SIMPLY
SECURE

G Data
Whitepaper 30/10/2017

Analysis of

Xafecopy
&
Android.Application.SMSreg.A

Analysis by: https://twitter.com/RansomBleed

G DATA Software AG | Application Development | October 2017

Contents

1

Xafecopy **3**

1. Introduction **3**

2. Xafecopy **3**

3. Overview of SMSreg.A **3**

3.1. Commonsdk package 3

3.2. hyjt package 4

3.3. wapsdk package 4

4. JavaScript files **5**

5. Log files **6**

6. Network traffic **7**

7. Admin panel **7**

8. YARA rule **8**

9. File hashes and resources **9**

Xafecopy

1. Introduction

Due to some articles [1][2] found on the web reporting about the Xafecopy[3] malware – a battery optimizer app which is secretly subscribing to premium services via SMS - this android dropper caught my attention in several ways.

Firstly, since there are no detailed articles about how this kind of malware works, the goal is to create a better understanding of this unique malware.

Secondly, when I was searching for a sample on Virustotal, most of the samples were quite new which is another good sign as there could be new features which no other malware in the past had built in.

2. Xafecopy

Inside Xafecopy the *o.t.t.c.class* is the interesting class file to look at. The class is using *System.loadLibrary* to load the native .so library *libhello.so* inside the lib folder. The purpose of *libhello.so* is to drop SMSreg.A[4], install and to execute it.

3. Overview of SMSreg.A

3.1. Commonsdk package

SMSreg.A contains multiple packages and sub-packages. The first package is the *com.commonsdk.commonsdk* package as you can see below in figure 4. The most interesting packages inside here is the *encrypt* as well as the *util* package.

The *encrypt* package contains encryption functions for the encrypted JavaScript files within the assets folder. The *util* package contains utility classes, for example *AppUtil. AppUtil* contains functions like *isServiceRunning*, *openVideo* and *isAppInstalled*. The functionality of those features should be pretty self-explanatory when reading the name.

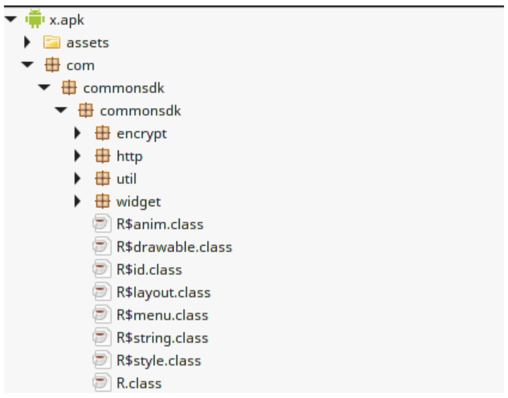
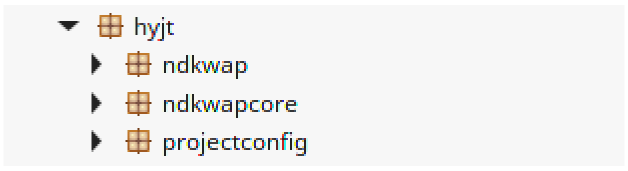


Figure 4. commonsdk package

3.2. hyjt package

Inside the package *hyit*, there is an interesting sub-package called *projectconfig*. The classes *CountryValues.class* and the *OperatorConstants.class* inside *projectconfig* are both suggesting that the malware is targeting Turkish users, since only the country code from Turkey and Turkish telephone providers were added. The *CaptchaUrl.class* file contains a list of a few IP addresses which are possibly being used to bypass captcha software used by the subscription services. The class *LocalUrlContants.class* initializes new Bean objects with parameters like the service subscription URL or the JavaScript file to execute on the loaded offer. We will have a more detailed look on the JavaScript files later on.



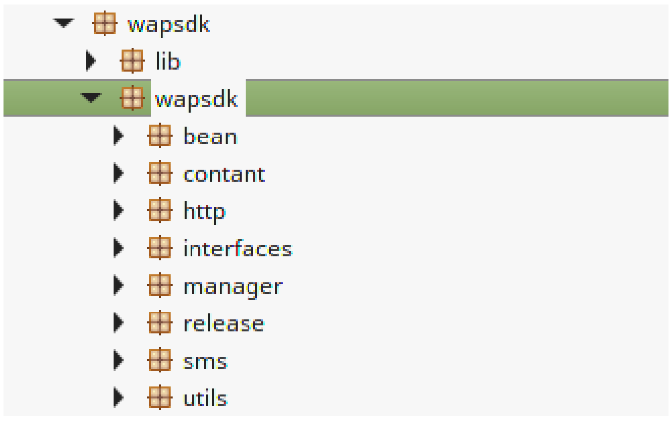
```
▼ hyjt
    ▶ ndkwap
    ▶ ndkwapcore
    ▶ projectconfig
```

Figure 5. hyjt package

3.3. wapsdk package

The *wapsdk* package is where all the magic happens. Most of the configuration settings like URLs are stored in packages like *contant* or *http*. The manager package contains class files, which are used to launch the subscription based URL's and also checks for incoming SMS for strings like "verification code". The *manager* package also contains a lot of logging functionalities which are mostly written in Chinese. This suggests that the malware was most likely written by a Chinese author.

The *sms* package contains hard-coded SMS messages which are seem to subscribe the user to some type of mobile gaming content since most of the messages contain the word "GAME". Inside the *util* package, there is an interesting class *UploadUtil*, which is used to upload bitmaps to a webserver. More details about this later. *AdBeanUtil.class* is used to parse URLs and for attaching various parameters used for tracking to it. Another interesting component are the digits 0-9 stored as Base64 in order to bypass the captchas. Multiple captcha classes are created in order to differentiate between image captchas with the length of 4, 5 and 10 digits.



```
▼ wapsdk
    ▶ lib
    ▼ wapsdk
        ▶ bean
        ▶ contant
        ▶ http
        ▶ interfaces
        ▶ manager
        ▶ release
        ▶ sms
        ▶ utils
```

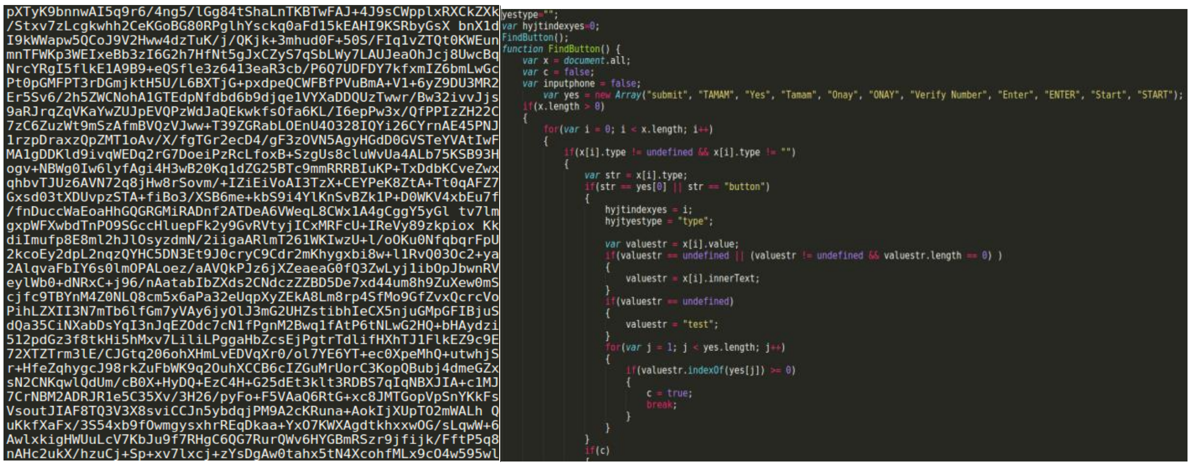
Figure 6. wapsdk package

In summary, the *hyjt* package is being used to store the configuration files for SMSreg.A. *Commonsdk* contains cryptography functions as well as basic utilities like networking or logging features. *Wapsdk* contains the core of the application. For instance, inside this package, the SMS functionality is stored. *Commonsdk* and *Wapsdk* seem to be libraries which are potentially used by other applications. When searching for code pieces within those packages, it turns out that some code pieces of the whole libraries can be found on websites like “www.codesnippet.cn” or “www.javatips.net” as the function names and variables are pretty similar. However, there is not a single page containing the whole library, which indicates that the malware developer must have been creating it by himself.

There is one last package *nineoldandroids* which is not listed here since it’s just used for compatibility with older android versions.

4. JavaScript files

So during my research on the Xafecopy malware I was initially only interested in the JavaScript files since they were encrypted as you can see in figure 7. This didn’t look legit at all. Luckily SMSreg.A contains hardcoded keys, which makes it really easy to decrypt the DES3 encrypted JavaScript files.



```
pXTyK9bnnwAI5q9r6/4ng5/lGg84tShaLnTKBTwFAJ+4J9sCWpplxRXCkZXk
/Stxv7zLcgkwhh2CeKGoBG80RPglhYsckq0aFd15kEAHI9KSRbyGsX bnX1d
I9kWWapw5QCoJ9V2Hww4dzTuK/j/QKjk+3mhud0F+50S/FIq1vZTQt0KWEun
mnTFWKp3WEIxeBb3zI6G2h7HfNt5gJxCZyS7qSbLWy7LAUJeaOhJcj8UwcBq
NrcYRgI5flkE1A9B9+eQSfle3z6413eaR3cb/P6Q7UDFDY7kfxmIZ6bmLwGc
Pt0pGMFPT3rDGmjktH5U/L6BXTjG+pxdpeQCWFBfPVuBmA+V1+6yZ9DU3MR2
Er5Sv6/2h5ZWCNohA1GTEdpNfdbd6b9djqe1VYXaDDQUzTwwr/Bw32ivvJjs
9aRJrqZqVKaYwZUJpEVQPzWdJaQEkwkfsOfa6KL/I6epPw3x/QfPPIzZH22C
7zC6ZuzWt9mSzAfmBVQzVJww+T39ZGRabLOEnU40328IQYi26CYrnAE45PNJ
1rzpDraxzQpZMT1oAv/X/fgTGr2ecD4/gF3zOVN5AgyHGdD0GVSTeYVAtIwF
MA1gDDKld9ivqWEDq2rG7DoeiPzRcLfoxB+SzgUs8cluWvUa4ALb75KSB93H
ogv+NBWg0Iw6lyfAgi4H3wB20Kq1dZG25BTc9mmRRRBIuKP+TxDdbKCveZwx
qhbvTJUz6AVN72q8jHw8rSovm/+IZiEiVoAI3TzX+CEYPeK8ZtA+Tt0qAFZ7
Gxsd03tXDUvpzSTA+fiBo3/XSB6me+kbS9i4YlKnSvBZk1P+D0WKV4xbEu7f
/fnDuccWaEoaHhGQGRGMiRADnf2ATDeA6VWeqL8CWx1A4gCggY5yGl tv7lm
gxpWFXwbdTnPO9SGccHluepFk2y9GvRVtyjICxMRFcU+IReVy89zkpiox Kk
diImufp8E8ml2hJlOsyzdmN/2iigaARlmT261WKIwzU+l/oOKu0NfqbqrFpU
2kcoEy2dpL2nqzQYHC5DN3Et9J0cryC9Cdr2mKhygxbi8w+l1RvQ03Oc2+ya
2AlqvaFbIY6s0lmOPALoez/aAVQkPJz6jXZeaeaG0fQ3ZwLyj1ibOpJbwnRV
eylWb0+dNRxC+j96/nAatabIbZXds2CNdczZZBD5De7xd44um8h9ZuXew0mS
cjfc9TBYnM4Z0NLQ8cm5x6aPa32eUqpXyZEkA8Lm8rp4SfMo9GfZvxQcrcVo
PihLZXII3N7mTb6lfGm7yVAy6jyOlJ3mG2UHZstibhIeCX5njuGMpGFIBjuS
dQa35CiNXabDsYqI3nJqEZOdc7cN1fPgnM2Bwq1fAtP6tNLwG2HQ+bHAydzi
512pdGz3f8tkHi5hMxv7LiliLPggaHbZcsEjPgtrTdlifHXhTJ1FlkEZ9c9E
72XTZTrm3lE/CJGtq206ohXHmLvEDVqXr0/ol7YE6YT+ec0XpeMhQ+utwhjS
r+HfeZqhygcJ98rkZuFbWK9q2OuhXCCB6cIZGuMrUorC3KopQBubj4dmeGZx
sN2CNKqwlQdUm/cB0X+HyDQ+EzC4H+G25dEt3klt3RDBS7qIqNBXJIA+c1MJ
7CrNBM2ADRJR1e5C35Xv/3H26/pyFo+F5VAaQ6RtG+xc8JMTGopVpSnYKkFs
VsoutJIAF8TQ3V3X8sviCCJn5ybdqjPM9A2cKRuna+AokIjXUpTO2mWALh Q
uKkfXaFx/3S54xb9fOwmgysxhrREqDkaa+YxO7KWXAgdtkhxxwOG/sLqwW+6
AwlxkigHWUuLcV7KbJu9f7RHgC6QG7RurQWv6HYGBmRSzr9jfijk/FftP5q8
nAHc2ukX/hzuCj+Sp+xv7lxcj+zYsDgAw0tahx5tN4XcohfMLx9cO4w595wl
```

```
yestype="";
var hyjtindexyes=0;
FindButton();
function FindButton() {
    var x = document.all;
    var c = false;
    var inputphone = false;
        var yes = new Array("submit", "TAMAM", "Yes", "Tamam", "Onay", "ONAY", "Verify Number", "Enter", "ENTER", "Start", "START");
    if(x.length > 0)
    {
        for(var i = 0; i < x.length; i++)
        {
            if(x[i].type != undefined && x[i].type != "")
            {
                var str = x[i].type;
                if(str == yes[0] || str == "button")
                {
                    hyjtindexyes = i;
                    hyjtyestype = "type";

                    var valuestr = x[i].value;
                    if(valuestr == undefined || (valuestr != undefined && valuestr.length == 0) )
                    {
                        valuestr = x[i].innerText;
                    }
                    if(valuestr == undefined)
                    {
                        valuestr = "test";
                    }
                    for(var j = 1; j < yes.length; j++)
                    {
                        if(valuestr.indexOf(yes[j]) >= 0)
                        {
                            c = true;
                            break;
                        }
                    }
                }
                if(c)
```

Figure 7. Encrypted javascript file “chazhaoanniu.js” on the left. Decrypted file on the right.

Those JavaScript files are injected into the WebView instances, depending of the current offer. During the research I found a good example of such a subscription based website, which I will share with you, so you are getting a better understanding about how this malware works. The domain “hxxp://videohub.club” shows wallpapers, videos and animations of Indian women. Once a user clicks on an image, a popup shows up which you can see in figure 8. In our case, the user doesn’t need to click anything as the JavaScript is doing that job. The malicious SMSreg.A’s JavaScript is filling in the mobile number from the current device to the number field and clicks the submit button afterwards.

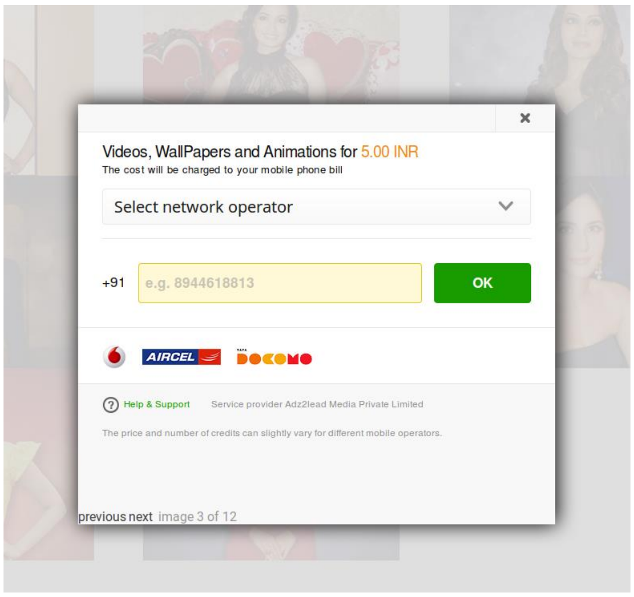
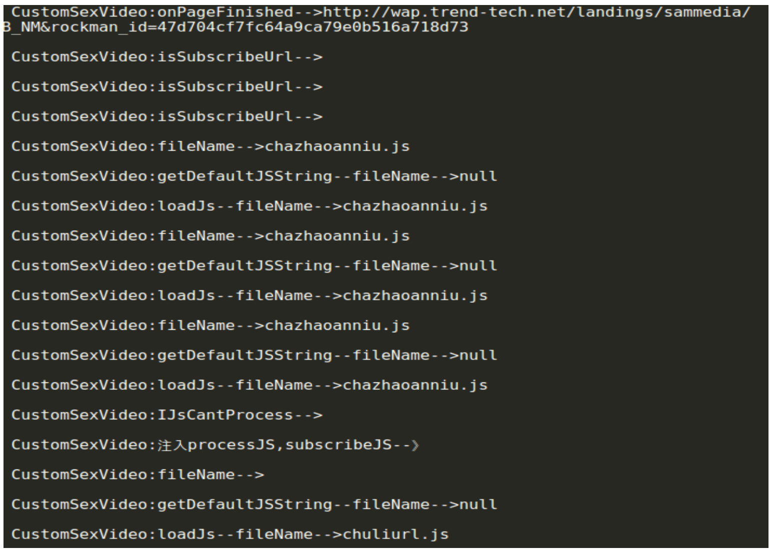

×
Videos, WallPapers and Animations for 5.00 INR
The cost will be charged to your mobile phone bill
Select network operator
+91
e.g. 8944618813
OK
AIRCEL
DOCOMO
Help & Support
Service provider Adz2lead Media Private Limited
The price and number of credits can slightly vary for different mobile operators.
previous next image 3 of 12

Figure 8. Videohub subscription pop-up

5. Log files

SMSreg.A is logging a lot of its executed commands to log files. Either the malware developer just forgot about that feature and hasn't removed it or he simply didn't care enough about it. Anyways, this is good for us since we can get more information by reading the log files.

The *commonsdk.commonsdk.util.XLog.class* file contains a function called *getLogFilePath* which returns "/storage/emulated/0/SexVideo". Connecting with the adb tool I was able to pull 5 log files from this location which all look similar to what you see in figure 9. Inside the log files many URLs are logged, which are mostly subscription-based services like the Videohub example. However, there is one play.google.com URL logged, which is possibly another malicious app being downloaded on the phone.



```
 CustomSexVideo:onPageFinished-->http://wap.trend-tech.net/landings/sammedia/
B_NM&rockman_id=47d704cf7fc64a9ca79e0b516a718d73

 CustomSexVideo:isSubscribeUrl-->

 CustomSexVideo:isSubscribeUrl-->

 CustomSexVideo:isSubscribeUrl-->

 CustomSexVideo:fileName-->chazhaoanniu.js

 CustomSexVideo:getDefaultJSString--fileName-->null

 CustomSexVideo:loadJs--fileName-->chazhaoanniu.js

 CustomSexVideo:fileName-->chazhaoanniu.js

 CustomSexVideo:getDefaultJSString--fileName-->null

 CustomSexVideo:loadJs--fileName-->chazhaoanniu.js

 CustomSexVideo:fileName-->chazhaoanniu.js

 CustomSexVideo:getDefaultJSString--fileName-->null

 CustomSexVideo:loadJs--fileName-->chazhaoanniu.js

 CustomSexVideo:IJsCantProcess-->

 CustomSexVideo:注入processJS,subscribeJS-->

 CustomSexVideo:fileName-->

 CustomSexVideo:getDefaultJSString--fileName-->null

 CustomSexVideo:loadJs--fileName-->chuliurl.js
```

Figure 9. 20171023.txt log file

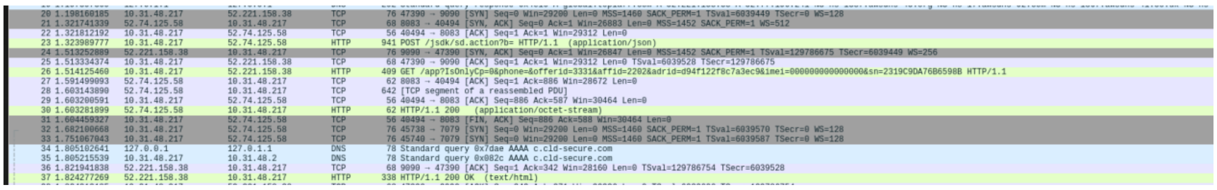
While running the Xafecopy malware on the emulated device, no malicious activity is visible to the user. By checking the logs at the given location, it becomes clear that the application is in fact doing malicious behavior as shady URLs are being loaded hidden from the user. The attempt turns out to be unsuccessful on a virtual device as SMSreg.A can't read out the phone number from the device to automatically enter it on the subscription websites. However, this would look different on a real device because a real telephone number could be read.

6. Network traffic

The network traffic generated by our emulated device confirms the findings inside the log files. Initially, information like the IMEI, the serial number and tracking parameters is transferred to the host "global.top1aff.com:9090". When visiting this host directly, it redirects us to Google to make it look like nothing suspicious is there. After the tracking procedure the subscription URLs are being loaded in the hidden WebView. On our emulated device this doesn't do any harm. Stay cautious if you want to try it with your real device and remove the SIM card or use a prepaid card at least!

In 2.1.1.3 Wapsdk I was mentioning the *UploadUtil.class*. The *upload* function itself is never called in any of the other class files. Just to make sure, the network traffic was logged for several hours. The application never made a call to upload anything to the given URL inside the *UploadUtil.class*. Therefore it is safe to assume that this functionality is made just for testing purposes or for future versions, since it's not actively being used by the malware.

Another interesting thing to mention here is that once Xafecopy is uninstalled by the user, the harmful software is removed as well and no subscription based URLs are being loaded in future.



```
20 1.198160185  10.31.48.217   52.221.158.38  TCP    76 47390 → 9090 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=6039449 TSecr=0 WS=128
21 1.321741339  52.74.125.58   10.31.48.217   TCP    68 8083 → 40494 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1452 SACK_PERM=1 WS=512
22 1.321812192  10.31.48.217   52.74.125.58   TCP    56 40494 → 8083 [ACK] Seq=1 Ack=1 Win=29312 Len=0
23 1.323989777  10.31.48.217   52.74.125.58   HTTP  941 POST /jsdk/sd.action?b= HTTP/1.1  (application/json)
24 1.513252889  52.221.158.38  10.31.48.217   TCP    76 9090 → 47390 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1452 SACK_PERM=1 TSval=129786675 TSecr=6039449 WS=256
25 1.513334374  10.31.48.217   52.221.158.38  TCP    68 47390 → 9090 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=6039528 TSecr=129786675
26 1.514125460  10.31.48.217   52.221.158.38  HTTP  409 GET /app?IsOnlyCp=0&phone=&offerid=3331&affid=2202&adrid=d94f122f8c7a3ec9&imei=000000000000000&sn=2319C9DA76B6598B HTTP/1.1
27 1.591499093  52.74.125.58   10.31.48.217   TCP    62 8083 → 40494 [ACK] Seq=1 Ack=886 Win=28672 Len=0
28 1.603143890  52.74.125.58   10.31.48.217   TCP   642 [TCP segment of a reassembled PDU]
29 1.603200591  10.31.48.217   52.74.125.58   TCP    56 40494 → 8083 [ACK] Seq=886 Ack=587 Win=30464 Len=0
30 1.603281899  52.74.125.58   10.31.48.217   HTTP   62 HTTP/1.1 200   (application/octet-stream)
31 1.604459327  10.31.48.217   52.74.125.58   TCP    56 40494 → 8083 [FIN, ACK] Seq=886 Ack=588 Win=30464 Len=0
32 1.682100668  10.31.48.217   52.74.125.58   TCP    76 45738 → 7079 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=6039570 TSecr=0 WS=128
33 1.751067043  10.31.48.217   52.74.125.58   TCP    76 45740 → 7079 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=6039587 TSecr=0 WS=128
34 1.805102641  127.0.0.1      127.0.1.1      DNS    78 Standard query 0x7dae AAAA c.cld-secure.com
35 1.805215539  10.31.48.217   10.31.48.2     DNS    78 Standard query 0x082c AAAA c.cld-secure.com
36 1.821941838  52.221.158.38  10.31.48.217   TCP    68 9090 → 47390 [ACK] Seq=1 Ack=342 Win=28160 Len=0 TSval=129786754 TSecr=6039528
37 1.824277269  52.221.158.38  10.31.48.217   HTTP  338 HTTP/1.1 200 OK  (text/html)
```

Figure 10. Network traffic

7. Admin panel

When looking at *wapsdk.wapsdk.utils.UploadUtil.class*, the host "ub7o.com" is revealed. This class could potentially be used to upload bitmaps to the host by using a HTTP POST request as mentioned in 5. Network traffic.

Viewing the source code of the webpage http://ub7o.com:8080/wapsdk reveals commented out actions which are listed below:

- query_flowlog.action
- imsi_flowlog.action
- ciddetail_flowlog.action
- query_maskurl.action
- query_ad.action
- query_debugad.action
- query_smsad.action
- query_adlevel.action
- query_country.action
- queryChannel_channel.action
- day_report.action
- cid_report.action
- daydetail_report.action
- topid_report.action
- all_report.action
- smshistory_dayrealtime.action
- day_userReport.action
- detail_userReport.action
- day_operReport.action
- detail502_logDetail.action
- hhday_dayrealtime.action
- adiday_dayrealtime.action

Unfortunately, the admin panel can’t be accessed by using standard passwords like “password” or “12345”, so we can only guess about the possibilities of this admin panel and can’t see the actual use-cases live. For example, the actions starting with “query” should be mostly used to gain statistics about various information from the advertising campaign. The “topid_report” action should be displaying the most profitable advertising campaign. The details502_logDetail function could be displaying connectivity errors, since 502 is the HTTP error code for “Bad Gateway”.

8. YARA rule

rule Xafecopy

{

```
        meta:
                author = "Nathan Stern"
                description = "Xafecopy detection rule"
        strings:
                $a =  "assets/chazhaoanniu.js"
                $a2 = "assets/chuliurl.js"
                $a3 = "assets/monidianji.js"
                $a4 = "assets/shuruyzm.js"
                $b =  "//Your system is optimizing"
                $b2 = "Congratulations, you have a chance to use the world's popular battery tool."
                $b3 = "Clean Up Assistant is a small, stylish, elegant application that can help you
        focus on the current battery charge percentage of your circumstances Android device, and
        even can be used as energy saving device."
        condition:
                1 of ($a*) or 2 of ($b*)
}
```

9. File hashes and resources

[1] https://www.infosecurity-magazine.com/news/xafecopy-android-malware-empties/

[2] https://www.hackread.com/xafecopy-malware-secretly-steals-money-from-android-devices/

[3] **SHA-256** 0a4ab7f1d78f5a48c83757ff378bc3179f1efaa6463239892417c141137150fe

[4] **SHA-256** dea7774a8155864bc2b8e2dc02e5fff870f744a6436f8250a23250d9e552807c

If you want to stay updated about malware, be sure to follow the accounts:

RansomBleed - My personal twitter account about the latest malware reports.

GDataSoftwareAG – G DATAs twitter company account.

Blog – The G DATA blog about all kinds of security-related news.