视频号下载器2.5版本完全开源免费

原创 kanadeblisst Python成长路 2024年07月23日 09:00 北京

前言

接触视频号下载源于看雪论坛的一篇文章(微信视频号视频加密逆向)，当时看了一下文章发现实现并不复杂，所以自己写了一个工具打算给公众号涨点粉。

不知不觉从发布的第一个版本到现在已经半年了，涨粉效果甚微，可能是视频号下载的受众面也不广，还有一堆优秀的软件可以下载。比如下面这个工具也是免费的：

```
https://github.com/qiye45/wechatVideoDownload
```

现在将这个视频号下载工具的源码公开，有兴趣的可以自己二次开发，我希望你二次开发后的工具也能免费使用。当然我只是希望，毕竟国内的环境有目共睹。之前开源的微信OCR在b站上被人包装一下(base64)就是自己的了。

为什么开源

之前抠抠搜搜的，现在为什么直接开源了？主要是现在我不想再维护了，这个并没有收益且浪费时间。很多人什么也不懂，加你微信问这问那，他们觉得理所当然，你写的软件当然要回答。

时间一长谁都会不耐烦，天天在回答一些重复且毫无技术含量的问题，比如有部分人问你使用码怎么获取。

不想维护但又想大家能使用，那就开源让有能力的人来维护。群里有些人就喜欢说，你开源啊，开源了让大家一起写代码不就好了。

发布过的文章

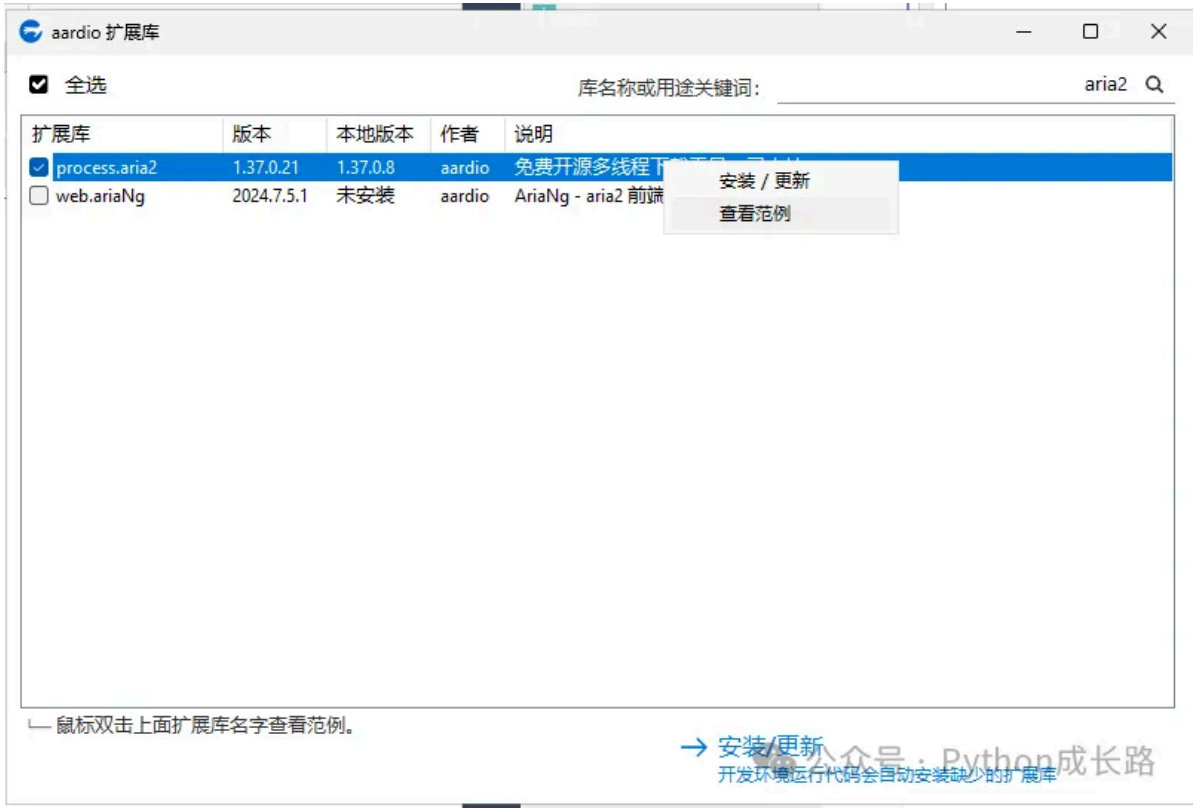
- **写个视频号下载工具**
- **视频号下载工具更新**
- **视频号下载工具重构版本2.0**
- **视频号下载器常见问题**
- **视频号下载器更新2.5版本**

实现原理

简单说一下软件的实现原理，方便后面想二次开发的人理解代码。软件分为两部分：下载器和监听器，用的是aardio的tab分开。

下载器

下载器没什么可说的，在aardio扩展库里搜aria2，然后右键查看范例就是下载器的代码了(基本照抄的)。



监听器

核心的内容都在监听器里，结构也不复杂，主要有两个内容：监听到链接和解密视频。

监听

监听使用的是sunny的dll，官网地址： `https://esunny.vip/` 。优点在于可以不设置代理监听，直接监听某个进程。

微信视频号的相关逻辑都在JavaScript里，可以通过代理修改JavaScript文件获取到想要的数据。修改JavaScript的核心代码如下：

```
if(string.find(url, "@virtual_svg-icons-register.publish")){
 var bodyLen = resp.GetResponseBodyLen();
 var body = resp.GetResponseBody();
 var regex = "async finderGetCommentDetail\((\w+)\)\{return(.*?)\}async";
 var replaceStr = "async finderGetCommentDetail(\1){const feedResult=await\2;var data_object=feedI
 var hookBody = string.replace(body, regex, replaceStr);
 resp.SetResponseData(hookBody, #hookBody);
}
```

replaceStr(替换后的JavaScript)格式化后：

```
async finderGetCommentDetail(\1) {
 const feedResult = await\2;
 var data_object = feedResult.data.object;
 var media = data_object.objectDesc.media[0];
 var fetch_body = {
  duration: media.spec[0].durationMs,
  title: data_object.objectDesc.description,
  url: media.url + media.urlToken,
  size: media.fileSize,
  key: media.decodeKey,
  id: data_object.id,
  nonce_id: data_object.objectNonceId,
  nickname: data_object.nickname,
  createtime: data_object.createtime,
  fileFormat: media.spec.map(o = >o.fileFormat)
 };
```

```
 fetch('https://www.httpbin.org/post', {
  method: 'POST',
  headers: {
   'Content-Type': 'application/json',
  },
  body: JSON.stringify(fetch_body)
 }).then(response = >{
  console.log(response.ok, response.body)
 });
 return feedResult;
}
```

就是将视频的一些信息(标题、链接、时长等)发送给 `https://www.httpbin.org/post` ，这是个接口测试的网站，做爬虫的应该知道。这个链接随便改成什么，反正也不需要网站能接收到数据。

然后在sunny里面监听 `https://www.httpbin.org/post` 这个网站，拿到 `fetch_body` 的数据。

```
if(string.find(url, "@www.httpbin.org")){
 var body = req.GetRequestBody();
 // 处理fetch_body
}
```

这样就拿到了视频的相关数据，主要是链接、标题和解密需要用到的seed。

解密

解密逻辑比较麻烦，微信是调用的wasm来解密的，使用的是一个叫 `Isaac64` 的算法。我试了下搜索到的 `Isaac64` 得到的结果和wasm不一样，如果想用C++调用wasm来解密的话也挺麻烦。

所以想跟微信一样，在浏览器中加载wasm，然后利用rpc来调用JavaScript。这种方式就简单多了，核心逻辑就几行html代码，后面更新也方便，只需将链接的版本号更新一下(如果改版不是很多)：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title></title>
    </head>
    <script>
        window.VTS_WASM_URL = "https://aladin.wxqcloud.qq.com/aladin/ffmepeg/video-decode/1.2.50/
        window.MAX_HEAP_SIZE = 33554432;
    </script>
    <script src="https://aladin.wxqcloud.qq.com/aladin/ffmepeg/video-decode/1.2.50/wasm_video_dec
    <script>
        var decryptor_array;

  function Uint8ArrayToBase64(bytes) {
     let binary = '';
     const len = bytes.byteLength;
     for (let i = 0; i < len; i++) {
       binary += String.fromCharCode(bytes[i]);
     }
     return window.btoa(binary);
  }
```

```
  function base64ToUint8Array(base64) {
     const binaryString = window.atob(base64);
     const len = binaryString.length;
     const bytes = new Uint8Array(len);
     for (let i = 0; i < len; i++) {
       bytes[i] = binaryString.charCodeAt(i);
     }
     return bytes;
  }


        function wasm_isaac_generate(t, e) {
            decryptor_array = new Uint8Array(e);
            var r = new Uint8Array(Module.HEAPU8.buffer,t,e);
            decryptor_array.set(r.reverse());
        }

        function get_decryptor_array(seed) {
            let decryptor = new Module.WxIsaac64(seed);
            decryptor.generate(131072);
            return Uint8ArrayToBase64(decryptor_array);
        }

    </script>
    <body>
    </body>
</html>
```

然后调用 `get_decryptor_array(seed)` 就拿到了解密需要的数组，再跟原视频做异或解密。

aardio

aardio调用浏览器和执行浏览器的JavaScript：

```
import web.view;

winform.webView = web.view(winform);
if(!winform.webView){
 if(win.msgboxTest("当前电脑可能未安装Edge WebView2(未安装无法使用),是否打开下载地址？","错误提示", wi
  process.openUrl("https://go.microsoft.com/fwlink/?linkid=2124701");
 }
 win.quitMessage();
 return;
}
// 非开发环境禁用F12
if(!_STUDIO_INVOKED) winform.webView.enableDevTools(false);
winform.webView.html = /*上面的html*/
```

接着通过 `winform.webView.xcall("get_decryptor_array", seed)` 来获取解密数组。

存在的问题

下面的这些问题就等其他大佬来修复了，我还是专心研究技术去了，不在这上面浪费时间了。也或许能在下一个工具再见。

有部分视频解密后无法播放

相同标题的文件会被覆盖，这个只需要判断一下重命名即可

还有一些兼容性问题，少部分人打开会有奇奇怪怪的问题

代码

完整代码： https://github.com/kanadeblisst00/WechatVideoSniffer2.0