

linux crypto API 学习

原创 ★临★ 最后发布于2019-04-24 14:50:37 阅读数 891 ☆ 收藏

linux 内核里实现了crypto模块，其用法介绍的很少：

参考：linux-3.18.21/Documentation/crypto/api-intro.txt

Here' s an example of how to use the API:

```
#include <linux/crypto.h>
#include <linux/err.h>
#include <linux/scatterlist.h>

struct scatterlist sg[2];
char result[128];
struct crypto_hash *tfm;
struct hash_desc desc;
tfm = crypto_alloc_hash( "md5" , 0, CRYPTO_ALG_ASYNC);
if (IS_ERR(tfm))
fail();
/* ... set up the scatterlists ... */
desc.tfm = tfm;
desc.flags = 0;
if (crypto_hash_digest(&desc, sg, 2, result))
fail();

crypto_free_hash(tfm);
Many real examples are available in the regression test module (tcrypt.c).
```

这提到 **Many real examples are available in the regression test module (tcrypt.c).**

这里就来看一下 linux-3.18.21/crypto/tcrypt.c

tcrypt.c

```
1 | tcrypt_test("cbc(aes)");
2 | alg_test(alg, alg, 0, 0);
3 | i = alg_find_test(alg);
4 | alg_test_descs[i].test(alg_test_descs + i, driver, type, mask);
5 | //alg_test_descs 是一个全局结构体数组，部分数据如下：
6 | {
7 |     .alg = "cbc(aes)",
8 |     .test = alg_test_skcipher, //测试函数
9 |     .fips_allowed = 1,
10 |     .suite = {
11 |         .cipher = {
12 |             .enc = {
13 |                 .vecs = aes_cbc_enc_tv_template, //测试数据
14 |                 .count = AES_CBC_ENC_TEST_VECTORS
15 |             },
16 |             .dec = {
17 |                 .vecs = aes_cbc_dec_tv_template,
18 |                 .count = AES_CBC_DEC_TEST_VECTORS
19 |             }
20 |         }
21 |     }
22 | static struct cipher_testvec aes_cbc_enc_tv_template[] = {
23 |     /* From RFC 3602 */
24 |     .key = "\x06\xa9\x21\x40\x36\xb8\xa1\x5b"
25 |         "\x51\x2e\x03\xd5\x34\x12\x00\x06",
26 |     .klen = 16,
27 |     .iv = "\x3d\xaf\xba\x42\x9d\x9e\xb4\x30"
28 |         "\xb4\x22\xda\x80\x2c\x9f\xac\x41",
29 |     .input = "Single block msg",
```



举报

```

30     .ilen = 16,
31     .result = "\xe3\x53\x77\x9c\x10\x79\xae\xb8"
32         "\x27\x08\x94\x2d\xbe\x77\x18\x1a",
33     .rlen = 16,
34 },
35 ...
36 }
37 static struct cipher_testvec aes_cbc_dec_tv_template[] = {
38     /* From RFC 3602 */
39     .key = "\x06\xa9\x21\x40\x36\xb8\xa1\x5b"
40         "\x51\x2e\x03\xd5\x34\x12\x00\x06",
41     .klen = 16,
42     .iv = "\x3d\xaf\xba\x42\x9d\x9e\xb4\x30"
43         "\xb4\x22\xda\x80\x2c\x9f\xac\x41",
44     .input = "\xe3\x53\x77\x9c\x10\x79\xae\xb8"
45         "\x27\x08\x94\x2d\xbe\x77\x18\x1a",
46     .ilen = 16,
47     .result = "Single block msg",
48     .rlen = 16,
49 },

```



继续跟踪 alg_test_skcipher

```

1 static int alg_test_skcipher(const struct alg_test_desc *desc, const char *driver, u32 type, u32 mask)
2 {
3     struct crypto_ablkcipher *tfm;
4     tfm = crypto_alloc_ablkcipher(driver, type, mask);
5     test_skcipher(tfm, ENCRYPT, desc->suite.cipher.enc.vecs, desc->suite.cipher.enc.count);
6     ret = __test_skcipher(tfm, enc, template, tcount, false, 0);
7     init_completion(&result.completion);
8     req = ablkcipher_request_alloc(tfm, GFP_KERNEL);
9     ablkcipher_request_set_callback(req, CRYPTO_TFM_REQ_MAY_BACKLOG,
10         tcrypt_complete, &result);
11     for (i = 0; i < tcount; i++) {
12         memcpy(iv, template[i].iv, MAX_IVLEN);
13         j++;
14         ret = -EINVAL;
15         data = xbuf[0];
16         data += align_offset;
17         memcpy(data, template[i].input, template[i].ilen);
18
19         crypto_ablkcipher_clear_flags(tfm, ~0);
20
21         ret = crypto_ablkcipher_setkey(tfm, template[i].key,
22             template[i].klen);
23         sg_init_one(&sg[0], data, template[i].ilen);
24         if (diff_dst) {
25             data = xoutbuf[0];
26             data += align_offset;
27             sg_init_one(&sgout[0], data, template[i].ilen);
28         }
29         ablkcipher_request_set_crypt(req, sg, (diff_dst) ? sgout : sg,
30             template[i].ilen, iv);
31
32         ret = enc ? crypto_ablkcipher_encrypt(req) :
33             crypto_ablkcipher_decrypt(req);
34
35         switch (ret) {
36             case 0:
37                 break;
38             case -EINPROGRESS:
39             case -EBUSY:
40                 ret = wait_for_completion_interruptible(
41                     &result.completion);
42                 if (!ret && !((ret = result.err))) {
43                     reinit_completion(&result.completion);
44                     break;
45                 }
46             /* fall through */
47             default:
48                 pr_err("alg: skcipher%: %s failed on test %d for %s: ret=%d\n",

```



```

48     d, e, j, algo, -ret);
49     goto out;
50 }
51
52 q = data;
53 if (memcmp(q, template[i].result, template[i].rlen)) {
54     pr_err("alg: skcipher%: Test %d failed on %s for %s\n",
55           d, j, e, algo);
56     hexdump(q, template[i].rlen);
57     ret = -EINVAL;
58     goto out;
59 }
60 }

```

自己写一个测试代码:

```

1  #include <crypto/hash.h>
2  #include <linux/err.h>
3  #include <linux/module.h>
4  #include <linux/scatterlist.h>
5  #include <linux/slab.h>
6  #include <linux/string.h>
7  #include <crypto/rng.h>
8  #include <crypto/drbg.h>
9  #include <linux/init.h>
10 #include <linux/gfp.h>
11 #include <linux/scatterlist.h>
12 #include <linux/moduleparam.h>
13 #include <linux/jiffies.h>
14 #include <linux/timex.h>
15 #include <linux/interrupt.h>
16
17 static char *alg = NULL;
18
19 struct tcrypt_result {
20     struct completion completion;
21     int err;
22 };
23
24 static void hexdump(unsigned char *buf, unsigned int len)
25 {
26     print_hex_dump(KERN_CONT, "", DUMP_PREFIX_OFFSET,
27                   16, 1,
28                   buf, len, false);
29 }
30
31 static void tcrypt_complete(struct crypto_async_request *req, int err)
32 {
33     struct tcrypt_result *res = req->data;
34
35     if (err == -EINPROGRESS)
36         return;
37
38     res->err = err;
39     complete(&res->completion);
40 }
41 /*
42 static struct cipher_testvec aes_enc_tv_template[] = {
43     {
44         .key = "\x00\x01\x02\x03\x04\x05\x06\x07"
45              "\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f",
46         .klen = 16,
47         .input = "\x00\x11\x22\x33\x44\x55\x66\x77"
48              "\x88\x99\xaa\xbb\xcc\xdd\xee\xff",
49         .ilen = 16,
50         .result = "\x69\xc4\xe0\xd8\x6a\x7b\x04\x30"
51              "\xd8\xcd\xbf\x80\x70\xb4\xc5\x5a",
52         .rlen = 16,
53     },
54 }

```



举报

```

55 */
56 int test(void)
57 {
58     char *key = "\x00\x01\x02\x03\x04\x05\x06\x07"
59               "\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f";
60     int keylen = 16;
61     char *str_input = "\x00\x11\x22\x33\x44\x55\x66\x77"
62                      "\x88\x99\xaa\xbb\xcc\xdd\xee\xff";
63     char *str_iv = "\x00\x11\x22\x33\x44\x55\x66\x77"
64                  "\x88\x99\xaa\xbb\xcc\xdd\xee\xff";
65     int inputlen = 16;
66
67     char iv[32];
68
69     struct scatterlist sg[2];
70     struct scatterlist sg2[2];
71     char *input;
72     char *output;
73     char *denc;
74     struct ablkcipher_request *req;
75     struct crypto_ablkcipher *tfm;
76     struct tcrypt_result result;
77     int ret;
78
79     input = (void *)__get_free_page(GFP_KERNEL);
80
81     //memcpy( iv, "test iv", strlen("test iv"));
82     memset(iv,0,sizeof(iv));
83
84     memcpy(input,str_input,inputlen);
85
86     output = (void *)__get_free_page(GFP_KERNEL);
87     denc = (void *)__get_free_page(GFP_KERNEL);
88
89     char *driver = "ecb(aes)";
90     if(alg)
91     {
92         driver = alg;
93     }
94     printk("alg : %s\n",driver);
95
96     if(strcmp(driver,"ecb(aes)") // != "ecb(aes)"
97     {
98         memcpy(iv,str_iv,16);
99     }
100
101     //1.加密
102     tfm = crypto_alloc_ablkcipher(driver, 0, 0);
103     if (IS_ERR(tfm)) {
104         printk(KERN_ERR "alg: skcipher: Failed to load transform for "
105               "%s: %ld\n", driver, PTR_ERR(tfm));
106         return PTR_ERR(tfm);
107     }
108
109     init_completion(&result.completion);
110     req = ablkcipher_request_alloc(tfm, GFP_KERNEL);
111     if(!req)
112     {
113         printk("error req = null\n");
114         crypto_free_ablkcipher(tfm);
115         return -1;
116     }
117     ablkcipher_request_set_callback(req, CRYPTO_TFM_REQ_MAY_BACKLOG,
118                                   tcrypt_complete, &result);
119
120     crypto_ablkcipher_clear_flags(tfm, ~0);
121     //set key
122     crypto_ablkcipher_setkey(tfm, key, keylen );
123     //sg init
124     sg_init_one(&sg[0], input, inputlen);
125     sg_init_one(&sg[1], output, inputlen);

```



举报

```

126 //set iv
127 ablkcipher_request_set_crypt(req, &sg[0], &sg[1], inputlen, iv);
128 //开始加密
129 ret = crypto_ablkcipher_encrypt(req);
130 if(ret == -EINPROGRESS || ret == -EBUSY)
131 {
132     ret = wait_for_completion_interruptible(
133         &result.completion);
134     if (!ret && !((ret = result.err)))
135     {
136         reinit_completion(&result.completion);
137     }
138 }
139 else if(ret == 0)
140 {
141     printk("succ\n");
142 }
143 else
144 {
145     printk("error ret = %d\n", ret);
146     goto _err;
147 }
148
149 hexdump(output, 16);
150
151 //2.解密
152 sg_init_one(&sg2[0], output, inputlen);
153 sg_init_one(&sg2[1], denc, inputlen);
154 crypto_ablkcipher_clear_flags(tfm, ~0);
155
156 if(strcmp(driver, "ecb(aes)") != "ecb(aes)")
157 {
158     memcpy(iv, str_iv, 16);
159 }
160
161 ablkcipher_request_set_crypt(req, &sg2[0], &sg2[1], inputlen, iv);
162
163 ret = crypto_ablkcipher_decrypt(req);
164 if(ret == -EINPROGRESS || ret == -EBUSY)
165 {
166     ret = wait_for_completion_interruptible(
167         &result.completion);
168     if (!ret && !((ret = result.err)))
169     {
170         reinit_completion(&result.completion);
171     }
172 }
173 else if(ret == 0)
174 {
175     printk("succ\n");
176     //printk("%s\n", denc);
177 }
178 else
179 {
180     printk("error ret = %d\n", ret);
181     goto _err;
182 }
183 hexdump(denc, 16);
184 _err:
185 ablkcipher_request_free(req);
186 crypto_free_ablkcipher(tfm);
187 return -1; //恒定返回-1，免得insmod后每次需要先rmmod
188 }
189
190 static int __init test_init(void)
191 {
192     test();
193 }
194
195 static void __exit test_exit(void)
196 {

```



举报

```
197 |
198 | }
199 |
200 | module_init(test_init);
201 | module_exit(test_exit);
202 |
203 | module_param(alg, charp, 0);
204 |
205 | MODULE_LICENSE("GPL");
206 | MODULE_AUTHOR("lql@lql.com");
207 |
208 |
```

测试

```
1 | root@100:/lib/modules/3.18.21# insmod /tmp/test_kernal_crypto.ko alg="cbc(aes)"
2 | [ 9332.350000] alg : cbc(aes)
3 | [ 9332.350000] succ
4 | [ 9332.360000] 00000000: c6 a1 3b 37 87 8f 5b 82 6f 4f 81 62 a1 c8 d8 79
5 | [ 9332.370000] succ
6 | [ 9332.370000] retire_capture_urb: 11 callbacks suppressed
7 | [ 9332.390000] 00000000: 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
8 | failed to insert /tmp/test_kernal_crypto.ko
9 | root@100:/lib/modules/3.18.21# insmod /tmp/test_kernal_crypto.ko alg="ctr(aes)"
10 | [ 9335.680000] alg : ctr(aes)
11 | [ 9335.680000] succ
12 | [ 9335.690000] 00000000: 69 d5 c2 eb 2e 62 47 50 54 1d 3b bc 69 2b a5
13 | [ 9335.700000] succ
14 | [ 9335.700000] 00000000: 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
15 | failed to insert /tmp/test_kernal_crypto.ko
16 | root@100:/lib/modules/3.18.21# insmod /tmp/test_kernal_crypto.ko
17 | [ 9340.790000] alg : ecb(aes)
18 | [ 9340.790000] succ
19 | [ 9340.800000] 00000000: 69 c4 e0 d8 6a 7b 04 30 d8 cd b7 80 70 b4 c5 5a
20 | [ 9340.810000] succ
21 | [ 9340.810000] retire_capture_urb: 20 callbacks suppressed
22 | [ 9340.820000] 00000000: 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
23 | failed to insert /tmp/test_kernal_crypto.ko
24 | root@G100:/lib/modules/3.18.21#
```

总结：

要学习 linux crypto API ，可以直接参考 linux-3.18.21/crypto/tcrypt.c

👍 点赞 ☆ 收藏 📄 分享 ...



★临★

发布了64 篇原创文章 · 获赞 20 · 访问量 7万+

私信



原来大家都是在这里领取的免费虚拟云主机!再也不浪费钱了

万网云虚拟主机免费版



想对作者说点什么

使用Microsoft **CryptoAPI**进行加密、解密、签名及验证

2019独角兽企业重金招聘Python工程师标准>>>

Linux内核**crypto**子系统学习笔记

crypto notev0.1 2017/12/19 Sherlock init v0.2 2017/3/25 Sherlock add sync/asyns code analysis in ai...

Windows **Crypto** API

在一个老外的博客(忘记网址了...)上发现的，他用C#声明了些windows crypto API，包括常量，结构和函数。其中注...

阅

博文 来自: weixin_3372

阅

博文 scarecrow_

阅

博文 来自: weixin_3067



SM2第二十一篇：OpenSSL中关于RSA_new和RSA_free的内存泄漏（CRYPTO_cleanup_all_ex_data）

阅读

OpenSSL中关于RSA_new和RSA_free的内存泄漏（CRYPTO_cleanup_all_ex_data）

博文 来自： 判兵马俑的博



软考哪个含金量比较高

软考哪个含金量比较高

Cryptoapi中文手册

压缩包包含两篇文档内容分别为微软CryptoAPI的中文说明文档和微软CryptoAPI详细解释（陈锡铭已译Crypt类请再译Cert...

LINUX磁盘加密之CRYPTO

http://www.2cto.com/Article/201309/243900.html近分析rc.sysinit启动脚本，正遇磁盘加密之部分，经一番细研...

博文 来自： 武溪嵌人

学习CRYPTO第三天

1,CertOpenSystemStore打开系统最常用的证书存储区域。假如需要满足复杂的需要，请看CertOpenStore HCERTS...

博文 来自： agely's blog

CryptoAPI简介（一）

一、有关加密API的国际标准和规范Ø Generic Security Services API (GSS-API) Ø Intel/OpenGroup CDSAØ RSA PK...

博文 来自： Lasewang

Linux内核中使用crypto进行sha1方法

在编写Linux驱动的时候常常需要对内核空间的某些数据进行hash计算，而在编写内核模块的时候很多用户空间的方...

博文 来自： Rain的博客

linux sysfs 学习笔记 - 重拾梦想,继续前行 - CSDN博客

Openssl中的Libcrypto API_运维_明潮的BLOG-CSDN博客



原来大家都是在这里领取的免费虚拟云主机!再也不浪费钱了

万网云虚拟主机免费版

Microsoft CryptoAPI加密技术

Microsoft CryptoAPI加密技术在这个信息爆炸的时代，我们不得不对信息的安全提高警惕。加密作为保障数据信息...

博文 来自： wenyu826的


Linux应用开发自学之路_运维_码到成功-CSDN博客


LINUX磁盘加密之CRYPTO_运维_武溪嵌人-CSDN博客


CryptoAPI的几个用处

1. 加密和解密 可以用对称加密，即用Session Key。先用一个字符串做密码，然后对他进行hash运算，通过hash结...

博文 来自： weixin_337

 **weixin_33721344**
4501篇文章
[关注](#) 排名:千里之外

 **sherlock-wang**
160篇文章
[关注](#) 排名:千里之外

 **weixin_30675247**
4395篇文章
[关注](#) 排名:千里之外

 **qq_30866297**
116篇文章
[关注](#) 排名:8000+

Linux(学习)_运维_xxxsxxxxx的博客-CSDN博客

什么是Linux怎么学习?_运维_陈小仙的博客-CSDN博客

怎么使用CryptoAPI?

如题

Crypto API (Linux)

origin: https://en.wikipedia.org/wiki/Crypto_API_(Linux)Crypto API is a cryptography framework in th...

博文 来自： azhouren的

asterisk16 编译安装问题集锦_运维_重拾梦想,继续前行-CSDN博客



阅读

硬件加密框架ocf cryptodev-linux介绍

origin: <http://blog.chinaunix.net/uid-12076195-id-3423536.html>locf-linux ocf 是 “OpenBSD Cryptogra...

博文 来自 · azhouren的



CSP开发基础--CayptAPI函数库介绍

基本加密函数基本加密函数为开发加密应用程序提供了足够灵活的空间。所有CSP的通讯都是通过这些函数。一个CS·

博文 来 uhuiyi的专栏

Linux和Windows下：Python Crypto模块安装方式区别

一、Linux环境下: from Crypto.Signature import PKCS1_v1_5如果导包报错: ImportError: No module named 'C·

博文 : ☆ weixin_4175

Crypto++在Linux中使用及算法封装

<https://blog.csdn.net/tgbtgb/article/details/52495589><https://blog.csdn.net/liang19890820/article/det...>

博文 来 oln频道

1

使用CryptoAPI进行DES加密，如何导入自己的KEY

由于加密需要使用函数 `BOOL WINAPI CryptEncrypt(__in HCRYPTKEY hKey, __in HCRYPTHASH hHash, __in BOOL Fina...`

CryptoDev for Linux

origin: [http://www.logix.cz/michal/devel/cryptodev/CryptoDev for LinuxDevice /dev/crypto](http://www.logix.cz/michal/devel/cryptodev/CryptoDev%20for%20LinuxDevice%20dev/crypto) (aka Crypto...)

博文 来自: yazhouren的·

Cryptopp密码库在Linux下的安装与使用

首先，必须要看的是主要的参考资料是已有的：<http://www.cryptopp.com/wiki/Linux>编辑一个简单地程序测试…

博文 来自: [vingstar的专栏](#)

/dev/crypto for Linux

origin: <http://lwn.net/Articles/99175/>From: Michal Ludvig To: CryptoAPI List Subject: [PATCH] /dev/c...

博文 来自: yazhouren的·

CryptoAPI加解密签名验证

利用CryptoAPI进行文件的加解密以及签名和验证服务，文件包括附录ppt

Java	C语言	Python	C++	C#	Visual Basic .NET	JavaScript	PHP	SQL	Go语言	R语言	Assembly language	Swift	Rub
MATLAB	PL/SQL	Perl	Visual Basic	Objective-C	Delphi/Object Pascal	Unity3D							

©2019 CSDN 皮肤主题:精致技术 设计师:CSDN官方博客



★临★

[TA的个人主页>](#)

原创	粉丝	获赞	评论	访问
64	28	20	29	7万+

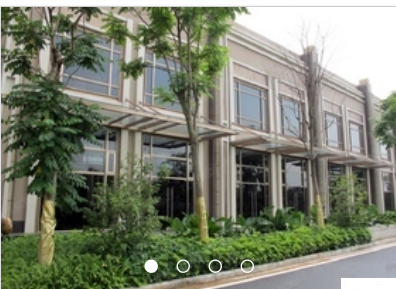
等级: **博客 4** 周排名: **1万+**

积分: 1478 总排名: 5万+

勋章:     

关注

私信



西山壹号院

最新文章

thermal 代码分析

cpufreq 代码分析



举报

【深度】韦东山：一文看看尽linux对中断处理的前世今生

【长文】说说UBOOT的几个核心问题

Linux动态频率调节系统CPUFreq之三：governor

分类专栏

	u-boot	2篇
	OpenWrt	22篇
	Linux	32篇
	cpufreq	5篇
	thermal	1篇

展开

归档

2020年3月	8篇
2019年12月	1篇
2019年11月	2篇
2019年10月	2篇
2019年8月	4篇
2019年7月	9篇
2019年6月	5篇
2019年5月	3篇

展开

最新评论

- cfns.gperf:101:1:...
qq_45438493：谢了！

cfns.gperf:101:1:...
qq_45438493：okk

cfns.gperf:101:1:...
agave7：[reply]qq_45438493[/reply]ln -s 指向路径 自定义名程

cfns.gperf:101:1:...
qq_45438493：请问用ls显示了这些代码之后该用什么命令来修改那些“-6”呢？

NanoPi M1 移植 Open...
agave7：[reply]yiexu[/reply]可以私信给我，我会尽量给出一些想法。 从你这个分区信息来看， ...

一个SDK
一套极简API

视频
全互

每月1万分钟免费 立即试

👍

🔗

💬

☆

📱

<

>

赏

🎧

举报

工作时间 8:30-22:00

[关于我们](#) [招聘](#) [广告服务](#) [网站地图](#)

京ICP备19004658号 [经营性网站备案信息](#)

 [公安备案号 11010502030143](#)

©1999-2020 北京创新乐知网络技术有限公司

[网络110报警服务](#)

[北京互联网违法和不良信息举报中心](#)

[中国互联网举报中心](#) [家长监护](#)

[版权与免责声明](#) [版权申诉](#)



举报