XLog & 日志平台使用

XLog的使用

注意：不要打印用户敏感信息

QQ阅读主线将XLog集成到com.qq.reader.common.monitor.debug.**Logger**下。
同时主工程中的常用日志工具类（例如：com.qq.reader.common.log.**QRLogger**，
com.qq.reader.common.monitor.**Log**），相关接口方法都已切换为**Logger**实现。

XLog目前配置收集Verbose及以上级别日志，但为了减少日志量，**Logger默认仅在Warn和Error级别使用XLog采集日志**，其他情况均使用android.util.Log打印日志

如果想采集Verbose、Debug、Info级别日志到XLog中，请使用下面的方法

> **Logger.v(String tag, final String log, boolean isWrite2File)**
> **Logger.d(String tag, final String log, boolean isWrite2File)**
> **Logger.i(String tag, final String log, boolean isWrite2File)**

V7.0.8 ~

日志上传

入口：

- QQ阅读客户端/我的/设置/意见反馈/所有意见反馈详情页（已登录）
  填写意见反馈内容后，勾选下方上传日志的选项后，提交即可。
- 或者使用，手动选择日期上传页面，此页面仅能通过 QURL 跳转。 QURL：
  uniteqqreader://nativepage/client/log

下载 & 查看日志

测试环境：http://bookmanager.bookcs.3g.qq.com/
正式环境：http://bookmanager.inner.yuewen.local/

选择顶部`用户管理`后，在左侧菜单栏依次进入`用户运营--用户反馈（新）--Android平台--用户反馈`中查看用户反馈并下载日志文件

运营平台在用户下载过程中进行解密操作，下载下来的日志zip包中“log”后缀的文件为日志文件，“xlog”文件为日志原文件，如日志文件发现乱码破损等异常时，可以按下面的解密步骤自行解密。

V7.0.7

日志上传

入口：QQ阅读客户端/我的/设置/意见反馈/所有意见反馈详情页（已登录）

填写意见反馈内容后，勾选下方上传日志的选项后，提交即可。

下载 & 查看日志

测试环境：http://bookmanager.bookcs.3g.qq.com/
正式环境：http://bookmanager.inner.yuewen.local/

选择顶部`用户管理`后，在左侧菜单栏依次进入`用户运营--用户反馈（新）--Android平台--用户反馈`中查看用户反馈并下载日志文件

运营平台在用户下载过程中进行解密操作，下载下来的日志zip包中“log”后缀的文件为日志文件，“xlog”文件为日志原文件，如日志文件发现乱码破损等异常时，可以按下面的解密步骤自行解密。

v7.0.3~v7.0.6

日志上传

入口：QQ阅读客户端/我的/设置/意见反馈/日志上报
QURL：uniteqqreader://nativepage/client/log

在日志上传界面选择日期，填写联系账号后，点击上传即可。

联系账号必须先在日志平台（登录日志平台后，在左侧菜单栏进入**用户日志--指定用户**）中配置白名单，或者使用已配置的通用白名单账号**88888888**，否则无法上传日志

下载日志

测试环境：http://ptqdrec.reader.qq.com
正式环境：https://qdrec.reader.qq.com

登录账号和密码
QQReaderRD
ln8W3IX6

登录日志平台后，在左侧菜单栏进入**用户日志--日志列表**，输入关键词查找对应日志并下载

解密&查看日志

1. 解压上一步中下载的zip文件到指定目录
2. 下载解密脚本，三种方式。
   - 直接使用主工程目录下的解密脚本`./doc/xlog/decode_mars_crypt_log_file.py`
   - 将下面的**解密脚本**代码复制并粘贴到文本文件中，并将文件名和后缀修改为`decode_mars_crypt_log_file.py`(python代码)
   - 从github工程中下载最新的解密脚本https://github.com/Tencent/mars/blob/master/mars/log/crypt/decode_mars_crypt_log_file.py ，并将其中的公钥私钥修改下面的公钥和私钥
3. 将此py文件拷贝到第1步中的目录中
4. 参考官方教程（https://github.com/Tencent/mars/wiki/Xlog-%E5%8A%A0%E5%AF%86%E4%BD%BF%E7%94%A8%E6%8C%87%E5%BC%95 ）安装环境
5. 最后然后运行`python decode_mars_crypt_log_file.py xxx.xlog`进行解密，解密成功后的文件为`xxx.xlog.log`文件。（如运行出错，参考下面“解密失败”解决）
6. 使用文件编辑器即查看`.log`文件

解密失败

> **错误：** `ImportError: No module named pyelliptic`
> 原因：没有正确安装 pyelliptic 的依赖库
> 解决: 运行命令 `pip install pyelliptic==1.5.7` 安装依赖库
> 如果安装后依然报错，可能是由于安装了多个版本的 python,pyelliptic 并没有安装在 2.+ 的 python 下. 在 MacOS 上，如果同时安装了 2.+ 和 3.+ 的 python, pip2 和 pip3 分别代表安装到 2.+ 和 3.+ 的库中。所以可以通过 pip2 install pyelliptic==1.5.7 安装

> **错误：** `AttributeError: dlsym (0x7fc443f02f50, EVP_CIPHER_CTX_reset): symbol not found`
> 原因: pyelliptic 的最新版本 1.5.8 的 bug
> 解决：卸载pyelliptic并重新安装使用 1.5.7 版本

```
pip uninstall pyelliptic
pip install pyelliptic==1.5.7
```

重新编译XLog所需so文件

目前端内XLog在所有渠道都已添加 `armeabi` CPU架构 so，google play 渠道额外添加 `arm64-v8a` 架构 so。 若因需要修改XLog加密方式、添加额外 CPU架构所需 so，或修改加密秘钥，请参考Mars开源项目（https://github.com/Tencent/mars ）中的接入说明，自行编译。

当前使用的秘钥为：

PRIV_KEY千万不可泄露
PRIV_KEY千万不可泄露
PRIV_KEY千万不可泄露

```
PRIV_KEY = "f7d791fbb3c9d129fdb3da0bbfbcedd0c858a5b0d2653ec6bc977d87139fd812"
PUB_KEY = "72946e01ac3f5398c28f37f4e6d9ca8193b06280bb9e27b19ae4e770a319490eba307427dbd2
db5043f3904265a291ec9106853a9a054223e4b0bf8e857d6e17"
```

解密脚本

PRIV_KEY千万不可泄露
PRIV_KEY千万不可泄露
PRIV_KEY千万不可泄露

```
#!/usr/bin/python

import sys
import os
import glob
import zlib
import struct
import binascii
import pyelliptic
import traceback


MAGIC_NO_COMPRESS_START = 0x03
MAGIC_NO_COMPRESS_START1 = 0x06
MAGIC_NO_COMPRESS_NO_CRYPT_START = 0x08
MAGIC_COMPRESS_START = 0x04
MAGIC_COMPRESS_START1 = 0x05
MAGIC_COMPRESS_START2 = 0x07
MAGIC_COMPRESS_NO_CRYPT_START = 0x09

MAGIC_END = 0x00

lastseq = 0
```

```
PRIV_KEY = "f7d791fbb3c9d129fdb3da0bbfbcedd0c858a5b0d2653ec6bc977d87139fd812"
PUB_KEY = "72946e01ac3f5398c28f37f4e6d9ca8193b06280bb9e27b19ae4e770a319490eba307427dbd2
db5043f3904265a291ec9106853a9a054223e4b0bf8e857d6e17"

def tea_decipher(v, k):
    op = 0xffffffffL
    v0, v1 = struct.unpack('=LL', v[0:8])
    k1, k2, k3, k4 = struct.unpack('=LLLL', k[0:16])
    delta = 0x9E3779B9L
    s = (delta << 4) & op
    for i in xrange(16):
        v1 = (v1 - (((v0<<4) + k3) ^ (v0 + s) ^ ((v0>>5) + k4))) & op
        v0 = (v0 - (((v1<<4) + k1) ^ (v1 + s) ^ ((v1>>5) + k2))) & op
        s = (s - delta) & op
    return struct.pack('=LL', v0, v1)


def tea_decrypt(v, k):
    num = len(v) / 8 * 8
    ret = ''
    for i in xrange(0, num, 8):
        x = tea_decipher(v[i:i+8], k)
        ret += x

    ret += v[num:]
    return ret


def IsGoodLogBuffer(_buffer, _offset, count):

    if _offset == len(_buffer): return (True, '')

    magic_start = _buffer[_offset]
    if MAGIC_NO_COMPRESS_START==magic_start or MAGIC_COMPRESS_START==magic_start or MAG
IC_COMPRESS_START1==magic_start:
        crypt_key_len = 4
    elif MAGIC_COMPRESS_START2==magic_start or MAGIC_NO_COMPRESS_START1==magic_start or
 MAGIC_NO_COMPRESS_NO_CRYPT_START==magic_start or MAGIC_COMPRESS_NO_CRYPT_START==magic_
start:
        crypt_key_len = 64
    else:
        return (False, '_buffer[%d]:%d != MAGIC_NUM_START'%(_offset, _buffer[_offset]))

    headerLen = 1 + 2 + 1 + 1 + 4 + crypt_key_len

    if _offset + headerLen + 1 + 1 > len(_buffer): return (False, 'offset:%d > len(buff
er):%d'%(_offset, len(_buffer)))
    length = struct.unpack_from("I", buffer(_buffer, _offset+headerLen-4-crypt_key_len,
 4))[0]
    if _offset + headerLen + length + 1 > len(_buffer): return (False, 'log length:%d,
end pos %d > len(buffer):%d'%(length, _offset + headerLen + length + 1, len(_buffer)))
    if MAGIC_END!=_buffer[_offset + headerLen + length]: return (False, 'log length:%d,
 buffer[%d]:%d != MAGIC_END'%(length, _offset + headerLen + length, _buffer[_offset + h
eaderLen + length]))


    if (1>=count): return (True, '')
    else: return IsGoodLogBuffer(_buffer, _offset+headerLen+length+1, count-1)


def GetLogStartPos(_buffer, _count):
    offset = 0
    while True:
        if offset >= len(_buffer): break

        if MAGIC_NO_COMPRESS_START==_buffer[offset] or MAGIC_NO_COMPRESS_START1==_buffe
r[offset] or MAGIC_COMPRESS_START==_buffer[offset] or MAGIC_COMPRESS_START1==_buffer[of
fset] or MAGIC_COMPRESS_START2==_buffer[offset] or MAGIC_COMPRESS_NO_CRYPT_START==_buff
er[offset] or MAGIC_NO_COMPRESS_NO_CRYPT_START==_buffer[offset]:
```

```
            if IsGoodLogBuffer(_buffer, offset, _count)[0]: return offset
        offset+=1

    return -1

def DecodeBuffer(_buffer, _offset, _outbuffer):

    if _offset >= len(_buffer): return -1
    # if _offset + 1 + 4 + 1 + 1 > len(_buffer): return -1
    ret = IsGoodLogBuffer(_buffer, _offset, 1)
    if not ret[0]:
        fixpos = GetLogStartPos(_buffer[_offset:], 1)
        if -1==fixpos:
            return -1
        else:
            _outbuffer.extend("[F]decode_log_file.py decode error len=%d, result:%s \n"
%(fixpos, ret[1]))
            _offset += fixpos

    magic_start = _buffer[_offset]
    if MAGIC_NO_COMPRESS_START==magic_start or MAGIC_COMPRESS_START==magic_start or MAG
IC_COMPRESS_START1==magic_start:
        crypt_key_len = 4
    elif MAGIC_COMPRESS_START2==magic_start or MAGIC_NO_COMPRESS_START1==magic_start or
 MAGIC_NO_COMPRESS_NO_CRYPT_START==magic_start or MAGIC_COMPRESS_NO_CRYPT_START==magic_
start:
        crypt_key_len = 64
    else:
        _outbuffer.extend('in DecodeBuffer _buffer[%d]:%d != MAGIC_NUM_START'%(_offset,
 magic_start))
        return -1

    headerLen = 1 + 2 + 1 + 1 + 4 + crypt_key_len
    length = struct.unpack_from("I", buffer(_buffer, _offset+headerLen-4-crypt_key_len,
 4))[0]
    tmpbuffer = bytearray(length)

    seq=struct.unpack_from("H", buffer(_buffer, _offset+headerLen-4-crypt_key_len-2-2,
2))[0]
    begin_hour=struct.unpack_from("c", buffer(_buffer, _offset+headerLen-4-crypt_key_le
n-1-1, 1))[0]
    end_hour=struct.unpack_from("c", buffer(_buffer, _offset+headerLen-4-crypt_key_len-
1, 1))[0]

    global lastseq
    if seq != 0 and seq != 1 and lastseq != 0 and seq != (lastseq+1):
        _outbuffer.extend("[F]decode_log_file.py log seq:%d-%d is missing\n" %(lastseq+
1, seq-1))

    if seq != 0:
        lastseq = seq

    tmpbuffer[:] = _buffer[_offset+headerLen:_offset+headerLen+length]

    try:
        decompressor = zlib.decompressobj(-zlib.MAX_WBITS)

        if MAGIC_NO_COMPRESS_START1==_buffer[_offset]:
            pass

        elif MAGIC_COMPRESS_START2==_buffer[_offset]:
            svr = pyelliptic.ECC(curve='secp256k1')
            client = pyelliptic.ECC(curve='secp256k1')
            client.pubkey_x = str(buffer(_buffer, _offset+headerLen-crypt_key_len, cryp
t_key_len/2))
            client.pubkey_y = str(buffer(_buffer, _offset+headerLen-crypt_key_len/2, cr
ypt_key_len/2))

            svr.privkey = binascii.unhexlify(PRIV_KEY)
```

```
            tea_key = svr.get_ecdh_key(client.get_pubkey())

            tmpbuffer = tea_decrypt(tmpbuffer, tea_key)
            tmpbuffer = decompressor.decompress(str(tmpbuffer))
        elif MAGIC_COMPRESS_START==_buffer[_offset] or MAGIC_COMPRESS_NO_CRYPT_START==_
buffer[_offset]:
            tmpbuffer = decompressor.decompress(str(tmpbuffer))
        elif MAGIC_COMPRESS_START1==_buffer[_offset]:
            decompress_data = bytearray()
            while len(tmpbuffer) > 0:
                single_log_len = struct.unpack_from("H", buffer(tmpbuffer, 0, 2))[0]
                decompress_data.extend(tmpbuffer[2:single_log_len+2])
                tmpbuffer[:] = tmpbuffer[single_log_len+2:len(tmpbuffer)]

            tmpbuffer = decompressor.decompress(str(decompress_data))

        else:
            pass

            # _outbuffer.extend('seq:%d, hour:%d-%d len:%d decompress:%d\n' %(seq, ord(
begin_hour), ord(end_hour), length, len(tmpbuffer)))
    except Exception, e:
        traceback.print_exc()
        _outbuffer.extend("[F]decode_log_file.py decompress err, " + str(e) + "\n")
        return _offset+headerLen+length+1

    _outbuffer.extend(tmpbuffer)

    return _offset+headerLen+length+1


def ParseFile(_file, _outfile):
    fp = open(_file, "rb")
    _buffer = bytearray(os.path.getsize(_file))
    fp.readinto(_buffer)
    fp.close()
    startpos = GetLogStartPos(_buffer, 2)
    if -1==startpos:
        return

    outbuffer = bytearray()

    while True:
        startpos = DecodeBuffer(_buffer, startpos, outbuffer)
        if -1==startpos: break;

    if 0==len(outbuffer): return

    fpout = open(_outfile, "wb")
    fpout.write(outbuffer)
    fpout.close()

def main(args):
    global lastseq

    if 1==len(args):
        if os.path.isdir(args[0]):
            filelist = glob.glob(args[0] + "/*.xlog")
            for filepath in filelist:
                lastseq = 0
                ParseFile(filepath, filepath+".log")
        else: ParseFile(args[0], args[0]+".log")
    elif 2==len(args):
        ParseFile(args[0], args[1])
    else:
        filelist = glob.glob("*.xlog")
        for filepath in filelist:
            lastseq = 0
            ParseFile(filepath, filepath+".log")
```

```
if __name__ == "__main__":
    main(sys.argv[1:])
```