瑞星杀毒引擎SDK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 2](#_Toc509906998)

[使用说明 2](#_Toc509906999)

[安装 2](#_Toc509907000)

[授权文件生成 2](#_Toc509907001)

[适应语言 2](#_Toc509907002)

[操作系统 2](#_Toc509907003)

[注意事项 2](#_Toc509907004)

[性能参数 2](#_Toc509907005)

[查杀速度 2](#_Toc509907006)

[异步多线程查杀 2](#_Toc509907007)

[病毒库引擎升级 2](#_Toc509907008)

[升级方式 2](#_Toc509907009)

[升级内容 3](#_Toc509907010)

[手工升级办法 3](#_Toc509907011)

[升级后工作异常处理办法 3](#_Toc509907012)

[增量升级方法 3](#_Toc509907013)

[接口说明 4](#_Toc509907014)

[rx\_obj\_create 4](#_Toc509907015)

[rx\_obj\_reference 5](#_Toc509907016)

[rx\_obj\_release 5](#_Toc509907017)

[rx\_conf\_set\_json 5](#_Toc509907018)

[rx\_conf\_set\_str 5](#_Toc509907019)

[rx\_conf\_set\_int 5](#_Toc509907020)

[rx\_conf\_set\_bool 6](#_Toc509907021)

[rx\_conf\_erase 6](#_Toc509907022)

[rx\_conf\_get\_str 6](#_Toc509907023)

[rx\_conf\_get\_int 6](#_Toc509907024)

[rx\_conf\_get\_bool 7](#_Toc509907025)

[rx\_conf\_cloud\_init 7](#_Toc509907026)

[rx\_lame\_init 7](#_Toc509907027)

[rx\_lame\_scan 7](#_Toc509907028)

[rx\_lame\_scan\_memory 7](#_Toc509907029)

[rx\_sdr\_get\_result 8](#_Toc509907030)

[rx\_lame\_get\_version 8](#_Toc509907031)

[rx\_lame\_get\_virus\_library\_version 8](#_Toc509907032)

[rx\_lame\_get\_licence\_info 8](#_Toc509907033)

[数据结构类型说明 9](#_Toc509907034)

[rx\_obj\_id\_t 9](#_Toc509907035)

[rx\_scan\_opt\_t 9](#_Toc509907036)

[rx\_treat\_result\_code\_t 9](#_Toc509907037)

[rx\_mk\_t 9](#_Toc509907038)

[rx\_result\_t 10](#_Toc509907039)

[rx\_obj\_t 10](#_Toc509907040)

[rx\_sdr\_t 11](#_Toc509907041)

[rx\_sfr\_t 11](#_Toc509907042)

[rx\_ptr\_t 11](#_Toc509907043)

[rx\_cstr\_t 11](#_Toc509907044)

[rx\_str\_t 11](#_Toc509907045)

[rx\_licence\_info 11](#_Toc509907046)

[代码编译执行示例 11](#_Toc509907047)

# 概述

本软件开发包（SDK）为瑞星信息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。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

## 操作系统

Windows系列、RedHat系列、ubuntu系列、centos系列等

## 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）

# 病毒库引擎升级

## 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

## 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 67255307 Sep 13 17:34 lame.tgz

压缩包内包含有如下内容：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| homeup | librxmss.so | multscan | rxmltk.so | scanelf.so |
| lameClient | librxruntim.so | revm.so | rxoffice.so | scanexec.so |
| lameSrv | librxsig64.so | rxarch.so | rxsecns.so | scanex.so |
| libcurl.so | libup | rxauth.so | rxvision.so | scanmac.so |
| licence.json | rxcore.so | rxwinpe.so | scanpe.so | librxcoml.so |
| rxfsm.so | rxxdu0.so | scansct.so | librxffr.so | methodex.so |
| rxxman.so | scriptci.so | uroutine.so | urutils.so | vex86.so |
| liblame.so | malware.rmd | rxmail.so | test | scan |

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“./scan /rsdata/virus.bin”检查引擎工作状态

有病毒的情况：



无毒的情况：



如出现FileScan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

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

1. 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}

1. libup使用实例



1. homeup用法

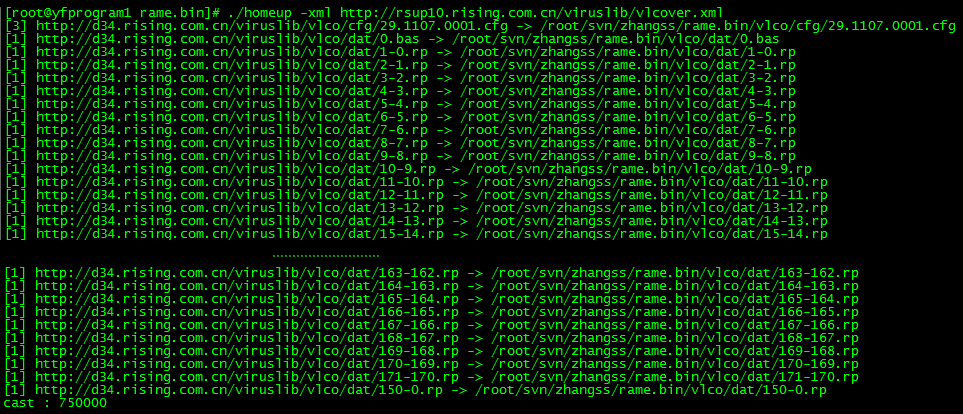
获取并升级中心系统的病毒库文件，包括bas文件、rp文件和配置文件

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

homeup -xml {xml url} {local root}

homeup -cfg {cfg url} {local root}

1. homeup使用实例



# 接口说明

扫描接口在liblame.so中，具体说明如下：

## rx\_obj\_create

* 声明

EXTERN\_C [rx\_obj\_t](#_rx_obj_t) rx\_obj\_create([rx\_obj\_id\_t](#_rx_obj_id_t) objid); // object has been referenced

* 功能说明：

根据传入的参数创建引擎对象、配置参数、扫描结果

* 参数说明：

objid：需要创建对象的类型，参见[rx\_obj\_id\_t](#_rx_obj_id_t)

* 返回值说明：

成功返回值非NULL，否则返回NULL

## rx\_obj\_reference

* 声明

EXTERN\_C void rx\_obj\_reference([rx\_obj\_t](#_rx_obj_t) obj);

* 功能说明：

引用创建的对象

* 参数说明：

obj：[rx\_obj\_create](#_rx_obj_create)创建的对象

* 返回值说明：

无

## rx\_obj\_release

* 声明

EXTERN\_C void rx\_obj\_release([rx\_obj\_t](#_rx_obj_t) obj);

* 功能说明：

obj：释放创建的对象

* 参数说明：

obj：[rx\_obj\_create](#_rx_obj_create)创建的对象

* 返回值说明：

无

## rx\_conf\_set\_json

* 声明

EXTERN\_C long rx\_conf\_set\_json([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) content);

* 功能说明：

根据输入的json串配置参数

* 参数说明：

conf：引擎参数配置对象

content：json串指针

* 返回值说明：

成功返回0，否则非0

## rx\_conf\_set\_str

* 声明

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);

* 功能说明：

配置参数项为字符串值

* 参数说明：

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

* 返回值说明：

成功返回0，否则非0

## rx\_conf\_set\_int

* 声明

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);

* 功能说明：

配置参数项为整型值

* 参数说明：

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

* 返回值说明：

成功返回0，否则非0

## rx\_conf\_set\_bool

* 声明

EXTERN\_C long rx\_conf\_set\_bool([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path, const bool val);

* 功能说明：

配置参数项为布尔值

* 参数说明：

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

* 返回值说明：

成功返回0，否则非0

## rx\_conf\_erase

* 声明

EXTERN\_C long rx\_conf\_erase([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path);

* 功能说明：

删除配置参数项

* 参数说明：

conf：引擎参数配置对象

path：参数配置项字符串指针

* 返回值说明：

成功返回0，否则非0

## rx\_conf\_get\_str

* 声明

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);

* 功能说明：

获取配置参数项字符串值

* 参数说明：

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

* 返回值说明：

成功返回0，否则非0

## rx\_conf\_get\_int

* 声明

EXTERN\_C long rx\_conf\_get\_int([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path, int64\_t \* pval);

* 功能说明：

获取配置参数项整型值

* 参数说明：

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

* 返回值说明：

成功返回0，否则非0

## rx\_conf\_get\_bool

* 声明

EXTERN\_C long rx\_conf\_get\_bool([rx\_obj\_t](#_rx_obj_t) conf, [rx\_cstr\_t](#_rx_cstr_t) path, bool \* pval);

* 功能说明：

获取配置参数项布尔值

* 参数说明：

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

* 返回值说明：

成功返回0，否则非0

## rx\_conf\_cloud\_init

* 声明

EXTERN\_C long rx\_conf\_cloud\_init([rx\_obj\_t](#_rx_obj_t) conf);

* 功能说明：

引擎云扫描功能初始化

* 参数说明：

conf：引擎配置参数对象

* 返回值说明：

成功返回0，否则非0

## rx\_lame\_init

* 声明

EXTERN\_C long rx\_lame\_init([rx\_obj\_t](#_rx_obj_t) eng, [rx\_obj\_t](#_rx_obj_t) conf);

* 功能说明：

引擎初始化

* 参数说明：

eng：引擎对象

conf：引擎配置参数对象

* 返回值说明：

成功返回0，否则非0

## rx\_lame\_scan

* 声明

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);

* 功能说明：

扫描文件并返回扫描结果

* 参数说明：

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)

* 返回值说明：

成功返回扫描结果的对象，否则NULL，参见[rx\_restult\_t](#_rx_result_t)

## rx\_lame\_scan\_memory

* 声明

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);

* 功能说明：

扫描文件内容流并返回扫描结果

* 参数说明：

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

* 返回值说明：

成功返回扫描结果的对象，否则NULL，参见[rx\_restult\_t](#_rx_result_t)

## rx\_sdr\_get\_result

* 声明

EXTERN\_C bool rx\_sdr\_get\_result([rx\_sdr\_t](#_rx_sdr_t) sdr, [rx\_result\_t](#_rx_result_t) \* result);

* 功能说明：

解析引擎扫描的返回结果

* 参数说明：

sdr：引擎扫描的结果对象

result：指向[rx\_result\_t](#_rx_result_t)的指针

* 返回值说明：

有病毒返回true，否则false

## rx\_lame\_get\_version

* 声明

EXTERN\_C bool rx\_lame\_get\_version([rx\_str\_t](#_rx_str_t) ver, int length);

* 功能说明：

返回引擎当前版本

* 参数说明：

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

* 返回值说明：

成功获取版本信息返回true，否则false

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

* 声明

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);

* 功能说明：

返回病毒库当前版本

* 参数说明：

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

* 返回值说明：

成功获取版本信息返回true，否则false

## rx\_lame\_get\_licence\_info

* 声明

EXTERN\_C bool rx\_lame\_get\_licence\_info([rx\_licence\_info](#_rx_licence_info)\* info, [rx\_cstr\_t](#_rx_cstr_t) path);

* 功能说明：

返回lame的授权文件信息

* 参数说明：

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

* 返回值说明：

成功获取版本信息返回true，否则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\_treat\_result\_code\_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);

getchar();

return 0;

}

Makefile

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

LAME\_DIR=/root/lame/

gcc scan.c -o $( LAME\_DIR )myscan -llame -lrxsig64 -lrxmss -lrxffr -lrxruntim -lrxcoml –L$( LAME\_DIR)

运行

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

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