瑞星自主安全引擎软件说明书

瑞星网安技术股份有限公司 | 北京市海淀区中关村大街22号中科大厦B座5层 010-82678866

2017

目录

[概述 1](#_Toc520552758)

[使用说明 1](#_Toc520552759)

[安装 1](#_Toc520552760)

[授权文件生成 1](#_Toc520552761)

[适应语言 1](#_Toc520552762)

[操作系统 1](#_Toc520552763)

[注意事项 1](#_Toc520552764)

[性能参数 1](#_Toc520552765)

[查杀速度 1](#_Toc520552766)

[异步多线程查杀 1](#_Toc520552767)

[产品升级 1](#_Toc520552768)

[升级方式 1](#_Toc520552769)

[覆盖升级方法 2](#_Toc520552770)

[升级内容 2](#_Toc520552771)

[手工升级办法 2](#_Toc520552772)

[升级后工作异常处理办法 2](#_Toc520552773)

[增量升级方法 3](#_Toc520552774)

[libup用法 3](#_Toc520552775)

[libup使用实例 3](#_Toc520552776)

[homeup用法 3](#_Toc520552777)

[homeup使用实例 3](#_Toc520552778)

[中心升级方法 4](#_Toc520552779)

[接口说明 4](#_Toc520552780)

[C/C++ 4](#_Toc520552781)

[V 1.0 4](#_Toc520552782)

[V 1.1 11](#_Toc520552783)

[DotNet接口 18](#_Toc520552784)

[病毒库 18](#_Toc520552785)

[Lame 19](#_Toc520552786)

[LameWithEvent 21](#_Toc520552787)

[PackageExtract 22](#_Toc520552788)

[Python接口 22](#_Toc520552789)

[病毒库 22](#_Toc520552790)

[Lame 23](#_Toc520552791)

[LameWithFeedback 25](#_Toc520552792)

[LameExtractor 26](#_Toc520552793)

[Java接口 28](#_Toc520552794)

[病毒库 28](#_Toc520552795)

[Lame 28](#_Toc520552796)

[go接口 32](#_Toc520552797)

[病毒库 32](#_Toc520552798)

[Lame 33](#_Toc520552799)

[附录 38](#_Toc520552800)

[常用参数设置 38](#_Toc520552801)

# 概述

本软件开发包（SDK）为瑞星网安技术股份有限公司完全自主知识产权产品，采用瑞星最先进的四核杀毒引擎并经优化而形成的独立型通用SDK。本SDK有速度快、资源占用稳定、易使用、易扩展等特点，其可应用于各种二次开发研发。

# 使用说明

## 安装

将lame.tgz解压到任意路径，编写配置文件rame.cfg，PATH后面是lame当前的绝对路径，并把配置文件放到应用程序当前的目录下。文件内容如下所示：

PATH:/root/lib/lame

## 授权文件生成

* 采集硬件信息

与用户达成销售协议后，用户在客户端通过运行硬件信息提取工具（mcgen），将生成MD5值提供给瑞星。

* 生成有效序列号

瑞星根据用户提供的MD5信息生成形如“licence.json”的授权文件供用户使用。此校验文件生成后与该硬件绑定，如更换硬件，需再次向瑞星提供新硬件的信息重新生成MD5值才能正常使用引擎SDK的扫描功能。

## 适应语言

C、C++、Java、C#、.Net、.Net core、nodejs、python、go

## 操作系统

Windows系列、RedHat系列、ubuntu系列、centos系列、Debian系列、Arm系列及各国产操作系统等

## 注意事项

1. 在使用引擎SDK的时候需要在调用liblame.so的应用目录下把rame.cfg配置好
2. 需要先设置LD\_LIBRARY\_PATH=引擎SDK所在的目录

# 性能参数

## 查杀速度

参考硬件环境：内存2G以上，CPU在I5以上，硬盘7200转/秒以上

按流量：20M/秒左右

按文件数：50个文件/秒左右

## 异步多线程查杀

可支持

单引擎支持最大并发数100（具体并发数需参考硬件参数）

# 产品升级

## 升级方式

引擎SDK模块文件采取全文件覆盖升级

病毒库可以用覆盖升级也可用增量升级方式，同时还支持通过中心升级的方式

## 覆盖升级方法

### 升级内容

2013病毒库、2013引擎模块文件、SDK模块文件

### 手工升级办法

1. 通过有效途径获取到最新的瑞星病毒引擎SDK升级包，属性如下：

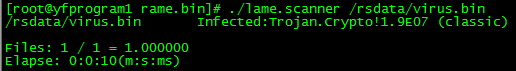
-rw-r--r-- 1 root root 52336431 5月 19 07:23 lame.tgz

压缩包内包含有如下内容：

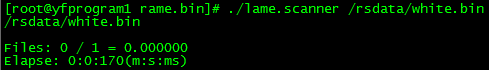
|  |  |
| --- | --- |
| homeup | 病毒库中心升级程序 |
| lame.extractor | 解包测试程序 |
| lame.jar | SDK的Java封装包 |
| lame.mt.scanner | SDK多线程示例程序（需支持c++11） |
| lame.scanner | SDK简单的扫描器程序 |
| libjlame.so | SDK封装java接口动态链接库 |
| liblame.so | SDK封装接口动态链接库 |
| libup | 病毒库客户端升级程序 |
| licence.json | 证书授权文件 |
| malware.rmd | 病毒库 |
| librxavx.so | 引擎动态链接库 |

1. 停止使用引擎SDK的服务。
2. 找到引擎SDK目录（lame），下面解说都以lame为例，改名为lame.bak。
3. 将引擎SDK包lame.tgz解压缩到引擎SDK目录（lame）下。
4. 再将备份的licence.json引擎SDK目录（lame）中。
5. 使用cd命令进入引擎SDK目录（lame）。
6. 在引擎SDK下（lame），使用“./lame.scanner /rsdata/virus.bin”检查引擎SDK工作状态

有病毒的情况：



无毒的情况：



如出现Elapse字样的扫描结果，表示扫描正常。

如果异常，则升级失败，按照“工作异常处理办法”进行后续处理。

1. 升级成功

### 升级后工作异常处理办法

1. 停止服务。
2. 找到引擎SDK目录（lame）。
3. 删除引擎SDK目录（lame）。
4. 将lame.bak目录更名到步骤3中的lame目录。
5. 再次重新启动服务。
6. 报告相关部门，升级发生故障，未能正常进行。
7. 异常处理完毕。

## 增量升级方法

此方法使用工具包中的libup工具进行升级，注意此升级方法为热升级，即可以在软件运行的过程中进行升级操作

### libup用法

升级本地病毒库文件

libup -loc {info file} {lib home} {want home}

libup -xml {xml url} {lib home}

libup -cfg {cfg url} {lib home}

### libup使用实例



### homeup用法

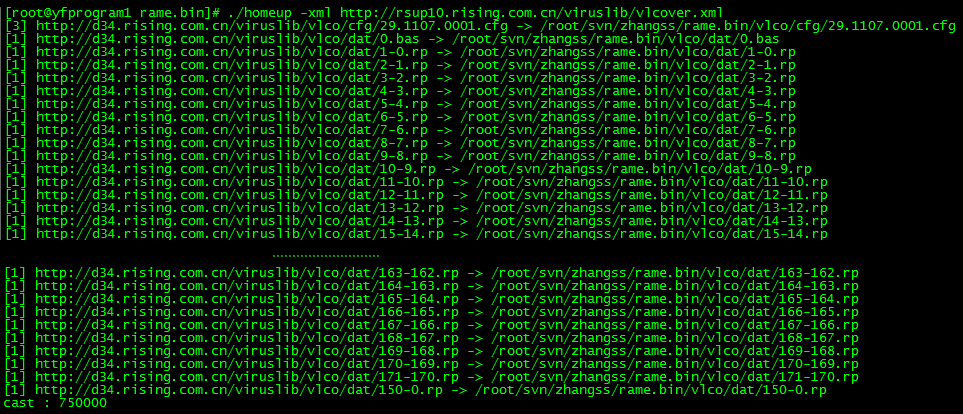
获取并升级中心系统的病毒库文件，包括bas文件、rp文件和配置文件

homeup -loc {info file} {local root} {remote root}

homeup -xml {xml url} {local root}

homeup -cfg {cfg url} {local root}

### homeup使用实例



## 中心升级方法

此方法使用工具包中的homeup、libup工具进行升级，这种方法相当于把瑞星的升级服务器移植到您的本地服务器上，客户端可连接本地服务器进行增量升级操作。

具体操作步骤如下：

1. 使用homeup对本地服务器进行升级（需要互联网）

homeup –xml <http://rsup10.rising.com.cn/viruslib/vlcover.xml> {病毒库数据文件存放目录}

1. 客户端使用libup连接本地服务器进行升级（需要局域网）

libup –cfg {本地服务器中cfg文件的链接} {病毒库文件目录}

# 接口说明

## C/C++

### V 1.0

#### 接口

扫描接口在liblame.so中，具体说明如下：

##### rx\_obj\_create

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C [rx\_obj\_t](#_rx_obj_t) rx\_obj\_create([rx\_obj\_id\_t](#_rx_obj_id_t) objid);  // object has been referenced |
| 功能 | 根据传入的参数创建引擎对象、配置参数、扫描结果 |
| 参数 | objid：需要创建对象的类型，  参见[rx\_obj\_id\_t](#_rx_obj_id_t) |
| 返回值 | 成功：非NULL  失败：NULL |

##### r**x\_obj\_reference**

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C void rx\_obj\_reference([rx\_obj\_t](#_rx_obj_t) obj); |
| 功能 | 引用创建的对象 |
| 参数 | obj：[rx\_obj\_create](#_rx_obj_create)创建的对象 |
| 返回值 | 无 |

##### rx\_obj\_release

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C void rx\_obj\_release([rx\_obj\_t](#_rx_obj_t) obj); |
| 功能 | obj：释放创建的对象 |
| 参数 | obj：[rx\_obj\_create](#_rx_obj_create)创建的对象 |
| 返回值 | 无 |

##### r**x\_conf\_set\_json**

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long rx\_conf\_set\_json([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) content); |
| 功能 | 根据输入的json串配置参数 |
| 参数 | conf：引擎参数配置对象  content：json串指针 |
| 返回值 | 成功：0  失败：非0 |

##### rx\_conf\_set\_str

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long rx\_conf\_set\_str([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path, rx\_cstr\_t val); |
| 功能 | 配置参数项为字符串值 |
| 参数 | conf：引擎参数配置对象  path：参数配置项字符串指针  val：参数配置值字符串指针 |
| 返回值 | 成功：0  失败：非0 |

##### rx\_conf\_set\_int

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long rx\_conf\_set\_int([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path, const int64\_t val); |
| 功能 | 配置参数项为整型值 |
| 参数 | conf：引擎参数配置对象  path：参数配置项字符串指针  val：参数配置值整型值 |
| 返回值 | 成功：0  失败：非0 |

##### rx\_conf\_set\_bool

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long rx\_conf\_set\_bool([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path, const bool val); |
| 功能 | 配置参数项为布尔值 |
| 参数 | conf：引擎参数配置对象  path：参数配置项字符串指针  val：参数配置布尔值 |
| 返回值 | 成功：0  失败：非0 |

##### rx\_conf\_erase

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long rx\_conf\_erase([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path); |
| 功能 | 删除配置参数项 |
| 参数 | conf：引擎参数配置对象  path：参数配置项字符串指针 |
| 返回值 | 成功：0  失败：非0 |

##### rx\_conf\_get\_str

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long rx\_conf\_get\_str([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path, rx\_cstr\_t \* pval); |
| 功能 | 获取配置参数项字符串值 |
| 参数 | conf：引擎参数配置对象  path：参数配置项字符串指针  pval：指向字符串指针的指针 |
| 返回值 | 成功：0  失败：非0 |

##### rx\_conf\_get\_int

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long rx\_conf\_get\_int([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path, int64\_t \* pval); |
| 功能 | 获取配置参数项整型值 |
| 参数 | conf：引擎参数配置对象  path：参数配置项字符串指针  pval：指向整型值的指针 |
| 返回值 | 成功：0  失败：非0 |

##### rx\_conf\_get\_bool

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long rx\_conf\_get\_bool([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path, bool \* pval); |
| 功能 | 获取配置参数项布尔值 |
| 参数 | conf：引擎参数配置对象  path：参数配置项字符串指针  pval：指向布尔值的指针 |
| 返回值 | 成功：0  失败：非0 |

##### rx\_conf\_cloud\_init

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long rx\_conf\_cloud\_init([rx\_obj\_t](#_rx_obj_t) conf); |
| 功能 | 引擎云扫描功能初始化 |
| 参数 | conf：引擎配置参数对象 |
| 返回值 | 成功：0  失败：非0 |

##### rx\_lame\_init

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long rx\_lame\_init([rx\_obj\_t](#_rx_obj_t) eng, [rx\_obj\_t](#_rx_obj_t) conf); |
| 功能 | 引擎初始化 |
| 参数 | eng：引擎对象  conf：引擎配置参数对象 |
| 返回值 | 成功：0  失败：非0 |

##### rx\_lame\_scan

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C rx\_sdr\_t rx\_lame\_scan([rx\_obj\_t](#_rx_obj_t) eng, [rx\_cstr\_t](#_rx_cstr_t) rx\_uri, [rx\_scan\_opt\_t](#_rx_scan_opt_t) opt); |
| 功能 | 扫描文件并返回扫描结果 |
| 参数 | eng：引擎对象  rx\_uri：被扫描的文件路径  opt：扫描选项，  参见[rx\_scan\_opt\_t](#_rx_scan_opt_t) |
| 返回值 | 成功：返回扫描结果的对象  失败：NULL  参见[rx\_restult\_t](#_rx_result_t) |

##### rx\_lame\_scan\_memory

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C rx\_sdr\_t rx\_lame\_scan\_memory([rx\_obj\_t](#_rx_obj_t) eng, [rx\_cstr\_t](#_rx_cstr_t) buffer, int length); |
| 功能 | 扫描文件内容流并返回扫描结果 |
| 参数 | eng：引擎对象  buffer：指向文件内容的内存指针  length：指向文件内容的内存长度 |
| 返回值 | 成功：返回扫描结果的对象  失败：NULL  参见[rx\_restult\_t](#_rx_result_t) |

##### rx\_sdr\_get\_result

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C bool rx\_sdr\_get\_result([rx\_sdr\_t](#_rx_sdr_t) sdr, [rx\_result\_t](#_rx_result_t) \* result); |
| 功能 | 解析引擎扫描的返回结果 |
| 参数 | sdr：引擎扫描的结果对象  result：指向[rx\_result\_t](#_rx_result_t)的指针 |
| 返回值 | 成功：有病毒返回true  失败：false |

##### rx\_lame\_get\_version

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C bool rx\_lame\_get\_version([rx\_str\_t](#_rx_str_t) ver, int length); |
| 功能 | 返回引擎当前版本 |
| 参数 | ver：指向字符串的指针，存储返回的版本信息  length：ver指针的大小 |
| 返回值 | 成功：true  失败：false |

##### rx\_lame\_get\_virus\_library\_version

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C bool rx\_lame\_get\_virus\_library\_version([rx\_str\_t](#_rx_str_t) ver, int length, [rx\_cstr\_t](#_rx_cstr_t) vlcfgpath); |
| 功能 | 返回病毒库当前版本 |
| 参数 | ver：指向字符串的指针，存储返回的版本信息  length：ver指针的大小  vlibpath：病毒库配置文件的绝对路径，如果为NULL，则认为是当前目录下的malware.cfg |
| 返回值 | 成功：true  失败：false |

##### rx\_lame\_get\_licence\_info

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C bool rx\_lame\_get\_licence\_info ([rx\_licence\_info](#_rx_licence_info) \* info, [rx\_cstr\_t](#_rx_cstr_t) path); |
| 功能 | 返回lame的授权文件信息 |
| 参数 | info：指向[rx\_licence\_info](#_rx_licence_info)结构的指针，存储返回的授权文件的信息  path：授权文件的绝对路径，如果为NULL，则认为是当前目录下的licence.json |
| 返回值 | 成功：true  失败：false |

#### 数据结构类型说明

##### rx\_obj\_id\_t

* 定义

typedef enum rx\_obj\_id\_t {

COBJ\_LAME, //创建引擎对象

COBJ\_CONF, //创建参数对象

COBJ\_RESULT, //创建结果对象

} rx\_obj\_id\_t;

* 说明

无

##### rx\_scan\_opt\_t

* 定义

typedef enum rx\_scan\_opt\_t {

OPT\_NO\_COW = 1, //无缓存

OPT\_TREAT = 2, //杀毒处理

} rx\_scan\_opt\_t;

* 说明

无

##### rx\_trc\_t

* 定义

typedef enum rx\_treat\_result\_code\_t {

TREAT\_FAIL\_FIX = -1,

REAT\_OK\_DELETED = 1,

TREAT\_OK\_FIXED = 2,

} rx\_trc\_t;

* 说明

无

##### rx\_mk\_t

* 定义

typedef enum rx\_mk\_t {

mkTrait = 0,

mkHidden = 1,

mkHiddenFU = 2,

mkComplier = 3,

mkPacker = 4,

mkFormat = 5, // 文件格式

mkAppType = 6,

mkTrusted = 9,

mkMalware = 0x10,

mkTrojan,

mkBackdoor,

mkWorm,

mkRootkit,

mkExploit,

mkHackTool,

mkAdware,

mkStealer,

mkSpammer,

mkSpyware,

mkVirus,

mkJoke,

mkJunk,

mkPUA,

mkDownloader,

mkDropper,

mkRansom,

mkHoax,

mkRiskware,

mkUnwanted,

mkMonetizer, // 套现

mkMobileBase = 0xC0,

// Mobile

mkPayment = mkMobileBase, // 恶意扣费

mkPrivacy, // 隐私窃取

mkRemote, // 远程控制

mkSpread, // 恶意传播

mkExpense, // 资费消耗

mkSystem, // 系统破坏

mkFraud, // 诱骗欺诈

mkRogue, // 流氓行为

mkAttention = 0xFE, // 注意!

mkTypeMax = 0x100,

} rx\_mk\_t;

* 说明

无

##### rx\_result\_t

* 定义

typedef struct rx\_result\_t {

rx\_mk\_t mklass;

rx\_cstr\_t engid;

rx\_cstr\_t vname;

uint32\_t vid32;

uint64\_t vid40;

rx\_cstr\_t hitag;

rx\_trc\_t treat;

} rx\_result\_t;

* 说明

无

##### rx\_obj\_t

* 定义

typedef void \* rx\_obj\_t;

* 说明

无

##### rx\_sdr\_t

* 定义

typedef rx\_obj\_t rx\_sdr\_t;

* 说明

无

##### rx\_sfr\_t

* 定义

typedef rx\_obj\_t rx\_sfr\_t;

* 说明

无

##### rx\_ptr\_t

* 定义

typedef void \* rx\_ptr\_t;

* 说明

无

##### rx\_cstr\_t

* 定义

typedef const char \*rx\_cstr\_t;

* 说明

无

##### rx\_str\_t

* 定义

typedef char \*rx\_str\_t;

* 说明

无

##### rx\_licence\_info

* 定义

typedef struct rx\_licence\_info

{

char Version[16];

char Owner[128];

char Date[64];

char Authm[32];

char Data[2048];

} rx\_licence\_info;

* 说明

无

#### 代码编译执行示例

scan.c

-----------------------------------------------------------------------------------------------------------------------------------------

#include "lame.h"

int main(int argc, char\* argv[])

{

rx\_obj\_t conf, lame;

rx\_sdr\_t sdr;

rx\_result\_t result;

if (argc == 1) return -1;

conf = rx\_obj\_create(COBJ\_CONF);

if (!conf){

printf("rx\_obj\_create(COBJ\_CONF) failed.\n");

return -1;

}

rx\_conf\_set\_int(conf, "cloud", 1);

lame = rx\_obj\_create(COBJ\_LAME);

if (!lame)

{

printf("rx\_obj\_create(COBJ\_LAME) failed.\n");

rx\_obj\_release(conf);

return -1;

}

if(rx\_lame\_init(lame, conf) != 0)

{

printf("rx\_lame\_init failed.\n");

rx\_obj\_release(conf);

rx\_obj\_release(lame);

return -1;

}

sdr = rx\_lame\_scan(lame, argv[1], OPT\_NO\_COW);

if( rx\_sdr\_get\_result(sdr, &result) )

printf("%s infected:%s\n", argv[1] , result.vname );

else

printf("%s\n" , argv[1]);

if( sdr ) rx\_obj\_release( sdr );

rx\_obj\_release(conf);

rx\_obj\_release(lame);

return 0;

}

Makefile

-----------------------------------------------------------------------------------------------------------------------------------------

LAME\_DIR=/root/lame/

gcc scan.c -o $( LAME\_DIR )myscan -llame -lrxavx –L$( LAME\_DIR)

运行

-----------------------------------------------------------------------------------------------------------------------------------------

LD\_LIBRARY\_PATH=/root/lame ./myscan /root/virus.bin

### V 1.1

#### 接口

扫描接口在liblame.so中，具体说明如下：

##### lame\_open\_vdb

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C void\* lame\_open\_vdb(const char\* vlibf); |
| 功能 | 根据传入的路径参数打开病毒库，并返回其指针 |
| 参数 | vlibf：字符串指针，可为空（默认加载当前路径下的病毒库） |
| 返回值 | 成功：病毒库对象指针  失败：NULL |

##### lame\_close\_vdb

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C void lame\_close\_vdb(void\* vdb); |
| 功能 | 关闭病毒库，释放病毒库资源 |
| 参数 | vdb：[lame\_open\_vdb](#_lame_open_vdb)创建的对象 |
| 返回值 | 无 |

##### lame\_create

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C void\* lame\_create(void\* vdb); |
| 功能 | 创建引擎对象 |
| 参数 | vdb：由[lame\_open\_vdb](#_rx_obj_create)打开的病毒库对象指针 |
| 返回值 | 成功：病毒库对象指针  失败：NULL |

##### lame\_destroy

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C void lame\_destroy(void\* lame); |
| 功能 | 释放引擎资源 |
| 参数 | lame：由[lame\_create](#_lame_create)创建的引擎对象指针 |
| 返回值 | 无 |

##### lame\_param\_set

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_param\_set(void\* lame , const char\* param); |
| 功能 | 根据传入的字符串配置引擎参数 |
| 参数 | lame：由[lame\_create](#_lame_create)创建的引擎对象指针  param：指向参数配置项和值的字符串指针 |
| 返回值 | 成功：大于等于0  失败：小于0 |

##### lame\_init

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_init(void\* lame); |
| 功能 | 引擎初始化 |
| 参数 | lame：由[lame\_create](#_lame_create)创建的引擎对象指针 |
| 返回值 | 成功：大于等于0  失败：小于0 |

##### lame\_scan\_file

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_scan\_file(void\* lame , const char\* fname , [rx\_scan\_result](#_rx_result_t)\* pResult) ; |
| 功能 | 扫描文件并返回扫描结果 |
| 参数 | lame：由[lame\_create](#_lame_create)创建的引擎对象指针  fname：被扫描的文件路径  pResult：指向扫描结果的指针，参见[rx\_scan\_result](#_rx_scan_result) |
| 返回值 | 成功：大于等于0  失败：小于0 |

##### lame\_scan\_mem

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_scan\_mem(void\* lame , uint8\_t\* data , uint32\_t size , [rx\_scan\_result](#_rx_result_t)\* pResult); |
| 功能 | 扫描文件内容流并返回扫描结果 |
| 参数 | lame：由[lame\_create](#_lame_create)创建的引擎对象指针  data：指向文件内容的内存指针  size：指向文件内容的内存长度  pResult：指向扫描结果的指针，  参见[rx\_scan\_result](#_rx_scan_result) |
| 返回值 | 成功：大于等于0  失败：小于0 |

##### lame\_get\_version

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_get\_version([lame\_info](#_lame_info)\* info); |
| 功能 | 返回引擎及病毒库的版本 |
| 参数 | info：指向版本信息的指针，  参见[lame\_info](#_lame_info) |
| 返回值 | 成功：大于等于0  失败：小于0 |

##### lame\_get\_licence\_info

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_get\_licence\_info([rx\_licence\_info](#_rx_licence_info)\* info); |
| 功能 | 返回lame的授权文件信息 |
| 参数 | info：指向[rx\_licence\_info](#_rx_licence_info)结构的指针，存储返回的授权文件的信息 |
| 返回值 | 成功：大于等于0  失败：小于0 |

##### lame\_extract\_file

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_extract\_file(void\* lame , const char\* fname , const char\* password, [lame\_extract\_feedback](#_lame_extract_feedback)\* cb , void\* userdata); |
| 功能 | 解压缩包格式的文件，可通过回调函数进行文件dump |
| 参数 | lame：由[lame\_create](#_lame_create)创建的引擎对象指针  fname：被扫描的文件路径  password：压缩包密码  cb：指向[lame\_extract\_feedback](#_lame_extract_feedback)的回调函数结构体指针  userdata：用户自定义数据，供回调函数使用 |
| 返回值 | 成功：大于等于0  失败：小于0 |

##### lame\_extract\_file\_w

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_extract\_file\_w(void\* lame , const wchar\_t\* fname , const wchar\_t \* password, [lame\_extract\_feedback](#_lame_extract_feedback)\* cb , void\* userdata); |
| 功能 | 解压缩包格式的文件，可通过回调函数进行文件dump |
| 参数 | lame：由[lame\_create](#_lame_create)创建的引擎对象指针  fname：被扫描的文件路径  password：压缩包密码  cb：指向[lame\_extract\_feedback](#_lame_extract_feedback)的回调函数结构体指针  userdata：用户自定义数据，供回调函数使用 |
| 返回值 | 成功：大于等于0  失败：小于0 |

##### lame\_file\_get\_size

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_file\_get\_size(void\* handle); |
| 功能 | 在lame\_extract\_file或lame\_extract\_file\_w回调函数中根据返回的句柄获取当前子文件大小 |
| 参数 | handle：子文件对象句柄 |
| 返回值 | 成功：大于0  失败：等于0 |

##### lame\_file\_seek

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_file\_seek(void\* handle, int32\_t offset, [LSMETHOD](#_LSMETHOD) method); |
| 功能 | 在lame\_extract\_file或lame\_extract\_file\_w回调函数中根据返回的句柄设定当前子文件的偏移 |
| 参数 | handle：子文件对象句柄 |
| 返回值 | 成功：大于等于0  失败：小于0 |

##### lame\_file\_tell

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_file\_tell(void\* handle); |
| 功能 | 在lame\_extract\_file或lame\_extract\_file\_w回调函数中根据返回的句柄获取当前子文件的偏移 |
| 参数 | handle：子文件对象句柄 |
| 返回值 | 成功：大于等于0  失败：小于0 |

##### lame\_file\_read

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | EXTERN\_C long lame\_file\_read(void\* handle, uint8\_t\* buf, uint32\_t size); |
| 功能 | 在lame\_extract\_file或lame\_extract\_file\_w回调函数中根据返回的句柄在当前子文件的偏移处读取文件内容，大小是size |
| 参数 | handle：子文件对象句柄  buf：uint8\_t类型的指针  size：读取文件的大小 |
| 返回值 | 成功：读取文件的字节数  失败：等于0 |

#### 数据结构类型说明

##### lame\_info

* 定义

typedef enum rx\_obj\_id\_t {

COBJ\_LAME, //创建引擎对象

COBJ\_CONF, //创建参数对象

COBJ\_RESULT, //创建结果对象

} rx\_obj\_id\_t;

* 说明

无

##### lame\_scan\_feedback

* 定义

typedef struct lame\_scan\_feedback\_

{

[lame\_scan\_enter\_file](#_lame_scan_enter_file) enter\_file;

[lame\_scan\_leave\_file](#_lame_scan_leave_file) leave\_file;

[lame\_scan\_alarm](#_lame_scan_alarm) alarm;

}lame\_scan\_feedback;

* 说明

无

##### lame\_extract\_feedback

* 定义

typedef struct lame\_extract\_feedback\_

{

[lame\_extract\_enter\_file](#_lame_extract_enter_file) enter\_file;

[lame\_extract\_leave\_file](#_lame_extract_leave_file) leave\_file;

}lame\_extract\_feedback;

* 说明

无

##### lame\_scan\_result

* 定义

typedef struct lame\_scan\_result

{

[rx\_mk\_t](#_rx_mk_t) mklass;

char kclass[CLASS\_NAME\_LEN];

char kclass\_desc\_a[CLASS\_NAME\_LEN];

wchar\_t kclass\_desc\_w[CLASS\_NAME\_LEN];

char engid[ENGID\_LENGTH];

char vname[VIRUS\_LENGTH];

uint32\_t vid32;

uint64\_t vid40;

char hitag[HITAG\_LENGTH];

[rx\_trc\_t](#_rx_trc_t) treat;

}lame\_scan\_result;

* 说明

无

##### LSMETHOD

* 定义

typedef enum LSMETHOD\_

{

LAME\_SEEK\_SET = 0,

LMAE\_SEEK\_CUT = 1,

LMAE\_SEEK\_TAIL =2

}LSMETHOD;

* 说明

无

##### LSCT

* 定义

typedef enum LSCT\_

{

LSCT\_CONTINUE = 0x01,

LSCT\_ABORT

}LSCT;

* 说明

无

##### LXCT

* 定义

typedef enum LXCT\_

{

LCT\_CONTINUE = 0x01,

LCT\_DO\_EXTRACT,

LCT\_ABORT,

}LXCT;

* 说明

无

##### LXLVT

* 定义

typedef enum LXLVT\_

{

LXLVT\_NORMAL = 0x01,

LXLVT\_ABORT = 0x02,

LXLVT\_ERROR = 0x03

}LXLVT;

* 说明

无

##### FCLS

* 定义

enum

{

FCLS\_PROGRAM = 0x00000010, // DOSCOM/PE//ELF/JAVA

FCLS\_DOCUMENT = 0x00000020, // PDF/DOC/DOCX/rtf

FCLS\_PACKAGE = 0x00000040, // Zip/7z/...

FCLS\_MMEDIA = 0x00000080, // Image|Moive|Audio

FCLS\_SCRIPT = 0x00000100, // All script

FCLS\_EMAIL = 0x00000200, // E-Mail

FCLS\_DISKIMG = 0x00000400, // Disk in file, VHD, VMDK

FCLS\_BOOT = 0x00000800, // Virtual Disk

};

* 说明

无

##### lame\_scan\_enter\_file

* 定义

typedef [LSCT](#_LSCT) (LAMESTDCALLTYPE \*lame\_scan\_enter\_file)(const char\* file\_name , uint32\_t depth , void\* usr\_data);

* 说明

无

##### lame\_scan\_leave\_file

* 定义

typedef void (LAMESTDCALLTYPE \*lame\_scan\_leave\_file)(const char\* file\_name , uint32\_t depth ,void\* usr\_data , LXLVT l);

* 说明

无

##### lame\_scan\_alarm

* 定义

typedef [LSCT](#_LSCT) (LAMESTDCALLTYPE \*lame\_scan\_alarm)(const char\* file\_name , [lame\_scan\_result](#_lame_scan_result)\* info,void\* usr\_data);

* 说明

无

##### lame\_extract\_enter\_file

* 定义

typedef [LXCT](#_LXCT) (LAMESTDCALLTYPE \*lame\_extract\_enter\_file)(const char\* file\_name , const char\* format , uint32\_t depth , uint64\_t FCLS ,void\* hanlde , void\* userdata);

* 说明

无

##### lame\_extract\_leave\_file

* 定义

typedef void (LAMESTDCALLTYPE \*lame\_extract\_leave\_file)(const char\* file\_name , const char\* format , uint32\_t depth , uint64\_t FCLS ,void\* hanlde , void\* userdata , [LXLVT](#_LXLVT) l);

* 说明

无

#### 代码编译执行示例

scan.c

-----------------------------------------------------------------------------------------------------------------------------------------

#include "lame.v2.h"

int32\_t ScanFile(const char\* fname)

{

lame\_info info;

if (LAME\_SUCCEEDED(lame\_get\_version(&info)))

{

printf("Engine Version: %s\n" , info.engv);

printf("Malware Version: %s\n" , info.vdbv);

}

void\* vdb = lame\_open\_vdb(0);

if (!vdb) return -1;

void\* lame = lame\_create(vdb);

if (!lame)

{

lame\_close\_vdb(vdb);

return -1;

}

lame\_param\_set(lame , "precise-format=0");

if(lame\_init(lame) < 0)

{

lame\_close\_vdb(vdb);

lame\_destroy(lame);

return -1;

}

lame\_scan\_result result;

if(lame\_scan\_file(lame , fname , &result) >= 0)

{

printf("EngineName:%s , Virus Name:%s" , result.engid , result.vname);

}

lame\_destroy(lame);

lame\_close\_vdb(vdb);

return 0;

}

Makefile

-----------------------------------------------------------------------------------------------------------------------------------------

LAME\_DIR=/root/lame/

gcc scan.c -o $( LAME\_DIR )myscan -llame -lrxavx –L$( LAME\_DIR)

运行

-----------------------------------------------------------------------------------------------------------------------------------------

LD\_LIBRARY\_PATH=/root/lame ./myscan /root/virus.bin

## DotNet接口

### 病毒库

#### 类概述

|  |  |
| --- | --- |
| 属性 | 描述 |
| OpenVdb | 打开病毒库 |
| CloseVdb | 关闭病毒库并释放资源 |

#### OpenVdb

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | Bool OpenVdb (string vdbf) |
| 功能 |  |
| 参数 |  |
| 返回值 | 成功：True  失败：False |

#### 3 . CloseVdb

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | Void CloseVdb() |
| 功能 |  |
| 参数 |  |
| 返回值 | 成功：True  失败：False |

### Lame

#### 类概述

|  |  |
| --- | --- |
| 序号 | 描述 |
| 1 | bool Load(VirusLib lib) |
| 2 | void Unload() |
| 3 | bool SetParameters(string param) |
| 4 | LameLicenceInfo GetLicense() |
| 5 | LameVesionInfo GetVersion() |
| 6 | LameScanResult ScanFile(string sFile) |
| 7 | LameScanResult ScanMem(byte[] bytes) |

#### Load

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | bool Load(VirusLib lib) |
| 功能 | 根据用户设置的引擎参数，及病毒库创建并加载引擎 |
| 参数 | 病毒库对象 |
| 返回值 | 成功：True  失败：False |

#### UnLoad

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | void UnLoad() |
| 功能 | 释放除病毒库之外的所有对象 |
| 参数 | 无 |
| 返回值 | 无 |

#### SetParameters

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | bool SetParameters(string param) |
| 功能 | 设置引擎参数，如杀毒参数kill，扫描压缩包的层数等 |
| 参数 | Parm：字符串参数，如：” kill”，” precise-format=1” |
| 返回值 | 成功：True  失败：False |

#### GetLicense

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | LameLicenceInfo GetLicense() |
| 功能 | 获取引擎的授权信息 |
| 参数 | 无 |
| 返回值 | 成功：返回授权信息对象  失败：null |

LameLicenceInfo

|  |  |
| --- | --- |
| 属性 | 描述 |
| Version | 获取SDK的版本信息，Get属性 |
| Owner | 获取SDK的拥有者信息，Get属性 |
| Date | 获取SDK的授权日期，Get属性 |
| Authm | 获取SDK的授权方式，Get属性 |

#### GetVersion

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | LameVesionInfo GetVersion() |
| 功能 | 获取引擎、病毒库信息信息 |
| 参数 | 无 |
| 返回值 | 成功：返回LameVesionInfo对象  失败：null |

LameVersionInfo

|  |  |
| --- | --- |
| 属性 | 描述 |
| EngineVersion | 获取引擎的版本信息，Get属性 |
| VirusLibVersion | 获取病毒库信息，Get属性 |

#### ScanFile

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | LameScanResult ScanFile(string sFile) |
| 功能 | 根据文件名扫描文件 |
| 参数 | 全路径文件名 |
| 返回值 | 成功：返回LameScanResult对象  失败：null |

LameScanResult

|  |  |
| --- | --- |
| 属性 | 描述 |
| VirusType | 病毒类型信息，Get属性 |
| VirusTypeName | 病毒类型名，Get属性 |
| VirusTypeDesc | 病毒类型描述，Get属性 |
| EngineID | 报毒的子引擎名称，Get属性 |
| VirusName | 病毒名，Get属性 |
| VirusID32 | 病毒ID，Get属性 |
| VirusID40 | 病毒ID，Get属性 |
| HitTag | 病毒的唯一标识，Get属性 |
| TreatResult | 病毒的处理结果，Get属性 |

#### ScanMem

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | LameScanResult ScanMem(byte[] bytes) |
| 功能 |  |
| 参数 |  |
| 返回值 | 成功：返回LameScanResult对象  失败：null |

### LameWithEvent

#### 类概述

|  |  |
| --- | --- |
| 序号 | 描述 |
| 1 | bool Load(VirusLib lib) |
| 2 | void Unload() |
| 3 | bool SetParameters(string param) |
| 4 | LameLicenceInfo GetLicense() |
| 5 | LameVesionInfo GetVersion() |
| 6 | void ScanFile(string sFile) |
| 7 | void ScanMem(byte[] bytes) |

说明：

Load、UnLoad、SetParameters、GetLicense、GetVersion同Lame类中方法一下，此处不再描述

#### ScanFile

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | void ScanFile(string sFile) |
| 功能 | 根据文件名扫描文件，扫描结果通过  LameScannerEnterFileEvent，LameScannerLeaveFileEvent，LameScannerAlarm事件通知 |
| 参数 | 全路径文件名 |
| 返回值 | 无 |

#### ScanMem

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | void ScanMem(byte[] bytes) |
| 功能 | 扫描内存文件，扫描结果通过  LameScannerEnterFileEvent，LameScannerLeaveFileEvent，LameScannerAlarm事件通知 |
| 参数 |  |
| 返回值 | 无 |

### PackageExtract

#### 类概述

|  |  |
| --- | --- |
| 序号 | 描述 |
| 1 | bool Load(VirusLib lib) |
| 2 | void Unload() |
| 3 | bool Extract(string file , string passwd) |

说明：

Load、UnLoad同Lame类中方法一下，此处不再描述

#### Extract

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | bool Extract(string file , string passwd) |
| 功能 | 根据文件名扫描文件，压缩包密码解包文件，结果通过OnExtractEnterFileEvent，OnExtractLeaveFileEvent事件通知 |
| 参数 | File：全路径文件名  Passwd：压缩包密码 |
| 返回值 | 成功：True  失败：False |

## Python接口

### 病毒库

#### 类概述

|  |  |
| --- | --- |
| 序号 | 描述 |
| 1 | OpenVdb(vdbf) |
| 2 | CloseVdb() |

#### OpenVdb

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | OpenVdb (vdbf) |
| 功能 | 打开病毒库 |
| 参数 | Vdbf：病毒库路径 |
| 返回值 | 成功：True  失败：False |

#### CloseVdb()

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | CloseVdb() |
| 功能 | 关闭病毒库信息 |
| 参数 | 无 |
| 返回值 | 无 |

### Lame

#### 类概述

|  |  |
| --- | --- |
| 方法 | 描述 |
| 1 | Load(vdb\_object) |
| 2 | Unload() |
| 3 | SetParam(param) |
| 4 | GetLicense() |
| 5 | GetVersion() |
| 6 | ScanFile(fname) |
| 7 | ScanMem(byte[] data) |

#### Load

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | Load(vdb\_object) |
| 功能 | 根据用户设置的引擎参数，及病毒库创建并加载引擎。当引擎不在使用时，必须通过Unload来释放，否则会造成内存泄露 |
| 参数 | Vdb：病毒库对象 |
| 返回值 | 成功：True  失败：False |

#### UnLoad

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | void UnLoad() |
| 功能 | 释放除病毒库之外的所有对象 |
| 参数 | 无 |
| 返回值 | 无 |

#### SetParameters

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | SetParam(param) |
| 功能 | 设置引擎参数，如杀毒参数kill，扫描压缩包的层数等 |
| 参数 | Parm：字符串参数，如：” kill”，” precise-format=1” |
| 返回值 | 成功：True，  失败：False |

#### GetLicense

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | GetLicense() |
| 功能 | 获取引擎的授权信息 |
| 参数 | 无 |
| 返回值 | 成功：返回LameLicenseInfo授权信息对象  失败：None |

LameLicenseInfo

|  |  |
| --- | --- |
| 属性 | 描述 |
| Version | 获取SDK的版本信息 |
| Owner | 获取SDK的拥有者信息 |
| Date | 获取SDK的授权日期 |
| Authm | 获取SDK的授权方式 |

#### GetVersion

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | GetVersion() |
| 功能 | 获取引擎的授权信息 |
| 参数 | 无 |
| 返回值 | 成功：返回LameVersionInfo信息对象  失败：None |

LameVersionInfo

|  |  |
| --- | --- |
| 属性 | 描述 |
| engv | 病毒类型信息 |
| vdbv | 报毒的子引擎名称 |

#### ScanFile

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | ScanFile(fname) |
| 功能 | 根据文件名扫描文件 |
| 参数 | sFile：全路径文件名 |
| 返回值 | 成功：返回LameScanResult对象  失败：None |

LameScanResult

|  |  |
| --- | --- |
| 属性 | 描述 |
| mkclass | 病毒类型信息 |
| kclass | 病毒类型名称 |
| kclass\_desc | 病毒类型名称描述 |
| kclass\_desc\_w | 病毒类型名称描述 |
| engid | 报毒的子引擎名称 |
| vname | 病毒名 |
| vid32 | 病毒ID |
| vid40 | 病毒ID |
| hitag | 病毒的唯一标识 |
| treat | 病毒的处理结果 |

#### 8. ScanMem

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | LameScanResult ScanMem(byte[] data) |
| 功能 | 扫描内存文件 |
| 参数 | Bytes：内存块 |
| 返回值 | 成功：返回LameScanResult对象  失败：None |

### LameWithFeedback

#### 类概述

|  |  |
| --- | --- |
| 序号 | 方法 |
| 1 | Load(vdb) |
| 2 | Unload() |
| 3 | SetParameters(param) |
| 4 | GetLicense() |
| 5 | GetVersion() |
| 6 | SetCallack(enter\_file , leave\_file , alram) |
| 7 | ScanFile(fname) |
| 8 | ScanMem(data) |

说明：

Load、UnLoad、SetParameters、GetLicense、GetVersion同Lame类中方法一下，此处

不再描述

#### SetCallack

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | SetCallback(enter\_file , leave\_file , alram) |
| 功能 | 设置扫描子文件时的通知函数 |
| 参数 | enter\_file：LameScanEnterFile扫描前文件信息  leave\_file：LameScannerLeaveFile扫描后文件信息  alram：LameScannerAlarm病毒信息 |
| 返回值 | 无 |

LameScanEnterFile函数原型

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | LameScanEnterFile(fname , depth , userdata) |
| 功能 | 设置扫描子文件前的通知函数 |
| 参数 | fname：当前正在扫描的文件名  depth：当前文件的深度  userdata：用户自定义数据 |
|  | 无 |

LameScannerLeaveFile函数原型

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | LameScannerLeaveFile(fname , depth , userdata , l) |
| 功能 | 设置扫描子文件后的通知函数 |
| 参数 | fname：当前正在扫描的文件名  depth：当前文件的深度  userdata：用户自定义数据  l：控制代码 |
|  | 无 |

LameScannerAlarm函数原型

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | LameScannerAlarm(fname , result , userdata) |
| 功能 | 设置子文件查到病毒的通知函数 |
| 参数 | fname：当前正在扫描的文件名  result：发现病毒LameScanResult对象，无毒None  userdata：用户自定义数据 |
|  | 无 |

#### ScanFile

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | ScanFile(sFile) |
| 功能 | 根据文件名扫描文件，扫描结果通过SetCallback设置的函数处理 |
| 参数 | 全路径文件名 |
| 返回值 | 无 |

#### ScanMem

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | void ScanMem(byte[] data) |
| 功能 | 扫描内存文件，扫描结果通过SetCallback设置的函数处理 |
| 参数 | 内存块 |
| 返回值 | 无 |

### LameExtractor

#### 类概述

|  |  |
| --- | --- |
| 序号 | 描述 |
| 1 | bool Load(vdbf) |
| 2 | void Unload() |
| 3 | Extract(fname, password) |
| 4 | SetEnterFileCallBack(cb) |
| 5 | SetLeaveFileCallBack(cb): |

说明：

Load、UnLoad同Lame类中方法一下，此处不再描述

#### ExtractFile

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | ExtractFile(fname , password) |
| 功能 | 根据文件名扫描文件，压缩包密码解文件包  lame\_extract\_enter\_file，lame\_extract\_leave\_file事件通知 |
| 参数 | fname：全路径文件名  password：压缩包密码 |
| 返回值 | 无 |

#### SetEnterFileCallBack

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | SetEnterFileCallBack(cb) |
| 功能 | 设置解文件时的通知函数 |
| 参数 | cb：lame\_extract\_enter\_file解包前文件信息 |
| 返回值 | 无 |

lame\_extract\_enter\_file函数原型

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | lame\_extract\_enter\_file(fname , fmt , depth , flag , fhandle , userdata) |
| 功能 | 设置解子文件前的通知函数 |
| 参数 | fname：当前正在扫描的文件名  fmt：当前子文件格式  depth：当前子文件的深度  flag：当前子文件类型标志  fhandle：子文件句柄  userdata：用户自定义数据 |
|  | 无 |

#### SetLeaveFileCallBack

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | SetLeaveFileCallBack(cb) |
| 功能 | 设置解文件时的通知函数 |
| 参数 | cb：lame\_extract\_leave\_file解包后文件信息 |
| 返回值 | 无 |

lame\_extract\_enter\_file函数原型

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | lame\_extract\_enter\_file(fname , fmt , depth , flag , fhandle , userdata) |
| 功能 | 设置解文件子文件后的通知函数 |
| 参数 | fname：当前正在扫描的文件名  fmt：当前子文件格式  depth：当前子文件的深度  flag：当前子文件类型标志  fhandle：子文件句柄  userdata：用户自定义数据 l：控制代码 |
|  | 无 |

## Java接口

### 病毒库

#### 类概述

|  |  |
| --- | --- |
| 属性 | 描述 |
| long OpenVdb(String vdbf) | 打开病毒库并加载 |
| void CloseVdb() | 关闭病毒库释放资源 |

#### OpenVdb

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | long OpenVdb(String vdbf) |
| 功能 | 打开病毒库并加载 |
| 参数 | vdbf病毒库绝对路径 |
| 返回值 | 成功：非0  失败：0 |

#### 3 . CloseVdb

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | void CloseVdb() |
| 功能 | 关闭病毒库并释放资源 |
| 参数 | 无 |
| 返回值 | 无 |

### Lame

#### 类概述

|  |  |
| --- | --- |
| 序号 | 描述 |
| 1 | boolean Load(long lib) |
| 2 | void Unload() |
| 3 | boolean Set(String param) |
| 4 | LicenceInfo GetLicense() |
| 5 | LameInfo GetVersion() |
| 6 | ScanResult ScanFile(String fname) |
| 7 | ScanResult ScanMem(byte [] data) |
| 8 | boolean ScanFileWithCallback(String fname, ScanFeedback cb, Object usrdata) |
| 9 | boolean ScanMemWithCallback(byte [] data, ScanFeedback cb, Object usrdata) |

#### Load

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | boolean Load(long lib) |
| 功能 | 根据用户设置的引擎参数，及病毒库创建并加载引擎 |
| 参数 | 病毒库对象句柄，可为0（加载当前位置病毒库） |
| 返回值 | 成功：true  失败：false |

#### UnLoad

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | void UnLoad() |
| 功能 | 释放所有对象 |
| 参数 | 无 |
| 返回值 | 无 |

#### Set

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | boolean Set (String param) |
| 功能 | 设置引擎参数，如杀毒参数kill，扫描压缩包的层数等 |
| 参数 | Parm：字符串参数，如：” kill”，” precise-format=1” |
| 返回值 | 成功：true  失败：false |

#### GetLicense

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | LicenceInfo GetLicense() |
| 功能 | 获取引擎的授权信息 |
| 参数 | 无 |
| 返回值 | 成功：返回授权信息对象  失败：null |

LicenceInfo

|  |  |
| --- | --- |
| 属性 | 描述 |
| Version | 获取SDK的版本信息 |
| Owner | 获取SDK的拥有者信息 |
| Date | 获取SDK的授权日期 |
| Authm | 获取SDK的授权方式 |
| Data | 获取SDK的数据信息 |

#### GetVersion

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | LameInfo GetVersion() |
| 功能 | 获取引擎、病毒库信息信息 |
| 参数 | 无 |
| 返回值 | 成功：返回LameInfo对象  失败：null |

LameInfo

|  |  |
| --- | --- |
| 属性 | 描述 |
| engine\_version | 获取引擎的版本信息 |
| virus\_db\_version | 获取病毒库信息 |

#### ScanFile

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | ScanResult ScanFile(String fname) |
| 功能 | 根据文件名扫描文件 |
| 参数 | 全路径文件名 |
| 返回值 | 成功：返回ScanResult对象  失败：null |

|  |  |
| --- | --- |
| 属性 | 描述 |
| mklass | 病毒类型信息 |
| kclass | 病毒类型名称 |
| kclass\_desc\_a | 病毒类型描述 |
| kclass\_desc\_w | 病毒类型描述 |
| engid | 报毒的子引擎名称 |
| vname | 病毒名 |
| vid32 | 病毒ID |
| hitag | 病毒的唯一标识 |
| treat | 病毒的处理结果 |

#### ScanMem

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | ScanResult ScanMem(byte [] data) |
| 功能 | 对传入的内存块进行扫描 |
| 参数 | 内存块数组 |
| 返回值 | 成功：返回ScanResult对象  失败：nil |

#### ScanFileWithCallback

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | boolean ScanFileWithCallback(String fname, ScanFeedback cb, Object usrdata) |
| 功能 | 根据文件名扫描文件，扫描结果通过cb回调函数获取 |
| 参数 | fname全路径文件名  cb 回调函数接口  usrdata 用户自定义数据，供回调函数使用 |
| 返回值 | 成功：返回true  失败：返回false |

#### ScanMemWithCallback

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | boolean ScanMemWithCallback(byte [] data, ScanFeedback cb, Object usrdata) |
| 功能 | 扫描内存文件，扫描结果通过cb回调函数获取 |
| 参数 | data内存块数组  cb 回调函数接口  usrdata 用户自定义数据，供回调函数使用 |
| 返回值 | 无 |

ScanFeedback

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | public interface ScanFeedback {  public abstract long DtEnterFile(String file\_name, long depth, Object usr\_data);  public abstract void DtLeaveFile(String file\_name, long depth, Object usr\_data, long l);  public abstract long DtAlarm(String file\_name, ScanResult sr, Object usr\_data);  } |
| 功能 | 回调函数接口 |
| 参数 |  |
| 返回值 |  |

#### ExtractFile

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | boolean ExtractFile(String fname, String passwd, ExtractFeedback cb, Object usrdata) |
| 功能 | 解压缩包文件，结果通过cb回调函数获取 |
| 参数 | fname：文件路径  passwd：压缩包密码  cb：回调函数接口  usrdata：用户自定义数据，供回调函数使用 |
| 返回值 | 成功：true  失败：false |

ExtractFeedback

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | public interface ExtractFeedback {  public abstract long ExtEnterFile(String file\_name, String format, long depth, long FCLS, long handle, Object usrdata);  public abstract void ExtLeaveFile(String file\_name, String format, long depth, long FCLS, long handle, Object usrdata, long l);  } |
| 功能 | 回调函数接口 |
| 参数 |  |
| 返回值 |  |

#### FileGetSize

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | long FileGetSize(long handle) |
| 功能 | 获取当前子文件大小 |
| 参数 | handle：当前子文件句柄 |
| 返回值 | 成功：大于0  失败：0 |

#### FileSeek

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | boolean FileSeek(long handle, long offset, long method) |
| 功能 | 设置当前子文件偏移 |
| 参数 | handle：当前子文件句柄  offset：偏移大小  method：偏移的方法 |
| 返回值 | 成功：true  失败：false |

#### FileTell

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | long FileTell(long handle) |
| 功能 | 获取当前子文件偏移位置 |
| 参数 | handle：当前子文件句柄 |
| 返回值 | 成功：大于0  失败：-1 |

#### FileRead

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | long FileRead(long handle, byte[] buf, long size) |
| 功能 | 获取当前子文件的文件内容 |
| 参数 | handle：当前子文件句柄  buf：缓存内存块  size：读取的文件大小 |
| 返回值 | 成功：大于0  失败：0 |

## go接口

### 病毒库

#### 类概述

|  |  |
| --- | --- |
| 属性 | 描述 |
| OpenVdb(vlib string) | 打开病毒库并加载 |
| CloseVdb() | 关闭病毒库释放资源 |

#### OpenVdb

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | OpenVdb(vlib string) uintptr |
| 功能 | 打开病毒库并加载 |
| 参数 | vdbf病毒库绝对路径 |
| 返回值 | 成功：非0  失败：0 |

#### 3 . CloseVdb

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | CloseVdb() |
| 功能 | 关闭病毒库并释放资源 |
| 参数 | 无 |
| 返回值 | 无 |

### Lame

#### 类概述

|  |  |
| --- | --- |
| 序号 | 描述 |
| 1 | Create(vdb uintptr) uintptr |
| 2 | Destory() |
| 3 | Init() bool |
| 4 | SetParam(param string) bool |
| 5 | GetLicenceInfo() \*LicenceInfo |
| 6 | GetVersion() \*LameVersion |
| 7 | ScanFile(path string) \*DetectResult |
| 8 | ScanMem(data [] byte, size int) \*DetectResult |
| 9 | ScanFileWithCallback(path string, cb ScanCallBack, userdata interface{}) bool |
| 10 | ScanMemWithCallback(data [] byte, size int, cb ScanCallBack, userdata interface{}) bool |
| 11 | ExtractFile(filename string, password string, cb ExtractCallBack, userdata interface{}) bool |
| 12 | Extract\_GetSize(handle uintptr) int |
| 13 | Extract\_Seek(handle uintptr, offset int, method int32) bool |
| 14 | Extract\_Tell(handle uintptr) int32 |
| 15 | Extract\_Read(handle uintptr, buf [] byte, size uint32) uint32 |

#### Create

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | Create(vdb uintptr) uintptr |
| 功能 | 根据打开的病毒库句柄创建引擎 |
| 参数 | 病毒库对象句柄 |
| 返回值 | 成功：非0  失败：0 |

#### Destory

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | Destory() |
| 功能 | 释放创建的引擎对象 |
| 参数 | 无 |
| 返回值 | 无 |

#### Init

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | Init() bool |
| 功能 | 根据创建的引擎对象初始化参数加载引擎 |
| 参数 | 无 |
| 返回值 | 成功：true  失败：false |

#### SetParam

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | SetParam(param string) bool |
| 功能 | 设置引擎参数，如杀毒参数kill，扫描压缩包的层数等 |
| 参数 | Parm：字符串参数，如：” kill”，” precise-format=1” |
| 返回值 | 成功：true  失败：false |

#### GetLicenceInfo

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | GetLicenceInfo() \*LicenceInfo |
| 功能 | 获取引擎的授权信息 |
| 参数 | 无 |
| 返回值 | 成功：返回授权信息对象  失败：nil |

LicenceInfo

|  |  |
| --- | --- |
| 属性 | 描述 |
| Version | 获取SDK的版本信息 |
| Owner | 获取SDK的拥有者信息 |
| Date | 获取SDK的授权日期 |
| Authm | 获取SDK的授权方式 |
| Data | 获取SDK的数据信息 |

#### GetVersion

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | GetVersion() \*LameVersion |
| 功能 | 获取引擎、病毒库信息信息 |
| 参数 | 无 |
| 返回值 | 成功：返回LameInfo对象  失败：nil |

LameVersion

|  |  |
| --- | --- |
| 属性 | 描述 |
| Engv | 获取引擎的版本信息 |
| Vdbv | 获取病毒库信息 |

#### ScanFile

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | ScanFile(path string) \*DetectResult |
| 功能 | 根据文件名扫描文件 |
| 参数 | 全路径文件名 |
| 返回值 | 成功：返回ScanResult对象  失败：nil |

DetectResult

|  |  |
| --- | --- |
| 属性 | 描述 |
| mklass | 病毒类型信息 |
| kclass | 病毒类型名称 |
| kclass\_desc\_a | 病毒类型描述 |
| kclass\_desc\_w | 病毒类型描述 |
| engid | 报毒的子引擎名称 |
| vname | 病毒名 |
| vid32 | 病毒ID |
| Vid40 | 病毒ID |
| hitag | 病毒的唯一标识 |
| treat | 病毒的处理结果 |

#### ScanMem

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | ScanMem(data [] byte, size int) \*DetectResult |
| 功能 | 对传入的内存块进行扫描 |
| 参数 | 内存块数组 |
| 返回值 | 成功：返回ScanResult对象  失败：nil |

#### ScanFileWithCallback

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | ScanFileWithCallback(path string, cb ScanCallBack, userdata interface{}) bool |
| 功能 | 根据文件名扫描文件，扫描结果通过cb回调函数获取 |
| 参数 | path全路径文件名  cb 回调函数接口  usrdata 用户自定义数据，供回调函数使用 |
| 返回值 | 成功：返回true  失败：返回false |

ScanCallBack

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | type scan\_enter\_file func(file\_name string, depth uint32, usr\_data interface{}) uint32  type scan\_leave\_file func(file\_name string, depth uint32, usr\_data interface{}, l uint32)  type scan\_alarm func(file\_name string, info \*DetectResult, usr\_data interface{}) uint32  type ScanCallBack struct {  EnterFile scan\_enter\_file  LeaveFile scan\_leave\_file  Alarm scan\_alarm  } |
| 功能 | 回调函数接口 |
| 参数 |  |
| 返回值 |  |

#### ScanMemWithCallback

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | ScanMemWithCallback(data [] byte, size int, cb ScanCallBack, userdata interface{}) bool |
| 功能 | 扫描内存文件，扫描结果通过cb回调函数获取 |
| 参数 | data内存块数组  size 内存块大小  cb 回调函数接口  usrdata 用户自定义数据，供回调函数使用 |
| 返回值 | 无 |

#### ExtractFile

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | ExtractFile(filename string, password string, cb ExtractCallBack, userdata interface{}) bool |
| 功能 | 解压缩包文件，结果通过cb回调函数获取 |
| 参数 | filename：文件路径  password：压缩包密码  cb：回调函数接口  userdata：用户自定义数据，供回调函数使用 |
| 返回值 | 成功：true  失败：false |

ExtractFeedback

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | type extract\_enter\_file func(file\_name string, format string, depth uint32, FCLS uint64, handle uintptr, userdata interface{}) uint32  type extract\_leave\_file func(file\_name string, format string, depth uint32, FCLS uint64, handle uintptr, userdata interface{}, l uint32)  type ExtractCallBack struct {  EnterFile extract\_enter\_file  LeaveFile extract\_leave\_file  } |
| 功能 | 回调函数接口 |
| 参数 |  |
| 返回值 |  |

#### Extract\_GetSize

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | Extract\_GetSize(handle uintptr) int |
| 功能 | 获取当前子文件大小 |
| 参数 | handle：当前子文件句柄 |
| 返回值 | 成功：大于0  失败：0 |

#### Extract\_Seek

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | Extract\_Seek(handle uintptr, offset int, method int32) bool |
| 功能 | 设置当前子文件偏移 |
| 参数 | handle：当前子文件句柄  offset：偏移大小  method：偏移的方法 |
| 返回值 | 成功：true  失败：false |

#### Extract\_Tell

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | Extract\_Tell(handle uintptr) int32 |
| 功能 | 获取当前子文件偏移位置 |
| 参数 | handle：当前子文件句柄 |
| 返回值 | 成功：大于0  失败：-1 |

#### Extract\_Read

|  |  |
| --- | --- |
| 方法 | 描述 |
| 原型 | Extract\_Read(handle uintptr, buf [] byte, size uint32) uint32 |
| 功能 | 获取当前子文件的文件内容 |
| 参数 | handle：当前子文件句柄  buf：缓存内存块  size：读取的文件大小 |
| 返回值 | 成功：大于0  失败：0 |

# 附录

## 常用参数设置

|  |  |
| --- | --- |
| 参数设置 | 描述 |
| precise-format=1/0 | 开启/关闭精确扫描（默认开启，扫描的记录会少一些） |
| max-treat-times=500 | 杀毒次数（默认500） |
| max-compound-size=-1 | 最大扫描复合文档的大小（单位byte，默认uint64（-1）） |
| max-nest-depth=50 | 压缩脱壳类文件最大扫描深度（默认50） |
| fast=0/1 | 杀毒模式下，使用能快速退出当前扫描文件（默认关闭） |