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

2018

目录

[概述 2](#_Toc509906555)

[使用说明 2](#_Toc509906556)

[安装 2](#_Toc509906557)

[授权文件生成 2](#_Toc509906558)

[适应语言 2](#_Toc509906559)

[操作系统 2](#_Toc509906560)

[注意事项 2](#_Toc509906561)

[性能参数 2](#_Toc509906562)

[查杀速度 2](#_Toc509906563)

[异步多线程查杀 2](#_Toc509906564)

[病毒库引擎升级 2](#_Toc509906565)

[升级方式 2](#_Toc509906566)

[升级内容 3](#_Toc509906567)

[手工升级办法 3](#_Toc509906568)

[升级后工作异常处理办法 3](#_Toc509906569)

[增量升级方法 3](#_Toc509906570)

[接口说明 4](#_Toc509906571)

[lame\_open\_vdb 4](#_Toc509906572)

[lame\_close\_vdb 5](#_Toc509906573)

[lame\_create 5](#_Toc509906574)

[lame\_destroy 5](#_Toc509906575)

[lame\_param\_set 5](#_Toc509906576)

[lame\_init 6](#_Toc509906577)

[lame\_scan\_file 6](#_Toc509906578)

[lame\_scan\_mem 6](#_Toc509906579)

[lame\_get\_version 6](#_Toc509906580)

[lame\_get\_licence\_info 6](#_Toc509906581)

[数据结构类型说明 7](#_Toc509906582)

[rx\_treat\_result\_code\_t 7](#_Toc509906583)

[rx\_mk\_t 7](#_Toc509906584)

[rx\_scan\_result 8](#_Toc509906585)

[rx\_lame\_info 8](#_Toc509906586)

[rx\_licence\_info 8](#_Toc509906587)

[代码编译执行示例 9](#_Toc509906588)

# 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：

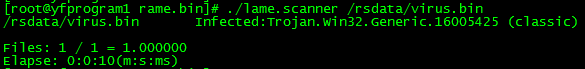
-rw-r--r--. 1 root root 67255307 Sep 13 17:34 lame.tgz

压缩包内包含有如下内容：

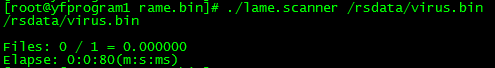
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| homeup | librxmss.so | multscan | rxmltk.so | scanelf.so |
| lameClient | librxruntim.so | revm.so | rxoffice.so | scanexec.so |
| lameSrv | librxsig64.so | rxarch.so | rxsecns.so | scanex.so |
| libcurl.so | libup | rxauth.so | rxvision.so | scanmac.so |
| 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 | lame.scanner |  |

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Files，Elapse字样统计信息的扫描结果，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

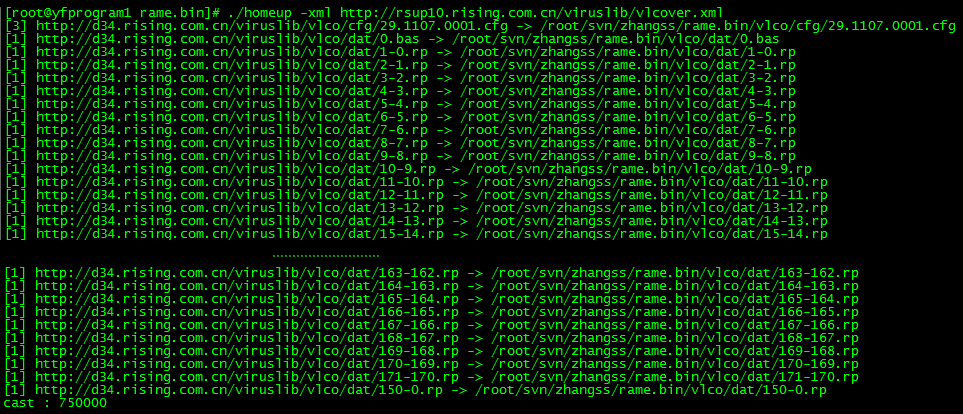
获取并升级中心系统的病毒库文件，包括bas文件、rp文件和配置文件

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：

## lame\_open\_vdb

* 声明

EXTERN\_C void\* lame\_open\_vdb(const char\* vlibf);

* 功能说明：

根据传入的路径参数打开病毒库，并返回其指针

* 参数说明：

vlibf：字符串指针，可为空（默认加载当前路径下的病毒库）

* 返回值说明：

成功返回病毒库对象指针，否则返回NULL

## lame\_close\_vdb

* 声明

EXTERN\_C void lame\_close\_vdb(void\* vdb);

* 功能说明：

关闭病毒库，释放病毒库资源

* 参数说明：

vdb：[lame\_open\_vdb](#_lame_open_vdb)创建的对象

* 返回值说明：

无

## lame\_create

* 声明

EXTERN\_C void\* lame\_create(void\* vdb);

* 功能说明：

创建引擎对象

* 参数说明：

vdb：由[lame\_open\_vdb](#_rx_obj_create)打开的病毒库对象指针

* 返回值说明：

成功返回引擎对象指针，否则返回NULL

## lame\_destroy

* 声明

EXTERN\_C void lame\_destroy(void\* lame);

* 功能说明：

释放引擎资源

* 参数说明：

lame：由[lame\_create](#_lame_create)创建的引擎对象指针

* 返回值说明：

无

## lame\_param\_set

* 声明

EXTERN\_C long lame\_param\_set(void\* lame , const char\* param);

* 功能说明：

根据传入的字符串配置引擎参数

* 参数说明：

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

* 返回值说明：

成功返回值大于等于0，否则小于0

## lame\_init

* 声明

EXTERN\_C long lame\_init(void\* lame);

* 功能说明：

引擎初始化

* 参数说明：

lame：由[lame\_create](#_lame_create)创建的引擎对象指针

* 返回值说明：

成功返回值大于等于0，否则小于0

## lame\_scan\_file

* 声明

EXTERN\_C long lame\_scan\_file(void\* lame , const char\* fname , [rx\_scan\_result](#_rx_result_t)\* pResult) ;

* 功能说明：

扫描文件并返回扫描结果

* 参数说明：

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)

* 返回值说明：

成功返回值大于等于0，否则小于0

## lame\_scan\_mem

* 声明

EXTERN\_C long lame\_scan\_mem(void\* lame , uint8\_t\* data , uint32\_t size , [rx\_scan\_result](#_rx_result_t)\* pResult);

* 功能说明：

扫描文件内容流并返回扫描结果

* 参数说明：

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)

* 返回值说明：

成功返回值大于等于0，否则小于0

## lame\_get\_version

* 声明

EXTERN\_C long lame\_get\_version(rx\_lame\_info\* info);

* 功能说明：

返回引擎及病毒库的版本

* 参数说明：

info：指向版本信息的指针，参见[rx\_lame\_info](#_rx_lame_info)

* 返回值说明：

成功返回值大于等于0，否则小于0

## lame\_get\_licence\_info

* 声明

EXTERN\_C long lame\_get\_licence\_info([rx\_licence\_info](#_rx_licence_info)\* info);

* 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

* 返回值说明：

成功返回值大于等于0，否则小于0

# 数据结构类型说明

## rx\_treat\_result\_code\_t

* 定义

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

TREAT\_FAIL\_FIX = -1,

TREAT\_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\_scan\_result

* 定义

typedef struct rx\_scan\_result

{

rx\_mk\_t mklass;

char engid[32];

char vname[256];

uint32\_t vid32;

uint64\_t vid40;

char hitag[32];

rx\_trc\_t treat;

} rx\_scan\_result;

* 说明

无

## rx\_lame\_info

* 定义

typedef struct rx\_lame\_info\_

{

const char\* engine\_version\_ref;

const char\* virus\_db\_version\_ref;

} rx\_lame\_info;

* 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.v2.h"

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

{

void \*vdb, \*lame;

char\* filename;

rx\_scan\_result result;

vdb = lame\_open\_vdb(0);

if (!vdb) return -1;

lame = lame\_create(vdb);

if (!lame) return -1;

if (0 > lame\_param\_set(lame , "precise-format=0")) return -1;

if (0 > lame\_init(lame)) return -1;

filename = argv[1];

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

{

printf("%s Infected:%s (%s)", filename, result.vname, result.engid);

}

if (vdb) lame\_close\_vdb(vdb);

if (lame) lame\_destroy(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