第26卷 第6期
2011年12月

郑州轻工业学院学报(自然科学版)
JOURNAL OF ZHENGZHOU UNIVERSITY OF LIGHT INDUSTRY(Natural Science)

Vol.26 No.6
Dec. 2011

文章编号:1004-1478(2011)06-0042-04

Lua脚本文件的加密解密研究

姜波, 张智斌, 李国, 姚文伟

(昆明理工大学 信息工程与自动化学院, 云南 昆明 650500)

摘要:针对开源Lua脚本没有提供任何安全机制的问题,采用AES算法和Sha256算法,将其加密和解密函数封装在动态链接库中,通过调用动态链接库中封装的入口函数,实现了更安全的Lua脚本.将Lua调用动态链接库的加密速度与其他加密方法进行比较,结果表明本文提供的方法,其加密的速度更快.

关键词:Lua脚本;AES算法;Sha256算法;动态链接库;加密;解密

中图分类号:TP309.7 **文献标志码**:A

Research of Lua scripting encrypt and decrypt

JIANG Bo, ZHANG Zhi-bin, LI Guo, YAO Wen-wei

(*Faculty of Infor. Eng. and Auto., Kunming Univ. of Sci. and Tech., Kunming* 650500, *China*)

Abstract: Aiming at the problem that open Lua scripting has no security mechanism, the encrypt and decrypt functions are loaded into dynamic linking library(DLL) using ASE and Sha256 algorithms, the import function within DLL is called in the Lua scripting. It was indicated by testing that the encrypted Lua scripting is more secure than the original one. In addition, the encrypt speed of Lua calling DLL is compared with that of other method, and the results showed that the former is greatly improved in the speed of encrypt.

Key words: Lua scripting; AES algorithm; Sha256 algorithm; dynamic linking library(DLL); encrypt; decrypt

0 引言

脚本语言或扩建的语言,又叫动态语言,是一种编程语言控制软件应用程序.脚本通常以文本形式保存,只在被调用时才进行解释或编译.目前运行最快的脚本语言是Lua,很多程序采用Lua作为嵌入式语言,以此来实现程序的可配置性和可扩充性,例如魔兽世界、博德之门等.Lua已经被广泛地应用于游戏脚本和工业控制中.

但开源的Lua没有提供任何安全机制,即任意的文本编辑器(如记事本、写字板、SciTE等)都可以得到Lua脚本中的数据.对于那些有安全需求的系统而言,这是Lua的致命缺点.Matthias Hilbig用纯Lua语言实现了AES算法,但加密和解密的速度不高,且只能对少量的数据进行加密与解密,而典型的AES算法能够实现每秒几百兆的加密和解密速度.故本文拟使用能够与Lua紧密结合的C语言实现AES算法,通过调用动态链接库,更好地实现Lua脚本文件的加密与解密,以增加数据的安全性,提高加密与解密的速度.

收稿日期:2011-04-25

作者简介:姜波(1985—),男,安徽省淮北市人,昆明理工大学硕士研究生,主要研究方向为计算机网络应用、数据库等.

通信作者:张智斌(1965—),男,云南省昆明市人,昆明理工大学副教授,主要研究方向为计算机图络、图形图像处理.

万方数据

1 脚本语言 Lua

脚本语言是一种在执行期间才检验数据类型的程序设计语言,通过脚本能够动态改变主程序的逻辑行为,而不需要重新编译整个主程序.

Lua 语言是动态的[1],可以与其他语言相互调用,也很容易与传统的 C/C ++ 整合. 尽管 Lua 由标准 C 编写而成,但 Lua 提供了 C 所不善长的机制,如高级语言、动态结构、简洁、易于测试和调试等. 正因为如此,Lua 具有良好的安全保证、自动内存管理、简便的字符串处理及其他动态数据的改变等功能.

Lua 语言的主要特点有[2-3]:变量没有类型,只有值才有类型,运行期间可以赋给变量任何类型的值; 函数是一阶值,并支持 closure 概念;采用词法定界;提供协程机制;系统自带内存回收功能;表是唯一的数据结构,相当于关联数组,任何类型的值都可以作为表的键来检索数据;提供较丰富的反射功能;可以为每个表设定元表和元方法,从而改变表操作的原始语义,即所谓的动态元机制.

2 Lua 安全机制设计

2.1 AES 算法原理

AES 算法是一种可以用于保护电子数据的加密算法[4],其加密速度快、解密时间长、资源消耗低. 其构成是一个迭代的、对称密钥分组的密码,可以使用128 b,192 b 和 256 b 密钥,并且用 128 b(16 B)分组加密和解密数据. AES 算法基于排列和置换运算,排列是对数据重新进行安排,置换是将一个数据单元替换为另一个数据单元.

AES 加密和解密算法使用了一个由种子密钥字节数组生成的密钥调度表,可以从最初的种子密钥中生成多重轮密钥. 以一个密钥扩展算法生成一个密钥调度并以某种方式进行替代和置换,在这种方式中,加密和解密算法极其相似. 所以 AES 解密算法背后的基本原则很简单:解密一个加密块,也就是以反向顺序还原每个加密操作.

AES 算法的 4 种加密模式分别为 ECB 模式、CBC 模式、CFB 模式和 OFB 模式[5].

本文采用 AES 算法的 CBC 模式,且使用 Sha256 算法生成 256 b 密钥对明文进行加密和解密. CBC 模式适用于传输较长的报文,而 Sha256 算法生成的密钥足够长.

2.2 Lua_AES 库中加密与解密设计

Lua 脚本按照每 16 个字节块进行加密与解密,过程如下:

1)加密函数 Lua_AES_Crypt(Lua_State * L). 调用 Lua C API 中 LuaL_checklstring 函数可以获得用户输入的明文和密钥,利用 Sha256 算法进行重新散列求值得到加密的密钥和初始向量 IV,最后调用 AES_Cipher 函数加密 Lua 脚本文件,并将加密密文通过 Lua_pushlstring 函数压入堆栈中.

2)解密函数 Lua_AES_ Decrypt(Lua_State * L). AES 算法中解密是加密的逆过程. 同样通过调用 LuaL_checklstring 函数来分别获得密文和密钥,利用 Sha256 算法进行散列求值得到解密的密钥和初始向量 IV,最后调用函数 AES_InvCipher 函数解密 Lua 脚本文件,并将解密的明文通过 Lua_pushlstring 函数压入堆栈中.

图 1 给出了利用 AES 算法和 Sha256 算法实现 Lua 脚本文件加密与解密的代码层次结构图.

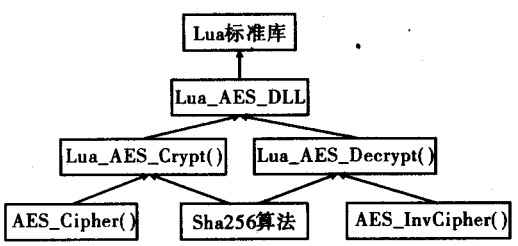

图 1　Lua 脚本文件加密与解密的代码层次结构

2.3 Lua 脚本调用动态库

C API 是 C 代码与 Lua 进行交互的函数集,由读写 Lua 全局变量的函数、调用 Lua 函数的函数、运行 Lua 代码片段的函数、注册 C 函数后可以在 Lua 中被调用的函数等部分组成[6].

本文采用从 C 代码调用 Lua 的 C API 函数,将 C 库注册到 Lua 的环境中,然后在 Lua 环境中调用动态库方式. 这种方式使得 Lua 的加密与解密库具有良好的扩展性. 如果需要添加新的功能,只需要在原有的 C 库中添加 1 个函数或者 1 个 C 或 Lua 的库就可以了. 这个库仅包含 1 个导出函数,它实现的功能就是根据输入的参数来调用 Lua_AES 相应的加密函数与解密函数. 要实现这个功能,先要定义一个结构体(这里命名为 LuaL_Reg),以实现将一个字符串注册到对应的函数中去的功能. LuaL_Reg 具体结构如下[7]:

```
typedef struct LuaL_Reg {
    const char * name;
```

万方数据

```
    Lua_CFunction func;
} LuaL_Reg;
```

这个导出函数的名字必须是以 Luaopen_开头，然后加上函数名. 库的名字必须跟这个函数的名字相同. 本文中需要导出的函数是 Lua_AES_DLL()，这个函数必须写为：

```
__declspec(dllexport) int Luaopen_Lua_AES_DLL(Lua_State * L)
{
    LuaL_openlib(L, "aes", Lua_aeslib, 0);
    return 1;
}
```

这个库的名字也应该是 Lua_AES_DLL. dll. 编译好这个库之后保存，然后在需要加密或解密的 Lua 脚本中，用 package. loadlib 命令获取函数

```
local func = assert(package.loadlib("Lua_AES_DLL.dll", "luaopen_aes"))
```

第 1 个参数是库文件的完整路径，第 2 个参数是用于打开库的函数的名字. 该命令可以获取 DLL 中抛出的函数地址并判断地址是否有效. 但必须再执行 func 函数才能真正加载库，再输入 func()才真正打开 Lua 的动态库. 执行 func 时做的工作主要是建立一个表，表名就是 LuaL_Reg 的第 2 个参数. Lua_AES_DLL. dll 中包含加密函数 Lua_AES_Crypt 和解密函数 Lua_AES_Decrypt，并且注册到结构体数组 Lua_aeslib 中对应名字也是 Lua_AES_Crypt 和 Lua_AES_Decrypt. 那么在 Lua 中，就可以像使用 Lua 的标准库函数一样使用这 2 个函数了：

```
aes.Lua_AES_Crypt(filename,key)
aes.Lua_AES_Decrypt(filename,key)
```

通过以上的准备工作，Lua 动态库在 Visual Studio 2008 或 Cygwin 环境下都能够成功进行编译.

3 加密功能测试

对生成的 Lua_AES_DLL 进行功能测试，测试内容包括：1)用 SciTE 打开加密前的 Lua 脚本文件；2)用 SciTE 打开加密后的 Lua 脚本文件；3)用 SciTE 打开解密后的 Lua 脚本文件.

3.1 加密功能测试

编写一个调用加密函数的 Lua 文件 Lua_encrypt. lua. 程序主要代码片段如下：

```
local file = assert(io.open(arg[1], "r"));
local text = file:read("*all");
local cipher = aes.Lua_AES_Crypt(arg[2], text);
io.write(cipher);
```

其中，aes. Lua_AES_Crypt(arg[2], text)是调用动态库中的加密函数，arg[1] 存放 Lua_encrypt. lua，arg[2]得到从命令行中输入的用户需要加密的 Lua 脚本文件，text 中存储的是加密 Lua 脚本文件的密钥. 图 2 是用 SciTE 打开加密前的 Lua 脚本文件的显示结果.

图 2 用 SciTE 打开加密前的 Lua 脚本文件

图 3 是用 SciTE 打开加密的 Lua 脚本文件的显示结果. 可见 Lua 脚本文件的内容已经变为乱码，达到了保密 Lua 脚本的目的.

图 3 用 SciTE 打开加密后的 Lua 脚本文件

3.2 解密功能测试

编写一个调用解密函数的 Lua 脚本文件 Lua_decrypt. lua. 程序主要代码片段如下：

```
local file = assert(io.open(arg[1], "r"));
local text = file:read("*all");
local cipher = aes.Lua_AES_Decrypt(arg[2], text);
io.write(cipher);
```

其中，aes. Lua_AES_Decrypt(arg[2], text)是调用动态库中的解密函数，arg[1] 存放 Lua_decrypt. lua，arg[2]得到从命令行中输入的用户需要解密的 Lua 脚本文件，text 中存储的是解密 Lua 脚本文件的密钥.

图 4 是用 SciTE 打开输入正确密码之后的解密 Lua 文件的显示结果. 可见被加密的 Lua 文件经过解密之后，又恢复到了原来的 Lua 脚本文件，到达了

万方数据

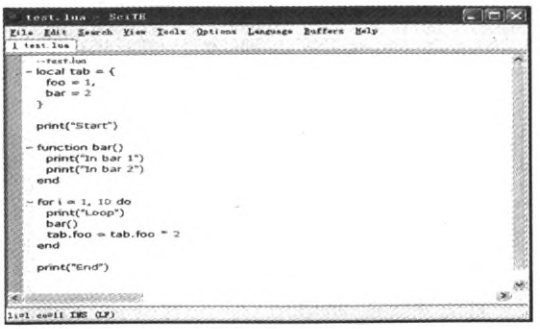

图4　用 SciTE 打开解密后的 Lua 脚本文件

解密 Lua 脚本文件的目的.

通过比较调用 Matthias Hilbig 的 aeslua 和 Lua_AES_DLL 动态库对 Lua 脚本文件加密的速度可以看出[8],C 语言实现的 AES 算法比 Lua 语言实现的 AES 算法加密速度更快,更适合加密大量的数据. 表1 给出了比较结果.

表1　AES 算法不同实现的加密速度比较

语言名称	加密数据量/KB	消耗时间/s	平均速率/(KB·s^{-1})
aeslua	156.25	2.266	68.954
Lua_AES_DLL	156.25	0.109	1 433.486

4　结语

本文通过使用能够与 Lua 紧密结合的 C 语言实现 AES 算法,调用动态链接库,更好地实现了 Lua 脚本文件的加密与解密. 动态链接库不仅封装了共享资源和代码,节约了内存,而且在系统定制和升级时只需替换 DLL 中相应的函数即可,这正符合 Lua 的可配置和可扩充性的设计理念. AES 加密算法速度快及内存消耗低等特点也是 Lua 语言所要求的,通过它为 Lua 脚本文件的安全性提供了足够的保障. 因此本文提供的方法可以实现更快、更安全的 Lua 脚本.

参考文献:

[1] Roberto Ierusalimschy. Lua 程序设计[M]. 2 版. 周惟迪,译. 北京:电子工业出版社,2008.

[2] Kurt Jung, Aaron Brown. Beginning Lua Programming[M]. Indianapolis: Wiley Publishing, 2007.

[3] Ierusalimschy R, Figueiredo L H de, Celes W. Reference Manual for Lua 5.1[EB/OL]. (2011-09-28)[2011-10-08]. http://www.lua.org/manual/5.1/.

[4] 谷大武. 高级加密标准(AES)算法——Rijndael 的设计[M]. 北京:清华大学出版社,2003.

[5] 张景文. 基于 AES 加密算法的数据文件安全策略与实现[J]. 电脑与信息术,2010,18(4):25.

[6] 邓正阳,陈和平,苏鹏. 动态脚本语言 Lua 与 C++交互方法的研究与实现[J]. 计算机系统应用,2010,19(5):198.

[7] 石琦. Lua 在自动测试中的应用[J]. 自动化技术与应用,2009,28(12):31.

[8] Garry. Leamenu[EB/OL](2010-07-15)[2011-08-25]. http://code:google.com/P/Luast oned/.

万方数据