

SSL 协议安全缺陷分析

李 玮, 侯整风

(合肥工业大学 计算机与信息学院, 安徽 合肥 230009)

摘 要:近年来,随着对 Internet 上传输数据保密性的需要,安全套接字层(SSL)被广泛地使用。SSL 协议基于公开密钥技术,提供了一种保护客户端/服务器通信安全的机制。但是,SSL 协议仍存在一些安全缺陷,对使用 SSL 协议的通信带来安全隐患。文中简要介绍了 SSL 3.0 协议的内容,重点讨论了 SSL 3.0 协议的安全缺陷并针对缺陷提出了相应的改进方法,为协议的实现提供了参考。

关键词:安全套接字层; 安全; 缺陷

中图分类号: TP393.08

文献标识码: A

文章编号: 1673-629X(2006)12-0224-03

Analysis of the Vulnerabilities of SSL Protocol

LI Wei, HOU Zheng-feng

(Dept. of Computer, Hefei University of Technology, Hefei 230009, China)

Abstract: The use of secure sockets layer (SSL) on the Internet has grown significantly in recent years as more applications use the Internet to deliver data which require traffic to be confidential. The SSL protocol which is based on the public key technique is intended to provide a mechanism for Internet client/server communications security. However, SSL protocol still has some vulnerabilities, which bring some dangers to Internet communications. Introduce the content of the SSL 3.0 protocol and discuss chiefly several security vulnerabilities and attacks on the protocol. According to these vulnerabilities, also present some improvement suggestions, which give some references to the implementation of the protocol.

Key words: SSL; security; vulnerability

0 引言

随着电子商务的广泛应用,在开放的网络上进行安全可靠的数据通信已经成为安全交易的重要内容。其中安全套接字层(SSL)协议是目前使用最广泛的用于 Web 浏览器的安全协议。该协议向基于 TCP/IP 的客户端/服务器应用程序提供了客户机和服务器的鉴别、数据完整性以及机密性等安全措施,为网络上的客户机/服务器提供了一个实际的、应用层的、面向连接的安全通信机制。

1996 年, Netscape 公司发布了 SSL 3.0, 该版本增加了对除了 RSA 算法以外的其他算法的支持和一些新的安全特性,并且修改了前一个版本中存在的安全缺陷。

1 SSL 3.0 结构

SSL 3.0 是一种分层协议,它是由记录层以及记录层上承载的不同消息类型组成,而记录层又由某种可靠的传输协议 TCP 承载。下面简要分析一下协议的结构^[1]。

1.1 记录协议

记录协议位于 SSL 协议的底层,用于定义传输数据

的格式、加密/解密、压缩/解压缩、MAC 计算等操作。记录由记录头和记录数据组成。记录头包括的信息有:记录头的长度、记录数据的长度、记录数据中是否有填充数据。其中填充数据是在使用块加密算法时填充实际数据使其长度恰好是块的整数倍。记录数据由三部分组成:消息认证码、实际数据和填充数据。

1.2 握手协议

握手协议处于记录协议之上,它产生会话的压缩、MAC、加密的计算参数。当客户端发起 SSL 会话后,通过握手协议,双方协商随后通信中使用的协议版本、密码算法,彼此互相鉴别验证,使用公开密钥密码技术协商产生共享密钥。典型的握手过程如下:

(1) 客户端发送第一条消息 client_hello, 其中包含了客户端所推荐的加密参数,包括它准备使用的加密算法。此外,还包括一个在密钥产生过程中使用的随机值。

(2) 服务器以三条消息进行响应:首先发送选择加密与压缩算法的 server_hello, 这条消息包含一个从服务器过来的随机值。然后,服务器发送 certificate 消息,其中包含服务器的公用密钥。服务器可选地发送 certificate_request 消息,要求客户端的证书。最后,服务器发送表示握手阶段不再有任何消息的 server_hello_done。

(3) 如果服务器要求客户端证书,客户端应发送一条

收稿日期:2006-03-22

作者简介:李 玮(1983-),男,安徽合肥人,硕士研究生,研究方向为网络安全;侯整风,教授,研究方向为计算机网络与信息安全。

certificate 消息给服务器。然后客户端发送一条 client-key-exchange 消息,其中包含了一个随机产生的用服务器的 RSA 密钥加密的 pre-master-secret。如果客户端证书具有签名功能的话,客户端还应发送 certificate-verify 消息。这条消息后面跟着一条指示客户端在此之后发送的所有消息都将使用刚刚商定的密码进行加密的 change-cipher-spec 消息。finished 消息包含了对整个连接过程的校验,这样服务器就能够判断要使用的加密算法是否是安全商定的。

(4) 一旦服务器接受到了客户端的 finished 消息,它就会发送自己的 change-cipher-spec 和 finished 消息,于是连接就准备好进行应用数据的传输了。

1.3 警示消息

警示主要用于报告各种类型的错误。大多数警示用于报告握手中出现的问题,但是也有一些指示在对记录试图进行解密或认证时发送的错误。

1.4 ChangeCipherSpec 消息

change-cipher-spec 表示记录加密及认证的改变。一旦握手商定了一组新的密钥,就发送 change-cipher-spec 来指示此刻将启用新的密钥。

2 SSL 3.0 安全缺陷和改进

2.1 通信业务流分析攻击

SSL 协议提供了通信消息的机密性和完整性,在选择恰当的密码算法的基础上,所有在网络中传输的消息都被加密,并且使用 MAC 对消息的完整性进行保护。因此,对通信信道的窃听是不能获得机密信息的。于是攻击者就可能采用另一种被动攻击手段——通信业务流分析^[2]。通信业务流分析目的通过分析 IP 包未经加密的字段和未受保护的属性,恢复受保护会话的机密信息。如通过检验未经加密的 IP 源和目的地址或观察网络的流量状况,可确定会话双方的身份、正在使用何种服务、甚至猜测出商务或个人关系的信息。

在用户使用 HTTP 协议进行 WWW 浏览的时候,攻击者使用通信业务流的分析方法,对浏览器和 WWW 服务器之间的 SSL 通信进行攻击,可以发现非常有效的攻击方法。通过检查密文信息的长度等综合的业务流分析,可以得到双方的 IP 地址、端口号、URL 请求的长度、Web 页面的长度等。结合现在高效的 Web 搜索引擎技术,以上信息使攻击者发现用户调用的 Web 页面。这种攻击的关键是得到密文的长度,而由于在 SSL 中,无论是分组密码算法还是流密码算法,密文的长度都是近似准确的。

SSL 协议对这种攻击的脆弱是由于其所处的网络协议的层次决定的。要想从根本上防止这种攻击必须从网络层甚至数据链路层着手,分层解决。用户也应该注意在通信过程中,尽量避免机密信息的暴露。

2.2 ChangeCipherSpec 消息的丢弃攻击

在 SSL 3.0 协议中,一个小的漏洞是在 finished 消息

中没有对 change-cipher-spec 消息的认证^[3]。这一缺陷将会导致一种潜在的攻击方法:丢弃 change-cipher-spec 消息攻击。

在正常的情况下,双方的通信流程如下:

C→S: [change-cipher-spec]

C→S: [finished:]{a}_k

S→C: [change-cipher-spec]

S→C: [finished:]{a}_k

C→S: {m}_k

其中 {*} 指对数据进行加密保护;m 代表密钥交换完成后传输的数据,a 代表消息认证码,是对所有握手消息进行 MAC 计算的结果。在接受到 change-cipher-spec 消息之前,当前的 cipher-suite 一般不加密和不作 MAC 保护,而此时待决的 cipher-suite 中包含了商定的加密算法等参数。通信实体在接受到 change-cipher-spec 消息之后,就把待决的 cipher-suite 复制成当前的 cipher-suite,记录层开始对通信数据进行加密和完整性保护。

假设一种特殊情形:商定的 cipher-suite 只对消息进行认证,不作加密保护。在这种情形下攻击者就可以发起中间人攻击。双方的通信流程如下:

C→M: [change-cipher-spec]

C→M: [finished:]{a}_k

M→S: [finished:]a

S→M: [change-cipher-spec]

S→M: [finished:]{a}_k

M→C: [finished:]a

C→M: {m}_k

M→S: m

攻击者 M 截取并删除了 change-cipher-spec 消息,那么通信双方将不再更新当前的 cipher-suite,也就不再对发送的数据作 MAC 认证了。由于商定的 cipher-suite 不使用加密,则很容易从 {m}_k 得到数据 m。这样,协议对通信数据失去了认证能力,攻击者在通信双方不知道的情况下获得了对数据进行任意篡改的能力。

如果通信双方商定的 cipher-suite 使用加密,该攻击就不容易实现了。但是,如果系统使用的是弱的密码算法,例如当密钥长度为 40 比特的 DES 算法时,那么在现有的计算能力下进行密钥的穷举攻击还是能够成功的。

避免上述攻击的方法是将 change-cipher-spec 加入到 finished 消息的消息认证计算中。该安全缺陷也可以在协议实现中使用某种手段避免而不需要修改协议的基本框架。强调在通信实体发送 finished 消息之前必然接受到 change-cipher-spec 消息,否则必然引起协议的致命错误。但是,使用这种方法改进协议的安全缺陷需要依赖协议实现者的谨慎。

2.3 密钥交换算法欺骗攻击

首先描述一下 SSL 3.0 中 server-key-exchange 消息的数据结构^[4]。

```

struct{
    select(KeyExchangeAlgorithm){
        case diffie_hellman:
            ServerDHParams params;
            Signature signed_params;
        case rsa:
            ServerRSAParams params;
            Signature signed_params;
    }
} ServerKeyExchange;
struct{
    opaque RSA_modulus;
    opaque RSA_exponent;
} ServerRSAParams;
struct{
    opaque DH_p;
    opaque DH_g;
    opaque DH_Ys;
} ServerDHParams;
enum{rsa, diffie_hellman}
KeyExchangeAlgorithm;

```

服务器使用 server_key_exchange 消息来发送用服务器私钥签名的临时公开参数,客户端使用这些公开参数和服务器交换密钥,获得共享的 pre_master_secret。协议规定可以使用多种密钥交换算法,例如 RSA 算法和 Diffie Hellman 算法。由于服务器对公开参数的签名内容没有包含 KeyExchangeAlgorithm 域,因此给攻击者提供了可乘之机^[3]。攻击者使用 cipher_suite 回环攻击使服务器使用临时 DH 密钥交换,而客户端使用临时 RSA 密钥交换。这样,服务器的 DH 素数模 p 和生成因子 g 被客户端理解为临时 RSA 的模 p 和指数 g 。客户端使用伪造的参数加密 pre_master_secret。攻击者截获用 RSA 加密的值 $kg \bmod p$,因为 p 是素数,所以容易恢复出 pre_master_secret 的 PKCS 编码 k 。这样 pre_master_secret 就泄露给攻击者了。以后所有的握手消息都可以被伪造,包括 finished 消息。此后,攻击者可以在 SSL 连接上解密传输过程中所有机密应用数据和伪造任何数据。下面描述了这种攻击的通信流:

```

[client_hello:]
C → M: SSL_RSA...
M → S: SSL_DHE_RSA...
[server_hello:]
S → M: SSL_DHE_RSA...
M → C: SSL_RSA...
[server_key_exchange:]
S → M: {p, g, y}_k, diffie_hellman
M → C: {p, g, y}_k, rsa
[client_key_exchange:]
C → M: k^g mod p

```

$$M \rightarrow S: g^x \bmod p$$

其中客户端的 pre_master_secret 值是 k ,而服务器端通过计算 $g^{xy} \bmod p$ 得到 pre_master_secret,这里 x 是攻击者 M 选择的。

该安全缺陷也可以通过协议实现者的特殊处理加以避免。协议实现者需要在客户端接受到 server_key_exchange 消息时检查公开参数域的长度,这样就能够区分所使用的密钥交换算法,从而避免这种攻击。

2.4 密钥交换中的重放攻击

密钥交换算法中在 server_key_exchange 消息中绑定了公开参数和服务器和客户端的随机值的签名,但是在匿名的密钥交换算法中没有绑定签名^[3]。随机值用来阻止在新握手过程中重放前一次的 server_key_exchange 消息。下面描述了 Signature 消息的数据结构^[4]。

```

select(SignatureAlgorithm){
    case anonymous: struct{};
    case rsa:
        digitally_signed struct{
            opaque md5_hash[16];
            opaque sha_hash[20];
        };
    case dsa:
        digitally_signed struct{
            opaque sha_hash[20];
        };
} Signature;
md5_hash = MD5(client_random + server_random + ServerParams);

```

如果攻击者知道服务器使用了匿名的密钥交换,则攻击者能够重放服务器的 server_key_exchange。客户端在匿名的密钥交换中容易被欺骗。为避免这种攻击,服务器应选择对公开参数和随机值的签名算法。

2.5 密码分析方法攻击

在加密应用数据时,数据长度可以很大,如传输多媒体信息。在 SSL 协议中,基于同一密钥传输大量数据是不安全的,它提供大量的密文信息和其他附加信息,攻击者可以利用差分分析方法和线性分析方法破解密码,获得密钥^[5]。改进的出发点是定义加密的粒度,即同一密钥能加密的最大长度的明文数据,当长度超过粒度时,密钥必须更新,粒度大小应在握手中协商。下面是对 client_hello 消息的扩展:

```

struct{
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suite;
    CompressionMethod compression_method;
    UInt8 key_refresh;
} client_hello;

```

(下转第 229 页)

判断,实现了系统信息的安全性。

2 两种方法的比较

(1)两种方法都实现了用户角色权限的验证,保证了系统信息的安全性。

(2)第一种方法没有将模式和视图分开,随着系统的增大,将会导致系统更新的复杂化,以及系统的可扩展性较差。

(3)第二种方法将逻辑和视图分开。通过对 Action 的覆写,实际上实现了对 JSP 页面的定义,将单个的 JSP 页面映射称为一个权限,从而通过对应的权限实现了对相应页面的访问。这样提高了系统的可扩展性。对于新增的 JSP 页面只要新增对应的 Action 以及系统权限,即可保证系统权限验证的完整性。

(4)第二种方法通过禁止对“*.jsp”页面的直接请求,隐藏了系统的文件信息,即隐藏了系统的文件结构,无形中也提高了系统安全性。

3 结论

在 B/S 结构中,系统的权限验证比 C/S 中更显重要,C/S 结构由于具有特定的客户端,因此用户的权限验证可

以通过客户端实现或通过客户端+服务器检测实现,而在 B/S 结构中,用户通过浏览器进行系统信息的访问,如果没有一个全面的权限验证,那么“非法用户”很可能通过浏览器访问到 B/S 结构系统中非授权功能。因此 B/S 结构中用户角色的权限验证将是保证系统信息安全性的一个核心部分。文中构建的基于 Struts 框架的 Web 系统从各方面对用户提交的请求进行验证,防止了用户提交不合法的参数而获取非法信息的可能性,并隐藏了系统的文件结构,最大程度上保证了系统信息的安全性。

参考文献:

- [1] 张桂元,贾燕枫. Struts 开发入门与项目实践[M]. 北京:人民邮电出版社,2005:2-8.
- [2] 暴志刚,胡艳军,顾新建. 基于 Web 的系统权限管理实现方法[J]. 计算机工程,2006,32(1):169-170.
- [3] 张祖平,王 磊. 基于多种模式的权限控制技术研究[J]. 计算机工程,2006,32(1):177-178.
- [4] 周 杰. 对 Struts 应用开发框架的研究和改进[J]. 计算机工程,2004,30(增刊):144-145.
- [5] 孙卫琴. 基于 MVC 的 JavaWeb 设计与开发[M]. 北京:电子工业出版社,2004:5-10.

(上接第 219 页)

程序的编写,有这方面编程经验者可参考 $\mu\text{C}/\text{GUI}$ 用户手册 API 部分。

3 总结

以上详细论述了 S3C44B0X 微处理器的 GUI 硬件设计及 LCD 模块的驱动机理,在 $\mu\text{C}/\text{GUI}$ 的基础上,着重介绍了 GUI 工作机理,详细论述了基于 ARM 的 GUI 的移植方法。其中很多部分仅是提出了概要的描述,实际工作中还有很多细节工作需要完善。

(上接第 226 页)

这里添加了 key_refresh 字段,表示加密的粒度为 $2^{\text{key_refresh}}$ 个字节。协商了加密的粒度之后,现在对原来的密钥导出函数进行修改。密钥分组的计算过程如下:

$\text{key_block} = \text{PRF}(\text{master_secret}, \text{"key expansion"}, \text{server_random} + \text{client_random} + \text{Seq_Num})$

其中 Seq_Num 取值依次为 $0, 2^{\text{key_refresh}}, 2 \times 2^{\text{key_refresh}}, 3 \times 2^{\text{key_refresh}}$ 。协议实现中记录已传输的字节数,当传输的数据达到 $2^{\text{key_refresh}}$ 个字节的时候,SSL 连接两端重新计算密钥分组。

3 结束语

通过对 SSL 协议本身安全性的分析和几种针对 SSL 协议攻击手段的讨论,发现虽然上述攻击手段可以通过协议实现中进行特殊处理而避免,但是从安全协议设计的角

参考文献:

- [1] 王田苗. 嵌入式系统设计与实例开发[M]. 北京:清华大学出版社,2002:123-126.
- [2] Samsung 公司. S3C44B0X data sheet. Samsung 公司技术资料[M]. 韩国:Samsung,2004.
- [3] Micrium 公司. $\mu\text{C}/\text{GUI}$ user manual. Micrium 公司用户手册[M]. [s.l.]:Micrium,2002:325-347.
- [4] 陈冰峰,姜 卓,王剑钢. 基于 ARM7 S3C44B0X 嵌入式系统 GUI 设计[J]. 仪器仪表用户,2004(4):30-31.
- [5] 杜春雷. ARM 体系结构与编程[M]. 北京:清华大学出版社,2001:377-379.

度看,协议自身应当避免安全隐患,而不要依赖于协议实现者的谨慎。

参考文献:

- [1] Frier A, Karlton P, Kocher P. The SSL 3.0 Protocol[EB/OL]. Netscape Communications Corp. 1996-11. <http://wp.netscape.com/eng/ssl3/ssl-toc.html>.
- [2] 戴英侠. SSL 协议的安全缺陷与改进[J]. 中国科学院研究生院学报,2000,17(1):86-92.
- [3] Wagner D, Schneier B. Analyse of the SSL 3.0 Protocol[C]// Proceedings of the 2nd Usenix workshop on electronic commerce. [s.l.]:USENIX Press, 1996:29-40.
- [4] Rescorla E. SSL 与 TLS[M]. 崔 凯译. 北京:中国电力出版社,2002.
- [5] 孙红林. 传输层安全协议的安全性分析及改进[J]. 软件学报,2003,14(3):518-523.