mitmproxy-6.0.2-windows.zip	2020/12/16 上午12:49:49	34.8 MB
pathod-6.0.2-linux.tar.gz	2020/12/16 上午12:49:49	40.2 MB
pathod-6.0.2-osx.tar.gz	2020/12/16 上午12:49:52	20.1 MB
pathod-6.0.2-windows.zip	2020/12/16 上午12:49:51	22.0 MB

mitmproxy, a project by @cortesi, @maximilianhils, and @raumfresser.
Maintained by the core team with the help of our fantastic contributors.
Code licensed MIT, website © 2020 Mitmproxy Project.

即可下载到对应系统的二进制可执行文件：

- Mac
    - https://snapshots.mitmproxy.org/6.0.2/mitmproxy-6.0.2-osx.tar.gz
- Win
    - https://snapshots.mitmproxy.org/6.0.2/mitmproxy-6.0.2-windows-installer.exe
    - https://snapshots.mitmproxy.org/6.0.2/mitmproxy-6.0.2-windows.zip
- Linux
    - https://snapshots.mitmproxy.org/6.0.2/mitmproxy-6.0.2-linux.tar.gz

安装mitmproxy

下载到二进制文件后，（Mac、Win、Linux等）各个系统中，即可正常安装。

Mac

Mac中也可以直接用 `brew` 去安装：

```
brew install mitmproxy
```

也可以用Python中的pip去（给Python环境中）安装：

```
pip install mitmproxy
```

注：如果后续 `mitmdump` 用到 `-s` 去加载的 `.py` 的python脚本中，用到了 `pyaml` 的话，则记得要先用 `pip` 安装 `pyyaml` ：

```
pip instal pyyaml
```

Win

安装遇到的问题

之前在win10中安装mitmproxy，遇到过2个问题：

build _openssl.c error C2065 X509_V_FLAG_CB_ISSUER_CHECK undeclared identifier

```
creating build\temp.win-amd64-3.8\Release\build\temp.win-ar
  C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\BI
  _openssl.c
  build\temp.win-amd64-3.8\Release\_openssl.c(1369): warnir
  build\temp.win-amd64-3.8\Release\_openssl.c(11095): erroi
  build\temp.win-amd64-3.8\Release\_openssl.c(11096): erroi
  build\temp.win-amd64-3.8\Release\_openssl.c(12429): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(12429): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(12452): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(13565): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(13575): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(13589): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(13599): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(16441): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(16465): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(16465): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(16475): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(16575): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(16599): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(19290): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(19290): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(19300): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(19350): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(19350): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(19360): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(19374): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(19374): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(19384): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(22275): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(23380): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(23404): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(25957): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(26094): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(32153): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(34129): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(34152): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(34609): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(34709): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(34793): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(38288): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(46480): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(46520): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(46713): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(46773): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(49146): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(49146): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(49156): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(49170): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(49170): warni
  build\temp.win-amd64-3.8\Release\_openssl.c(49180): warni
```

```
    build\temp.win-amd64-3.8\Release\_openssl.c(49194): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49194): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49204): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49218): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49218): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49228): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49242): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49242): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49252): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49266): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49266): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49276): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49290): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49290): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49300): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49314): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49314): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49324): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49338): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49338): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(49348): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(50421): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(50421): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(50444): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(50457): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(50457): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(50480): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(50659): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(50712): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(53826): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(53826): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(53849): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(54363): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(54620): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(57001): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(57001): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(57024): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(57037): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(57037): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(57060): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(57500): warn
    build\temp.win-amd64-3.8\Release\_openssl.c(57553): warn
    error: command 'C:\\Program Files (x86)\\Microsoft Visua
    ----------------------------------------
    ERROR: Failed building wheel for cryptography
    Running setup.py clean for cryptography
  Failed to build cryptography
  ERROR: Could not build wheels for cryptography which use PE
```

经过一番折腾，但最后是没解决。

具体过程详见：

- 【未解决】windows中pip安装mitmproxy报错： build _openssl.c error C2065 X509_V_FLAG_CB_ISSUER_CHECK undeclared identifier

ERROR: Could not build wheels for cryptography which use PEP 517 and cannot be installed directly

```
> pip install mitmproxy
......
Collecting pycparser
  Using cached https://files.pythonhosted.org/packages/68/9
Building wheels for collected packages: cryptography
  Building wheel for cryptography (PEP 517) ... error
  ERROR: Command errored out with exit status 1:
   command: 'c:\program files\python38\python.exe' 'c:\prog
       cwd: C:\Users\xxx\AppData\Local\Temp\pip-install-x4h
  Complete output (112 lines):
  running bdist_wheel
  running build
  running build_py
  creating build
  creating build\lib.win-amd64-3.8
  creating build\lib.win-amd64-3.8\cryptography
  copying src\cryptography\exceptions.py -> build\lib.win-a
  copying src\cryptography\fernet.py -> build\lib.win-amd64
  copying src\cryptography\utils.py -> build\lib.win-amd64-
  copying src\cryptography\__about__.py -> build\lib.win-am
  copying src\cryptography\__init__.py -> build\lib.win-amd
  creating build\lib.win-amd64-3.8\cryptography\hazmat
  copying src\cryptography\hazmat\_oid.py -> build\lib.win-
  copying src\cryptography\hazmat\__init__.py -> build\lib.
  creating build\lib.win-amd64-3.8\cryptography\x509
  copying src\cryptography\x509\base.py -> build\lib.win-am
  copying src\cryptography\x509\certificate_transparency.py
  copying src\cryptography\x509\extensions.py -> build\lib.
  copying src\cryptography\x509\general_name.py -> build\li
  copying src\cryptography\x509\name.py -> build\lib.win-am
  copying src\cryptography\x509\ocsp.py -> build\lib.win-am
  copying src\cryptography\x509\oid.py -> build\lib.win-amd
  copying src\cryptography\x509\__init__.py -> build\lib.wi
  creating build\lib.win-amd64-3.8\cryptography\hazmat\back
  copying src\cryptography\hazmat\backends\interfaces.py ->
  copying src\cryptography\hazmat\backends\__init__.py -> b
  creating build\lib.win-amd64-3.8\cryptography\hazmat\bind
  copying src\cryptography\hazmat\bindings\__init__.py -> b
  creating build\lib.win-amd64-3.8\cryptography\hazmat\prim
  copying src\cryptography\hazmat\primitives\cmac.py -> bui
  copying src\cryptography\hazmat\primitives\constant_time.
  copying src\cryptography\hazmat\primitives\hashes.py -> b
  copying src\cryptography\hazmat\primitives\hmac.py -> bui
  copying src\cryptography\hazmat\primitives\keywrap.py ->
  copying src\cryptography\hazmat\primitives\mac.py -> bui
  copying src\cryptography\hazmat\primitives\padding.py ->
  copying src\cryptography\hazmat\primitives\serialization.
  copying src\cryptography\hazmat\primitives\__init__.py ->
  creating build\lib.win-amd64-3.8\cryptography\hazmat\back
  copying src\cryptography\hazmat\backends\openssl\aead.py
```

```
    copying src\cryptography\hazmat\backends\openssl\backend.
    copying src\cryptography\hazmat\backends\openssl\ciphers.
    copying src\cryptography\hazmat\backends\openssl\cmac.py
    copying src\cryptography\hazmat\backends\openssl\decode_a
    copying src\cryptography\hazmat\backends\openssl\dh.py ->
    copying src\cryptography\hazmat\backends\openssl\dsa.py -
    copying src\cryptography\hazmat\backends\openssl\ec.py ->
    copying src\cryptography\hazmat\backends\openssl\encode_a
    copying src\cryptography\hazmat\backends\openssl\hashes.p
    copying src\cryptography\hazmat\backends\openssl\hmac.py
    copying src\cryptography\hazmat\backends\openssl\ocsp.py
    copying src\cryptography\hazmat\backends\openssl\rsa.py -
    copying src\cryptography\hazmat\backends\openssl\utils.py
    copying src\cryptography\hazmat\backends\openssl\x25519.p
    copying src\cryptography\hazmat\backends\openssl\x509.py
    copying src\cryptography\hazmat\backends\openssl\__init__
    creating build\lib.win-amd64-3.8\cryptography\hazmat\bind
    copying src\cryptography\hazmat\bindings\openssl\binding.
    copying src\cryptography\hazmat\bindings\openssl\_conditi
    copying src\cryptography\hazmat\bindings\openssl\__init__
    creating build\lib.win-amd64-3.8\cryptography\hazmat\prim
    copying src\cryptography\hazmat\primitives\asymmetric\dh.
    copying src\cryptography\hazmat\primitives\asymmetric\dsa
    copying src\cryptography\hazmat\primitives\asymmetric\ec.
    copying src\cryptography\hazmat\primitives\asymmetric\pad
    copying src\cryptography\hazmat\primitives\asymmetric\rsa
    copying src\cryptography\hazmat\primitives\asymmetric\uti
    copying src\cryptography\hazmat\primitives\asymmetric\x25
    copying src\cryptography\hazmat\primitives\asymmetric\__i
    creating build\lib.win-amd64-3.8\cryptography\hazmat\prim
    copying src\cryptography\hazmat\primitives\ciphers\aead.p
    copying src\cryptography\hazmat\primitives\ciphers\algori
    copying src\cryptography\hazmat\primitives\ciphers\base.p
    copying src\cryptography\hazmat\primitives\ciphers\modes.
    copying src\cryptography\hazmat\primitives\ciphers\__init
    creating build\lib.win-amd64-3.8\cryptography\hazmat\prim
    copying src\cryptography\hazmat\primitives\kdf\concatkdf.
    copying src\cryptography\hazmat\primitives\kdf\hkdf.py ->
    copying src\cryptography\hazmat\primitives\kdf\kbkdf.py -
    copying src\cryptography\hazmat\primitives\kdf\pbkdf2.py
    copying src\cryptography\hazmat\primitives\kdf\scrypt.py
    copying src\cryptography\hazmat\primitives\kdf\x963kdf.py
    copying src\cryptography\hazmat\primitives\kdf\__init__.p
    creating build\lib.win-amd64-3.8\cryptography\hazmat\prim
    copying src\cryptography\hazmat\primitives\twofactor\hotp
    copying src\cryptography\hazmat\primitives\twofactor\totp
    copying src\cryptography\hazmat\primitives\twofactor\util
    copying src\cryptography\hazmat\primitives\twofactor\__in
    running egg_info
    writing src\cryptography.egg-info\PKG-INFO
    writing dependency_links to src\cryptography.egg-info\dep
    writing requirements to src\cryptography.egg-info\require
    writing top-level names to src\cryptography.egg-info\top_
    reading manifest file 'src\cryptography.egg-info\SOURCES.
    reading manifest template 'MANIFEST.in'
    no previously-included directories found matching 'docs\_
    warning: no previously-included files matching '*' found
    writing manifest file 'src\cryptography.egg-info\SOURCES.
    running build_ext
    generating cffi module 'build\\temp.win-amd64-3.8\\Releas
    creating build\temp.win-amd64-3.8
    creating build\temp.win-amd64-3.8\Release
    generating cffi module 'build\\temp.win-amd64-3.8\\Releas
    generating cffi module 'build\\temp.win-amd64-3.8\\Releas
    building '_openssl' extension
    creating build\temp.win-amd64-3.8\Release\build
    creating build\temp.win-amd64-3.8\Release\build\temp.win-
    creating build\temp.win-amd64-3.8\Release\build\temp.win-
    C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\BI
    _openssl.c
    build\temp.win-amd64-3.8\Release\_openssl.c(498): fatal e
    error: command 'C:\\Program Files (x86)\\Microsoft Visua
    ----------------------------------------
    ERROR: Failed building wheel for cryptography
    Running setup.py clean for cryptography
  Failed to build cryptography
  ERROR: Could not build wheels for cryptography which use PE
```

也试了：

```
pip install cryptography
```

但问题依旧。

以及：

```
python -m pip install --no-use-pep517 mitmproxy
```

但报其他错误：

```
      copying src\cryptography\hazmat\primitives\twofactor\_
      running egg_info
      writing src\cryptography.egg-info\PKG-INFO
      writing dependency_links to src\cryptography.egg-info\d
      writing requirements to src\cryptography.egg-info\requi
      writing top-level names to src\cryptography.egg-info\to
      reading manifest file 'src\cryptography.egg-info\SOURCE
      reading manifest template 'MANIFEST.in'
      no previously-included directories found matching 'docs
      warning: no previously-included files matching '*' foun
      writing manifest file 'src\cryptography.egg-info\SOURCE
      running build_ext
      generating cffi module 'build\\temp.win-amd64-3.8\\Rele
      creating build\temp.win-amd64-3.8
      creating build\temp.win-amd64-3.8\Release
      generating cffi module 'build\\temp.win-amd64-3.8\\Rele
      generating cffi module 'build\\temp.win-amd64-3.8\\Rele
      building '_openssl' extension
      creating build\temp.win-amd64-3.8\Release\build
      creating build\temp.win-amd64-3.8\Release\build\temp.wi
      creating build\temp.win-amd64-3.8\Release\build\temp.wi
      C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\
      _openssl.c
      build\temp.win-amd64-3.8\Release\_openssl.c(498): fatal
      error: command 'C:\\Program Files (x86)\\Microsoft Visu
      ----------------------------------------
    Rolling back uninstall of cryptography
    Moving to c:\program files\python38\lib\site-packages\cry
     from C:\Program Files\Python38\Lib\site-packages\~ryptog
    Moving to c:\program files\python38\lib\site-packages\cry
     from C:\Program Files\Python38\Lib\site-packages\~ryptog
  ERROR: Command errored out with exit status 1: 'C:\Program
```

以及其他折腾。

但最后是没解决。

具体过程详见：

- 【未解决】windows中pip install mitmproxy失败：ERROR Could not build wheels for cryptography which use PEP 517 and cannot be installed directly

如何使用

在（Mac、Win等）PC端安装了mitmproxy后，自带mitmdump。

此处介绍如何去使用mitmdump。

通用逻辑

核心的通用逻辑是：

- 电脑端
  - 启动 `mitmdump` 代理
- 移动端
  - （初始化，只需第一次）安装 `mitmproxy` 的**根证书**
  - 给WiFi设置（PC端的mitmdump的）代理

即可正常使用代理，实现给移动端抓包等功能。

电脑端启动mitmdump代理

比如，PC端运行对应命令：

```
mitmdump -k -p 8081 -s middleware/Save1.py
```

即可。

接下来分不同平台详细介绍具体细节。

Mac

接着介绍，如何在 `Mac` 中使用 `mitmdump`

举例：

`Mac` 中终端去运行：

```
mitmdump -k -p 8081 -s middleware/Save1.py
```

启动mitmdump的代理

然后给手机端加上此处Mac的mitmdump的代理

即可实现：脚本 `Save1.py` 把手机端发出的所有的 `url` = `请求` = `链接地址` （还可以根据自己需要做一定过滤处理后再）保存起来（比如保存到一个文件中），供后续使用。

说明

此处的Save1.py是个python脚本

具体内容：

```python
# _*_ coding: utf-8 _*_

import json
import re
import os
import sys
# print("sys.executable=%s" % sys.executable)

try:
    import yaml
except Exception as err:
    print("Failed to import yaml: %s" % err)

class Saver:

    def __init__(self):
        self.Allurls = set()
        self.DataFilePath = self.get_DataFilePath()
        self.REMOVED = self.get_NeedSkip()

    def get_DataFilePath(self):
        # SavePath = "./middleware/Save.json"
        SavePath = os.path.join("middleware", "Save.json")
        with open(SavePath,"r",encoding="utf-8") as f:
            text = f.read()
            data = json.loads(text)
            return data["1"]

    def get_NeedSkip(self):
        # filepath = "./middleware/config.yml"
        filepath = os.path.join("middleware", "config.yml"
        try:
            with open(filepath, "r", encoding="utf-8") as
                text = f.read()
        except Exception:
            with open(filepath, "r") as f:
                text = f.read()
        config = yaml.load(text)
        REMOVED = [item.replace('.','\.') for item in conf
        return "|".join(REMOVED)

    def get_ContentType(self, headers):
        ContentType = "None"
        patten = "b'Accept', b'(.*?)'"
        result = re.search(patten, headers)
        if result:
            ContentType = result.group(1)
            ContentType = ContentType.split(",")[0]
        return ContentType if not "*" in ContentType else
    def request(self, flow):
        url = flow.request.url
        ContentType = self.get_ContentType(str(flow.request
        if not url in self.Allurls and not re.search(self.F
            self.Allurls.add(url)
            print(url)
            with open(self.DataFilePath, "a", encoding="ut
                f.write(url + "|" + ContentType)
                f.write('\n')

addons = [Saver()]
```

若想要后台运行，则后面加 `&`

```
mitmdump -k -p 8081 -s middleware/Save1.py &
```

移动端安装mitmproxy根证书

通用逻辑

即给移动端手机中安装 `mitmproxy` 的

`SSL代理证书` = `ssl证书` = `根证书` = `root CA`

核心逻辑：

- 手机中浏览器中打开 http://mitm.it
- 然后根据提示去下载（ `pem` 或 `crt` ）证书（文件）
- 点击安装证书文件

iOS

此处整理iOS的iPhone手机中，安装mitmproxy的根证书的详细情况：

- iOS
  - iPhone
    - 详见
      - 【已解决】给iPhone添加mitmproxy的mitmdump代理用于保存抓包链接到文件
      - 【已解决】iPhone8P中安装mitmproxy的CA的ssl证书

Android

此处整理安卓手机中，安装mitmproxy的根证书，对于不同手机的详细情况：

- Android
  - 华为
    - 荣耀
      - 【记录】给安卓手机中安装mitmproxy代理的SSL证书
      - 【记录】给自动抓包工具的安卓手机设置mitmproxy代理用于能抓包到链接地址
  - 小米
    - 小米9
      - 相关
        - 【已解决】安卓手机小米9中安装mitmproxy的SSL代理证书
          - 【无需解决】小米9中WLAN或WAPI证书中找不到mitmproxy的SSL的pem证书文件
          - 【无法解决】小米9中用ES文件管理器安装pem证书
    - 红米Note8Pro
      - 问题
        - 用微信或小米浏览器无法下载pem证书文件
      - 解决办法：
        - 换QQ浏览器就可以正常下载pem证书文件mitmproxy-ca-cert.pem
          - 细节
            - 不能用：
              - 微信
                - 点击Android无反应
              - 小米浏览器
                - 点击Android，弹框下载：perm.crt
                  - 而不是希望的：mitmproxy-ca-cert.pem
                  - 关键是：始终无法下载成功
            - 只能用：QQ浏览器
              - 点击Android，可以弹框下载：mitmproxy-ca-cert.pem
                - 是我们希望的pem证书
                - 也可以正常（瞬间）下载完毕
      - 详见

            - 【无法解决】红米Note8Pro中用微信或小米浏览器下载mitmproxy的SSL代理证书
            - 【已解决】红米Note8Pro中用QQ浏览器下载mitmproxy的Android的SSL代理证书
        - 相关
            - 【已解决】红米Note8Pro中安装mitmproxy的SSL代理证书
    - 红米10X
        - 问题：下载证书失败
            - 自带小米浏览器
                - 可弹框下载pem.crt，但下载失败
            - QQ浏览器
                - 可弹框下载mitmproxy-ca-cert.pem，但下载失败
                - 偶然甚至会提示：
                    - if you can see this, traffic is not passing through mitmproxy
            - UC浏览器
                - 可弹框下载mitmproxy-ca-cert.pem，但下载失败
                - 偶然甚至会提示：
                    - if you can see this, traffic is not passing through mitmproxy
        - 解决办法：
            - 试了多次，最后终于：
                - UC浏览器
                    - 可弹框并成功下载mitmproxy-ca-cert.pem
        - 详见：
            - 【已解决】红米10X安卓手机中无法下载mitmproxy的证书文件
- Vivo
    - iQOO U1x
        - 用QQ浏览器无法下载pem文件，提示下载失败
            - 解决办法：换Vivo的内置浏览器，即可下载mitmproxy-ca-cert.pem
        - 直接点击pem证书文件，无法安装：未找到证书文件
            - 问题现象
                - QQ浏览器下载到mitmproxy-ca-cert.pem，直接点击提示：找不到对应程序打开该文件
                - 更多安全设置-》从手机存储和SD卡安装，点击提示：未找到证书文件
                - 从文件管理中点击已下载的mitmproxy-ca-cert.pem，选 证书安装程序，也提示：未找到证书文件

        - 原因：Vivo不支持pem证书文件，只支持crt证书文件
        - 解决办法：把文件pem后缀改为crt
            - 点击即可正常安装
    - 详见：
        - 【已解决】给安卓手机ViVo的iQOO U1x下载和安装mitmproxy的SSL代理证书
        - 【已解决】安卓手机Vivo的iQOO U1x中手动安装mitmproxy-ca-cert.pem证书文件
        - 【已解决】安卓手机Vivo的iQOO U1x中点击安装mitmproxy的pem证书报错：未找到证书文件
        - 【未解决】给安卓手机Vivo的iQOO U1x初始化mitmdump的代理环境

移动端给WiFi设置代理

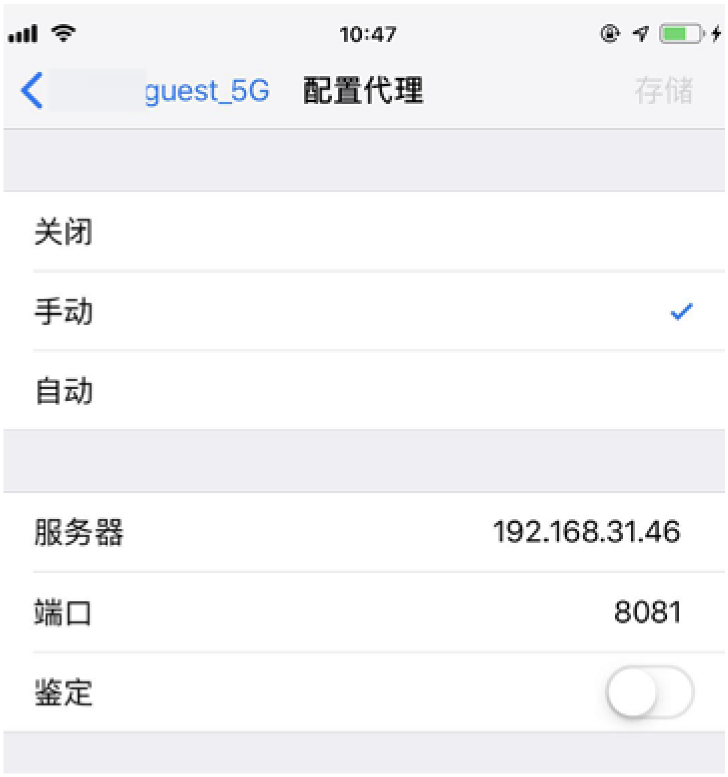
之后再去给移动端的WiFi设置（PC端的mitmdump的）代理（信息）。

- 细节详见
    - 如何添加代理 移动端 · 网络中转站：代理技术

此处简单举例如下：

iOS

iPhone



关闭
手动 ✓
自动

服务器 192.168.31.46
端口 8081
鉴定

常见问题

No module named yaml

- **现象**

Mac中用brew安装了mitmproxy，然后去运行：

```
mitmdump -p 8081 -s middleware/Save1.py
```

但是报错：

```
No module named yaml
```

- **原因**

Mac中通过brew安装的mitmproxy，会调用自己内部安装的python（此处是3.7.5）

而不是Mac中自己Python（2.7或3.8），mac中的python中都安装过yaml了

而mitmproxy中python，没有安装过yaml，所以上述脚本会报错。

- **解决办法**

不要用brew安装，而是用系统中的python的pip去安装mitmproxy

```
pip install mitmproxy
```

注：系统中的python是,此处是用的3.8，用pyenv设置全局为3.8

另外此处2.7的python中，pip安装mitmproxy会失败。

之后即可正常调用

```
mitmdump -p 8081 -s middleware/Save1.py
```

其中python解析器用的是此处系统的python了，因此可以正常找到（系统中python中已安装的）yaml，而不会报错了。

具体细节详见：

- 【基本解决】Mac中mitmdump运行命令报错：in script py No module named yaml

if you can see this, traffic is not passing through mitmproxy

- **现象**

手机中浏览器打开 http://mitm.it 后，看到页面提示：

`if you can see this, traffic is not passing through mitmproxy`

- **原因**

需要你手机中WiFi加上PC端的mitmproxy（mitmdump）的代理后，打开 http://mitm.it 后才能正常显示页面

- **解决办法**
  - PC端（Mac）中启动mitmproxy的代理
    - 举例
      - `mitmdump -k -p 8081 -s middleware/Save1.py`
  - 然后再给手机端的当前WiFi中加上对应的mitmdump的代理

细节详见：

【已解决】红米Note8Pro中去下载mitmproxy证书提示：if you can see this, traffic is not passing through mitmproxy

其他

此处整理一些和 `mitmdump` 相关的其他内容。

代码调用

- **背景需求**

Mac中想要用Python代码去控制 `mitmdump` ，即可以启动和停止 `mitmdump`

问题就转化为，Mac中如何写Python代码，能够检测到mitmdump的进程状态，如何解析具体信息，如何杀死对应，mitmdump进程等过程。

- **最后代码**

```
def stopExistingMitmproxy(curDevId):
    logging.info("curDevId=%s", curDevId)
    curDevIdInt = int(curDevId)
    isCheckCmdRunOk, mitmdumpInfoList = checkMitmdumpStatus
    logging.info("isCheckCmdRunOk=%s, mitmdumpInfoList=%s",
    isRunning = bool(mitmdumpInfoList)
    logging.info("isRunning=%s", isRunning)

    if isCheckCmdRunOk and isRunning:
        foundExistedDevId = False
        existedPid = None

        for eachMitmdumpInfo in mitmdumpInfoList:
            eachDevId = eachMitmdumpInfo["devId"]
            if eachDevId == curDevIdInt:
                foundExistedDevId = True
                existedPid = eachMitmdumpInfo["pid"]
                break

        if foundExistedDevId:
            killOK, errCode = utils.killProcess(existedPid)
            logging.info("killOK=%s, errCode=%s", killOK, e

def debugStartProxy():
    stopExistingMitmproxy(gCurDevId)

    taskFileFullPath = "/Users/limao/dev/xxx/crawler/appAut
    taskId = "5e9552d1c5c2eb3ccdf777bc"
    startTaskProxy(taskId, gCurDevId, taskFileFullPath)

    time.sleep(2)

    isCheckCmdRunOk, mitmdumpInfoList = checkMitmdumpStatus
    logging.info("isCheckCmdRunOk=%s, mitmdumpInfoList=%s",

def checkMitmdumpStatus():
    # check mitmdump is indeed running
    isCheckCmdRunOk, mitmdumpInfoList = False, []
    checkMitmdumpCmd = "ps aux | grep mitmdump"
    isCheckCmdRunOk, cmdResult = utils.getCommandOutput(che
    logging.info("isCheckCmdRunOk=%s, cmdResult=%s", isChec
    if isCheckCmdRunOk:
        # resultList = cmdResult.split("\n")
        resultList = cmdResult.split(os.linesep)
        logging.info("resultList=%s", resultList)
        # limao            56562   0.0  0.0  4267948    664
        # limao            56560   0.0  0.0  4268636   1112
        # limao            55396   0.0  0.1  4381268  11568
        if resultList:
            for eachLine in resultList:
                logging.info("eachLine=%s", eachLine)
                mitmdumpP = "^\s*(?P<username>\w+)\s+(?P<p
                foundMitmdump = re.search(mitmdumpP, eachL
                logging.info("foundMitmdump=%s", foundMitm
                if foundMitmdump:
                    username = foundMitmdump.group("usernar
                    pid = foundMitmdump.group("pid")
                    port = foundMitmdump.group("port")
                    devId = foundMitmdump.group("devId")
                    scriptFile = foundMitmdump.group("scrip
                    logging.info("username=%s, pid=%s, port
                    curMitmdumpDict = {
                        "username": username,
                        "pid": int(pid),
                        "port": int(port),
                        "scriptFile": scriptFile,
                        "devId": int(devId),
                    }
                    mitmdumpInfoList.append(curMitmdumpDict
    logging.info("mitmdumpInfoList=%s", mitmdumpInfoList)
    return isCheckCmdRunOk, mitmdumpInfoList

def killProcess(pid):
    """Kill process by pid

    Args:
        pid (id): process ID
    Returns:
    Raises:
    """
    isKillOk, errCode = False, 0
    pidInt = int(pid)
    killCmd = "kill -9 %s" % pidInt
    returnCode = os.system(killCmd)
    logging.debug("Command: %s -> returnCode=%s", killCmd,
    RETURN_CODE_OK = 0
    if returnCode == RETURN_CODE_OK:
        isKillOk = True
    else:
        errCode = returnCode
    return isKillOk, errCode
```

基本完成了想要的功能：

- 在启动任务前，启动mitmproxy
- 如果之前已有当前设备id的mitmdump在运行，就kill掉
    - 因为很可能是之前的旧的task的对应的代理，所以要关闭掉，再重新启动，才能传递当前task的data文件
- 然后再去启动mitmproxy，之后再去检测看看是否的确已启动

后续优化版本

```
MitmdumpPortBase = 8080
curDevId = 1
RunProxyShellFilename = "runProxy.sh"


def generateShellFile(fileContentStr, shellFilename, taskId
    """Generate shell file, which is used to run command
        such as
            mitmdump proxy
            crawlerStart.py
            USB port forward
            wda server(xcodebuild/XCode)
    """
    logging.debug("fileContentStr=%s, shellFilename=%s, tas
    if taskId:
        shellFolder = getTaskShellFolder(taskId)
        # /Users/limao/dev/xxx/crawler/appAutoCrawler/AppCr
    else:
        shellFolder = OutputRootFolder
    logging.debug("shellFolder=%s", shellFolder)
    shellFullPath = os.path.join(shellFolder, shellFilename
    logging.debug("shellFullPath=%s", shellFullPath)
    # /Users/limao/dev/xxx/crawler/appAutoCrawler/AppCrawle
    shellAbsFullPath = os.path.abspath(shellFullPath)
    logging.debug("shellAbsFullPath=%s", shellAbsFullPath)
    respShellFullPath = shellAbsFullPath
    utils.saveTextToFile(respShellFullPath, fileContentStr)
    utils.chmodAddX(shellFullPath, isOnlySelf=False)
    # utils.chmodAddX(respShellFullPath)
    logging.debug("respShellFullPath=%s", respShellFullPath
    # /Users/limao/dev/xxx/crawler/appAutoCrawler/AppCrawle
    return respShellFullPath

#---------- generate start service command ----------

def generateMitmproxyStartCommand(curDevId):
    curMitmdumpPort = MitmdumpPortBase + int(curDevId)
    # mitmproxyStartCommand = "mitmdump -p %d -s middleware
    mitmproxyStartCommand = "mitmdump -k -p %d -s middlewar
    logging.debug("mitmproxyStartCommand=%s", mitmproxyStar
    # mitmdump -k -p 8081 -s middleware/Save1.py
    mitmproxyCommandList = [
        # "cd /Users/limao/dev/xxx/crawler/appAutoCrawler/A
        "cd %s" % AppCralwerFolder,
        "pwd",
        mitmproxyStartCommand,
    ]
    logging.debug("mitmproxyCommandList=%s", mitmproxyComma
    # ['cd /Users/limao/dev/xxx/crawler/appAutoCrawler/App(
    # mitmproxyCommandStr = ";".join(mitmproxyCommandList)
    # mitmproxyCommandStr = "; ".join(mitmproxyCommandList
    mitmproxyCommandStr = "\n".join(mitmproxyCommandList)
    # cd /Users/limao/dev/xxx/crawler/appAutoCrawler/AppCr
    # pwd
    # mitmdump -k -p 8081 -s middleware/Save1.py
    logging.debug("mitmproxyCommandStr=%s", mitmproxyComma
    return mitmproxyCommandStr

#---------- generate shell file ----------

def generateRunProxyShell(devId, taskId):
    mitmproxyCmdStr = generateMitmproxyStartCommand(devId)
    logging.debug("mitmproxyCmdStr=%s", mitmproxyCmdStr)
    return generateShellFile(mitmproxyCmdStr, RunProxyShel

#---------- detect service status ----------

def detectMitmdumpStatus():
    # crifanli 9428 0.0 0.6 4341956 19792 s006 S+ 9:16上午
    # crifanli 10982 0.0 0.0 4268032 776 s005 S+ 1:51下午 0
    # crifanli 10980 0.0 0.0 4278852 1116 s005 S+ 1:51下午
    # mitmdumpP = "^\s*(?P<username>\w+)\s+(?P<pid>\d+)\s+
    mitmdumpP = "^\s*(?P<username>\w+)\s+(?P<pid>\d+)\s+.+
    return utils.grepProcessStatus("mitmdump", mitmdumpP)

def startTaskProxy(devId, taskId):
    logging.info("Start proxy for: devId=%s, taskId=%s", d
    proxyShellFile = generateRunProxyShell(devId, taskId)
    logging.debug("proxyShellFile=%s", proxyShellFile)
    utils.launchTerminalRunShellCommand(proxyShellFile)

#---------- makesure service running ----------

def makesureProxyingRunning(devId, taskId):
    def checkProxyStatus():
        isCheckOk, isRunning, infoList = detectMitmdumpSta
        return isCheckOk and isRunning

    def startCurTaskProxy():
        startTaskProxy(devId, taskId)

    makesureServiceRunning(checkProxyStatus, startCurTaskP
```

相关的:

`other/common/libs/utils.py`

```
import re


#----------------------------------------------------------------
# Process
#----------------------------------------------------------------

def runCommand(consoleCommand):
    """run command using subprocess call"""
    isRunCmdOk = False
    errMsg = "Unknown Error"

    try:
        resultCode = subprocess.check_call(consoleCommand,
        if resultCode == 0:
            isRunCmdOk = True
            errMsg = ""
        else:
            isRunCmdOk = False
            errMsg = "%s return code %s" % (consoleCommand,
    except subprocess.CalledProcessError as callProcessErr
        isRunCmdOk = False
        errMsg = str(callProcessErr)
        # "Command 'ffmpeg -y -i /Users/crifan/.../debug/e

    return isRunCmdOk, errMsg


def getCommandOutput(consoleCommand, consoleOutputEncoding
    """
        get command output from terminal
    """
    # print("getCommandOutput: consoleCommand=%s" % consol
    isRunCmdOk = False
    consoleOutput = ""
    try:
        # consoleOutputByte = subprocess.check_output(cons

        consoleOutputByte = subprocess.check_output(consol

        # commandPartList = consoleCommand.split(" ")
        # print("commandPartList=%s" % commandPartList)
        # consoleOutputByte = subprocess.check_output(comma
        # print("type(consoleOutputByte)=%s" % type(consol
        # print("consoleOutputByte=%s" % consoleOutputByte

        consoleOutput = consoleOutputByte.decode(consoleOu
        consoleOutput = consoleOutput.strip() # '640x360'
        isRunCmdOk = True
    except subprocess.CalledProcessError as callProcessErr
```

```
        cmdErrStr = str(callProcessErr)
        print("Error %s for run command %s" % (cmdErrStr, c

    # print("isRunCmdOk=%s, consoleOutput=%s" % (isRunCmdOk
    return isRunCmdOk, consoleOutput

def launchTerminalRunShellCommand(shellFile, isForceNewInst
    """in Mac, Launch terminal(Mac Terminal / iTerm2) and e

    Args:
        shellFile (str): shell file full path
        isUseiTerm2 (bool): True to use iTerm2, False to us
        isForceNewInstance (bool): whether pase -n to open,
    Returns:
    Raises:
    """
    logging.debug("shellFile=%s, isForceNewInstance=%s, isU

    TerminalApp_iTerm2 = '/Applications/iTerm.app'
    TerminalApp_Terminal = 'Terminal'
    if isUseiTerm2:
        terminalApp = TerminalApp_iTerm2
    else:
        terminalApp = TerminalApp_Terminal

    cmdList = [
        "/usr/bin/open",
    ]

    if isForceNewInstance:
        cmdList.append("-n")

    extarArgs = shellFile
    restCmdList = [
        "-a",
        terminalApp,
        '--args',
        extarArgs,
    ]
    cmdList.extend(restCmdList)
    logging.debug("cmdList=%s" % cmdList)

    curProcess = subprocess.Popen(cmdList, stdin=subprocess
    logging.debug("curProcess=%s" % curProcess)

    returnCode = None
    while True:
        returnCode = curProcess.poll()
        logging.debug("returnCode=%s", returnCode)
        if returnCode is not None:
            logging.debug("subprocess end: returnCode=%s",
            break
        time.sleep(0.5)

    logging.debug("Final returnCode=%s", returnCode)
    logging.debug("Complete launch %s and run shell %s", te

def killProcess(pid):
    """Kill process by pid

    Args:
        pid (id): process ID
    Returns:
    Raises:
    """
    isKillOk, errCode = False, 0
    pidInt = int(pid)
    killCmd = "kill -9 %s" % pidInt
    returnCode = os.system(killCmd)
    logging.debug("Command: %s -> returnCode=%s", killCmd,
    RETURN_CODE_OK = 0
    if returnCode == RETURN_CODE_OK:
        isKillOk = True
    else:
        errCode = returnCode
    return isKillOk, errCode

def grepProcessStatus(processFile, singleLinePattern, psCm
    """grep process info status from ps output

    Args:
        processFile (str): process file name
        singleLinePattern (str): single process line searc
        psCmd (str): ps command, default: ps aux
    Returns:
    Raises:
    Examples:
        input: "crawlerStart.py", "^\s*(?P<username>\w+)\s
        output: [{'username': 'limao', 'pid': '64320', 'ta
    """
    logging.debug("processFile=%s, singleLinePattern=%s", p
    isCheckCmdRunOk, isRunning, processInfoList = False, Fa

    groupNameList = re.findall("\(\?P<(\w+)>", singleLinePa
    logging.debug("groupNameList=%s", groupNameList)
    # groupNameList=['username', 'pid', 'port', 'scriptFil
    grepProcessCmd = "%s | grep %s" % (psCmd, processFile)
    logging.debug("grepProcessCmd=%s", grepProcessCmd)
    isCheckCmdRunOk, cmdResult = getCommandOutput(grepProce
    logging.debug("isCheckCmdRunOk=%s, cmdResult=%s", isChe
    if isCheckCmdRunOk:
        # lineSeparator = "\n"
```

```
            lineSeparator = os.linesep
            resultList = cmdResult.split(lineSeparator)
            logging.debug("resultList=%s", resultList)
            # limao             56562   0.0  0.0  4267948    664
            # limao             56560   0.0  0.0  4268636   1112
            # limao             55396   0.0  0.1  4381268  11568
            if resultList:
                for eachLine in resultList:
                    logging.debug("eachLine=%s", eachLine)
                    foundProcess = re.search(singleLinePattern,
                    logging.debug("foundProcess=%s", foundProce
                    if foundProcess:
                        curProcessInfoDict = {}
                        for eachKey in groupNameList:
                            curValue = foundProcess.group(eachK
                            curProcessInfoDict[eachKey] = curVa
                        logging.debug("curProcessInfoDict=%s",
                        processInfoList.append(curProcessInfoD

        isRunning = bool(processInfoList)
        logging.debug("isRunning=%s, processInfoList=%s", isRun
        return isCheckCmdRunOk, isRunning, processInfoList
```

注：

相关库文件的最新版，详见：

- https://github.com/crifan/crifanLibPython/blob/master/python3/crifanLib/crifanSystem.py
    - `grepProcessStatus`
    - `killProcess`
    - `launchTerminalRunShellCommand`
    - `getCommandOutput`
    - `runCommand`

打包exe

`windows` 中用 `PyInstaller` 打包 `python` 脚本为exe文件

其中python脚本调用到 `mitmdump`

可以理解为：打包mitmdump的Python为exe

核心命令：

```
pyinstaller pymitmdump\mitmdumpStartApi.py --distpath pymit

pyinstaller pymitmdump\mitmdumpOtherApi.py --distpath pymit
```

可以生成2个exe文件。

- 细节详见：【已解决】windows中用PyInstaller打包mitmdump的Python脚本为exe

附录

下面列出相关参考资料。

help语法

参考资料

- 【已解决】Mac中安装Mitmdump和启动服务
- 【基本解决】Mac中mitmdump运行命令报错：in script py No module named yaml
- 【未解决】windows中pip安装mitmproxy报错： build _openssl.c error C2065 X509_V_FLAG_CB_ISSUER_CHECK undeclared identifier
- 【未解决】windows中pip install mitmproxy失败：ERROR Could not build wheels for cryptography which use PEP 517 and cannot be installed directly
- 【已解决】iPhone8P中安装mitmproxy的CA的ssl证书
- 【已解决】给iPhone添加mitmproxy的mitmdump代理用于保存抓包链接到文件
- 【记录】给安卓手机中安装mitmproxy代理的SSL证书
- 【记录】给自动抓包工具的安卓手机设置mitmproxy代理用于能抓包到链接地址
- 【已解决】安卓手机小米9中安装mitmproxy的SSL代理证书
- 【无需解决】小米9中WLAN或WAPI证书中找不到mitmproxy的SSL的pem证书文件
- 【无法解决】小米9中用ES文件管理器安装pem证书
- 【无法解决】红米Note8Pro中用微信或小米浏览器下载mitmproxy的SSL代理证书
- 【已解决】红米Note8Pro中用QQ浏览器下载mitmproxy的Android的SSL代理证书
- 【已解决】红米10X安卓手机中无法下载mitmproxy的证书文件
- 【已解决】给安卓手机ViVo的iQOO U1x下载和安装mitmproxy的SSL代理证书
- 【已解决】安卓手机Vivo的iQOO U1x中手动安装mitmproxy-ca-cert.pem证书文件
- 【已解决】安卓手机Vivo的iQOO U1x中点击安装mitmproxy的pem证书报错：未找到证书文件
- 【未解决】给安卓手机Vivo的iQOO U1x初始化mitmdump的代理环境
- 【已解决】给VMWare中macOS中抓包项目开启mitmdump代理
- 【已解决】红米Note8Pro中去下载mitmproxy证书提示：if you can see this, traffic is not passing through mitmproxy
- 【已解决】windows中用PyInstaller打包mitmdump的Python脚本为exe
- 【已解决】自动抓包平台化：Python调用命令行启动mitmproxy代理
-
- Mitmproxy教程 - zha0gongz1 - 博客园
-