HTTPS/TLS 协议设计和实现中的安全缺陷综述

韦俊琳[1], 段海新[1], 万 涛[2]

[1]清华大学, 网络科学与网络空间研究院 北京 中国 100084
[2]华为公司渥太华研究中心 渥太华 加拿大

摘要 SSL/TLS 协议是目前广泛使用的 HTTPS 的核心, 实现端到端通信的认证、保密性和完整性保护, 也被大量应用到非 web 应用的其他协议(如 SMTP)。因为 SSL/TLS 如此重要, 它的安全问题也引起了研究者们的兴趣, 近几年对于 SSL/TLS 协议的研究非常火热。本文总结了近几年四大安全顶级学术会议(Oakland, CCS, USENIX Secuity 和 NDSS)发表的相关论文, 分析该协议设计的设计问题、实现缺陷以及证书方面的相关研究, 希望对 SSL/TLS 协议的改进和其他协议的安全性设计有参考价值。

关键词 SSL, TLS, 网络安全, 证书
中图法分类号 TP309.2 DOI 号 10.19363/j.cnki.cn10-1380/tn.2018.03.01

A Survey of Security Deficiencies in Design and Implementation of HTTPS/TLS

WEI Junlin, DUAN Haixin, WAN Tao

[1] Institute of Network Science and Cyberspace, Tsinghua University, Beijing 100084, China
[2] Huwai Ottawa Research Center, 303 Terry Fox Drive, Ottawa, Ontario K2K 3J1, Canada

Abstract SSL/TLS is the fundamental component of HTTPS, which has been widely adopted in both web applications and other protocols like SMTP. Because the protocol is so critical to most of current web applications, the security issues of SSL/TLS attract so many attentions from scholars all over the world. In this paper, we surveyed related research papers published in the BIG4 top security conferences(Oakland, CCS, USENIX Security and NDSS), and systematically analyzed the security problems in the design and implement phases of SSL/TLS and certificate related concerns. We hope that this survey will increase the security of later version of SSL/TLS and design of other security protocols as well.

Key words SSL, TLS, network security, certificate

1 引言

在互联网设计之初, 其主要目标是增强通信能力, 没有过多考虑安全问题。因此, 现在互联网的核心通信协议 TCP、UDP 和 IP 本质上是不安全的。为了解决数据在 TCP 层的安全传输问题, Netscape 公司在 1994 年提出了 Secure Socket Layer (SSL)协议, 又称套接字安全协议。由于发布的 SSL 2 没有和 Netscape 之外的安全专家商讨, 设计得不够全面, 协议中存在着严重的漏洞 [1]。紧接着, 在 1995 年, Netscape 发布了 SSL 3, 修补了 SSL 2 协议上的很多漏洞。解决了 SSL 2 协议中存在的重放攻击, 消息认证等问题, 并在协议中增添了 ChangeCipherSpec 等新内容。SSL 3 发布以后, 得到了业界的高度重视。之后 IETF 成立 Transport Layer Security(TLS) 工作组, 基于 SSL 3 设计了 TLS, 分别于 1999 年、2006 年和 2008 年发布了 TLS 1.0 [2]、TLS 1.1 [3]和 TLS 1.2 [4], 逐步修补了协议在设计和实现中先后发现的漏洞, 并在 2016 年着手计划出台具有更好完整性的 TLS 1.3[5]。

目前, TLS 协议已经作为一种工业标准大量应用于电子商务等安全应用。如 HTTPS 就是使用 TLS 协议进行加密传输 HTTP, 旨在为 HTTP 客户端和 Web 服务器之间创建一条安全的连接, 来实现端到端的认证并保证数据传输的保密性和完整性。当客户端和服务器双方都同意使用 TLS 协议时, 他们会通过握手来建立安全连接。图 1 是一个完整的 TLS 握手过程。

通讯作者: 韦俊琳, 硕士, Email:weijl16@mails.tsinghua.edu.cn。
本课题得到国家自然科学基金(No.61472215, No. 61636204)资助。
收稿日期: 2018-2-19; 修改日期: 2018-3-5; 定稿日期: 2018-3-6

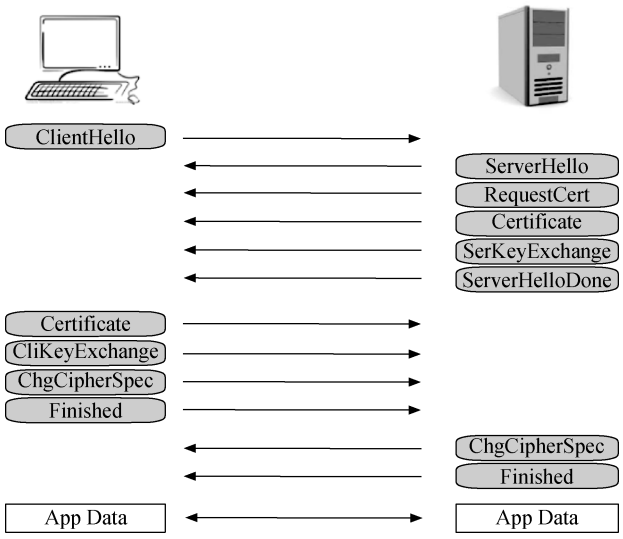

图 1 TLS 完整握手过程

Figure 1 TLS Handshake Process

随着 TLS 的广泛应用, 关于 TLS 安全的研究也越来越多、越来越深入。自 2013 年以来, 在安全领域的四个顶级会议(ISOC NDSS, IEEE Symposium on Security and Privacy, USENIX Security, ACM CCS)中, 关于 TLS 方向有近 50 篇研究论文。在这些对 TLS 的安全研究中, 发现了许多可以被利用的漏洞。我们可以把基于 TLS 的问题大致分为以下几类:

1) TLS 协议设计不足。例如早期设计基于 CBC 模式的一系列加密方式, 给现在带来很多的问题, 形成了一系列的漏洞。还有 RC4 之类的弱加密算法, 也带来严重的影响。

2) TLS 协议实现不足。由于很多 TLS 的开发人员, 没有严格按照 TLS 协议的标准规范来实现协议, 或者有的程序本身实现上有问题, 导致了 TLS 协议在实现上存在不少的漏洞。

3) TLS 所使用的数字证书的相关问题。证书管理和验证的复杂性, 也间接导致了一系列针对证书的相关攻击, 其中伪造证书, 过期证书的清除工作就较难解决。

我们将从一些经典的攻击入手对近年来 TLS 的部分研究展开介绍。第二节主要介绍与协议设计不足相关的漏洞, 第三节主要介绍协议实现不足相关的漏洞, 第四节主要介绍 TLS 证书相关研究, 第五节分析协议的发展趋势。

2 协议设计不足

SSL/TLS 协议设计方面存在一些安全缺陷, 可能会被攻击者利用。本节综合介绍 SSL/TLS 在使用 CBC 加密模式的情况下可能产生的攻击, 主要包括 CBC 加密模式的工作原理和基于 CBC 填充的相关攻击, 包括 Lucky Thirteen 攻击、POODLE 攻击、Bleichenbacher 攻击和 DROWN 攻击等。

2.1 CBC padding 部分

SSL 协议较早的版本大量使用 CBC 分组加密的模式对数据加密, 如图 2 所示, 加密解密过程都是分块进行。在实际应用中, 会在明文最后一块内容不足时进行填充。利用对填充内容检查不足的特点, Serge Vaudenay 在文献[6]中引入了填充攻击, 从理论上证明了利用填充内容进行攻击的可能性, 并阐述在 SSL/TLS、IPSEC、WTLS 和 SSH2 中使用 CBC 模式具有严重的安全隐患。

在 CBC 模式下, 明文会被分割为固定大小的块。如果明文内容最后一块不足整块大小, 根据 PKCS#5 [7-8]格式, 会在最后一块中进行填充, 构成完整的块; 如果刚好具有块大小, 则填充一个新块。CBC 模式加密通过块密钥和初始向量来加密明文块, 其中明文的每个块都会和上一个明文块加密后的密文块进行异或运算, 每个字符只和每个块的对应位置相关。为了保证对同一明文加密后的密文不同, 其初始向量通常是随机生成的。

对密文进行解密也是分块进行的, 解密完成之后通常会检查是否符合规则, 如果不符合, 则会抛出填充错误的异常信息, 提示填充不正确。如果攻击者获得了合法的密文, 可以通过不断的向服务器发送构造的信息, 通过观察服务器返回的信息, 判断构造内容是否正确。如果收到错误提示信息, 则再进行修改。从查询中试探出 IV 的内容, 然后通过中间密文逆向解析可以得到明文。当然这样做需要进行大量的查询, 在一般的情况下, 攻击是很难成立的。但如果利用填充内容的特性时, 攻击就会变得高效很多。如 Lucky Thirteen、POODLE attack 等都利用了填充内容的特性来提高攻击的效率, 使攻击具备更高的实用性。

2.1.1 Lucky Thirteen 攻击

利用填充内容的不确定性, 攻击者能够通过修改填充的内容来进行测试。AlFardan 和 Paterson 在文献[9]公布了 Lucky Thirteen 攻击。Paterson 在采访中表示, 攻击者作为中间人拦截 TLS 数据包, 并对数据包进行篡改。由于传输给服务器的数据包具有特殊的排列方式, 其中的一个包头域含有 13 字节, 所以命名为“Lucky Thirteen”。

该攻击可以用来破解小块的 CBC 模式加密数据, 这是一种针对 TLS 记录协议的时间侧信道攻击, 要求 TLS 记录协议采用 MEE(MAC-Encode-Encrypt)

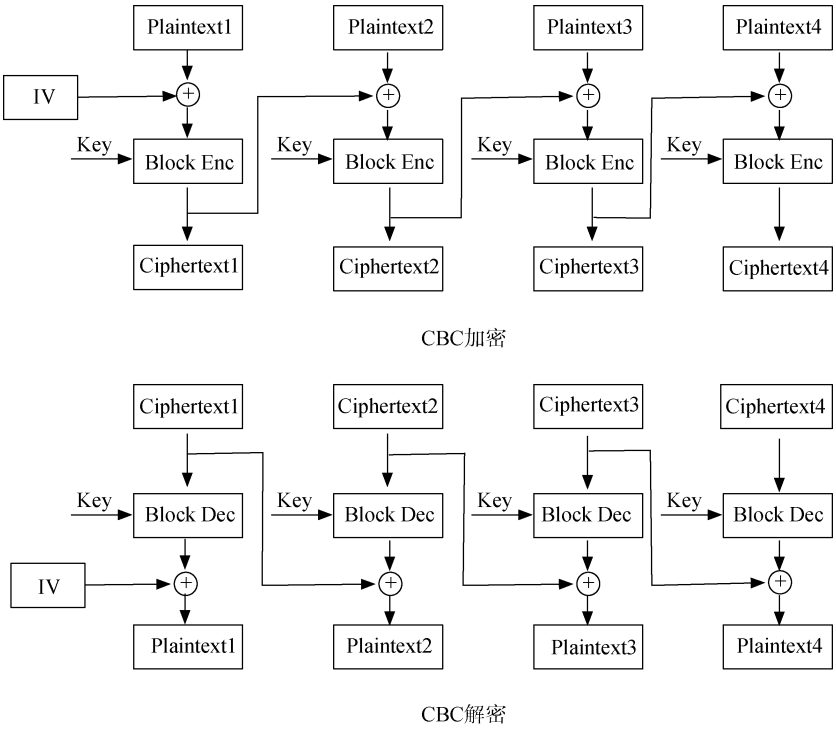

图 2　CBC 模式加密解密

Figure 2　Encryption and Decryption in CBC Mode

方式和分组密码的 CBC 模式。能够进行攻击的最重要原因是填充并使用 CBC 模式, 并且使用 TLS 完整性保护机制。攻击者可以通过修改填充字节, 然后观察服务器做出的响应和响应时间, 从中提取相关的信息, 来判断修改的内容是否正确。根据 AlFardan 等在文献[9]中的描述, TLS 协议会将错误的填充在 MAC 检测时当作零字节的填充, 这种错误填充的解密比正常协议解密所需要的时间要短很多, 从而形成了一个时间差。作者通过实验验证, 这个较小的时间足够被用来进行判断修改的内容是否正确。虽然实验中会存在各种因素的偶然对准(如 MAC 标签的大小, 密码块大小和报头字节的数量等), 但是这些因素对于处理 TLS 记录所花费时间上存在的时间差影响并不是很大。对于好的或者坏的填充, 这种差异最终将体现在错误消息出现在网络上的时间。这种利用时间的侧信道攻击在实际攻击中有广泛的应用, 具备一定程度的威胁性。

作者在文中提出的防范方案是添加随机时间延迟、使用 RC4、使用认证加密或小心应用 MEE-TLS-CBC 解密方式。然而在这些方法中, 使用 RC 4 代替加密方式是不可取的, RC 4 在近年来也被发现存在能够被利用的严重漏洞, 这种方式只是将漏洞转移到 RC 4 上。而增加时间延迟总体上会造成较长的延迟, 而且这个延迟的影响足以抵消它所带来的好处, 牺牲较高效率来换取安全的方式可能不是很好的选择, 所以增加随机时间延迟的方式实际上并不是可取的。而使用认证加密则是选择较安全的加密方式, 但是认证加密只在 TLS 1.2 版本上有实现, 对低版本的 TLS 协议并不能有效的解决。作者重点介绍的应用 MEE-TLS-CBC 解密方式, 核心的思想是确保一个对于 MEE-TLS-CBC 密文固定大小具有相同的处理时间, 把重点放在处理密文上, 而不是明文部分, 如块长度和其他长度信息, 可以一定程度上解决 TLS 的解密方式带来的时间侧信道问题。

2.1.2　POODLE 攻击

当对填充内容不做限定时, CBC 模式的缺陷将被放大, 利用填充的最后一个字节固定为填充长度的特性, 对 CBC 模式的攻击将变得更为可行。2014 年 10 月, Google 安全团队公布了 POODLE (Padding Oracle On Downgrade Legacy Encryption)[10], 这是 SSL 3 版本协议的漏洞, 攻击者可以利用这个漏洞破解小段的加密数据[9]。不同的浏览器对于 SSL 握手失败后采取的行为不同, 服务器端在协议握手匹配失败后, 会采取降级的措施。如表 1(内容来源于《HTTPS 权威指南》)所示, 开始握手时服务器会选择最高的协议版本给客户端, 如果握手失败, 服务器会重试旧版本的协议, 最后很可能被降级到 SSL 3 版本。而且存在类似 Windows 上的 IE 6 浏览器只支持 SSL 3。攻击者也可以用这种对协议的兼容性使握手降级到 SSL 3, 然后使用 POODLE 攻击来劫持会话。

表 1 浏览器自动降级

Table 1 Browser Automatic Downgrad List

Browser	First Con	Second Con	Third Con	Fourth Con
IE 6	SSL 3	SSL 2		
IE 7	TLS 1.0	SSL 3		
IE 8~10	TLS 1.0	SSL 3		
IE 11	TLS 1.2	TLS 1.0	SSL 3	
Safari 7	TLS 1.2	TLS 1.0	SSL 3	
Firefox 27	TLS 1.2	TLS 1.1	TLS 1.0	SSL 3
Chrome 33	TLS 1.2	TLS 1.1	TLS 1.0	SSL 3

POODLE 攻击能够在 SSLv3 上成功的主要原因是 CBC 填充模式在验证设计上的缺陷。这种验证方式只对明文进行了身份验证, 而没有对填充字节部分进行完整性验证。SSL 3 协议的填充方式是保留密文的最后一字节, 填写为填充的长度。但是没有限定填充的具体数据内容, 也没有进行 MAC 验证, 所以并没有规则来验证填充的内容是否被篡改过, 而只验证填充的长度是否正确。在这样的情况下, SSL 3 的填充就引入了严重的不安全性。

攻击者作为中间人进行 POODLE 攻击, 能够截获密文。正常情况下, 直接修改填充字节内容的方法是不可行的, 这会改变 MAC 值并引发错误, 只有当最后整个块都是填充数据时, 攻击者才能够可以进行自由的修改并且不会导致 MAC 校验的失败。而且攻击者尽管只对密文做了一位的修改, 也会导致密文解密发生大量的变化, 导致解密后的内容变成乱码。实际上, SSL 3 在做解密校验时, 只验证了最后的填充长度字节是否正确。在每次攻击尝试时, 攻击者只需要将要解密的块移至最后一块, 然后修改倒数第二块的最后一个字节进行查询, 多次修改直到服务器成功接收修改后的内容。攻击的主要思路是攻击者利用较长的 URL 和较短的请求体, 通过缩短 URL, 每次一个字节, 直到找到合适的填充长度。通过提交足够多次的修改来破解一个字节, 最后同步修改 URL 内容和请求体的大小来解密剩余字节。攻击的主要原理是数据块最后一个字节解密后的值为 15(假设块大小为 16)

$(C_i[15]) + C_{n-1}[15] = 15$

$P_i[15] + C_{i-1}[15] = (C_i[15]) = 15 + C_{n-1}[15]$

$P_i[15] = 15 + C_{n-1}[15] + C_{i-1}[15]$

面对这个严重的漏洞, Chrome 和 Firefox 等浏览器都禁止回退到 SSL 3, IETF 也出台了相应的方案来应对 POODLE 攻击。根据 RFC 中的标准化的防降级方法, 浏览器可以采用一个特殊的信号套件 TLS_FALLBACK_SCSV[11]信号, 当信号值改变时, 浏览器能够通知服务器自己被降级了。这个方案不仅是为了解决针对 SSL 3 的 POODLE 攻击, 也是为了能够长久解决降级攻击问题而提出的增添防降级信号方法。这个攻击的出现, 上大幅度推进了 SSL 3 的禁用, 促使服务器采用更安全的协议, 使用更安全的加密方式。

2.1.3 Bleichenbacher 攻击

当然填充的方式有多种, 其中具有固定格式编码的 PKCS 格式也受到填充的影响。Wagner 和 Schneier 在文献[12]中描述到, 攻击者能够通过修改 ClientHello 信息, 使 SSL 3 版本看起来像 SSL 2 版本的 ClientHello 信息, 强制服务器使用漏洞更多的 SSL 2。作为应对, 研究人员便提出了把版本信息包含在 PKCS 编码格式 ClientKeyExchange 的 PreMaster Secret 信息中, PKCS 编码格式如表 2。

表 2 PKCS# 1.5 编码格式

Table 2 PKCS# 1.5 Encoding Format

Blok Type	Padding	Separation Byte	Encapsulated Data
00 02	…	00	PreMasterSecret

这种编码方式具有固定的格式, 能够被利用来解析数据信息内容。Daniel Bleichenbacher 在文献[13]中, 利用编码方式的特性提出了新的攻击(Bleichenbacher attack)。基于 RSA 的 SSL 密码套件方式, 利用 PKCS# 1.5 的标准格式在可接受的时间内解密预主密钥内容。其中预主密钥是客户端使用 RSA 密码套件时生成的随机值, 使用服务器公钥加密后传送给服务器。服务器通过私钥解密后, 能够获得一份和客户端相同的预主密钥和随机值, 其前两个字节为版本号。中间人攻击者能够获取密文信息, 应用 Bleichenbacher 攻击 SSL 3 版本, 通过接收服务器返回的不同错误信息来辨别修改的正确性。

Bleichenbacher 攻击可以识别在 0x00 02 后以明文开始的密文信息, 通过 Padding Oracle 攻击来解密预主密钥, 更进一步利用可以取得会话密钥。充分利用 0x00 02 开头的特性, 假设攻击者获得密文 C_0, 想恢复出明文 M_0，m 为 M_0 的可取值空间。攻击方法是通过向服务器多次发送修改后的密文, 分析响应是正确还是错误来确定修改结果, 进而解密信息。如果收到正确, 则表示是 0x00 02 开头, 那么 $2B < m < 3B-1$, 且 $B= 2^{8(L-2)}$, 而且基于 RSA 加密的延展性, 可得 $C=(C_0 * S^e) \bmod N$, $N=(M_0 * S)^e \bmod N$。攻击者可用 C 进行查询, 如果收到错误则增加 S, 并重复上一步骤。攻击者可以利用 0x00 02 的特性大幅度缩小取值区间, $2B < M_0 * S - rN < 3B$, 因此能够降低范围

$(2B+rN)/S < M_0 < (3B+rN)/S$。然后迭代选择 S, 进行 Oracle 查询, 计算新的 r 值。攻击者可以不断缩小包含 M_0 的范围, 不断重复直到最后只剩唯一解。

这个攻击在被发现不久之后, 研究人员便对错误信息做了统一, 使得这个侧信道不能被使用, 也让这个攻击在短期内得以解决。针对这个改进, Klima、Pokorny 和 Rosa 对 Bleichenbacher 攻击在文献[14]中做了改进, 他们在优化中重新定义符合 PKCS 明文的可能区间值。根据 SSL/TLS 规范:

1. PreMasterSecret 正好是 46 个字节;

2. PreMasterSecret 前缀有两个版本字节;

3. 填充字节不等于 00;

4. 符合明文 Mi 的 PKCS 包含将填充与有效载荷数据分开的空字节。

根据协议标准, 可以得知一些有效信息。填充内容已知, 其中包含 2 个类型字节, 单个空字节作为分隔符, 2 个字节的版本号和 46 个 PreMasterSecret 字节。其中加密内容中不仅有 PreMasterSecret 的值, 还存在版本号信息, 攻击者可以利用验证构造的版本号的正确性来达到查询的目的。为应对这种握手信息的泄漏, 设定了直到对 Finished 消息的验证和解密发现密钥不同才中止会话。后来, Bleichenbacher 攻击又被 Bardou、Focardi 等人进行了改进, 在文献[15]中提出了相应的改进方法, 使得攻击执行得更快, 大量减少查询的次数, 极大的增加了攻击的效率。他们还结合他们的结果做出了分析, 从结果上来看是一个非常明显的改进。

为了应对 Bleichenbacher 攻击, 从 RFC 2246(TLS 1.0)开始的所有 TLS RFC 都建议“以与正确格式化的 RSA 块不可区分的方式, 处理错误格式化的消息”。在文献[14]中, Bardou 等人通过实验展示了这个补救工作并没有被成功实现。然后作者提出了四个新的 Bleichenbacher 侧信道攻击和三个成功的 Bleichenbacher 攻击针对 Java 安全套接字扩展(JSSE)SSL / TLS 实现和硬件安全家电使用 Cavium NITROX SSL 加速器芯片。作者验证了 Bleichenbacher 攻击还能够成功的对 SSL／TLS 协议进行攻击, 攻击的成功代表着对协议的改善工作并没有结束, 仍需要继续推进。

2.1.4 DROWN 攻击

研究者们进一步对 Bleichenbacher 攻击进行了深入的研究。在 2016 年, 攻击者通过利用 PKCS#1 v1.5 加密填充的遗留问题, 并结合 SSL 2 相关漏洞发起了对 SSL／TLS 近年来极具影响力的一场攻击。在文献[16]中, DROWN(Decrypting RSA using Obsolete and Weakened eNcryption)攻击威胁到了百万级的服务器。SSL 2 虽然是已经退役的协议, 但是依然还有大量的服务器支持该协议, 没有完全禁用, 从而导致严重的后果。利用该漏洞, 观察服务器的响应来解密的 RSA 密文信息。并且利用这个漏洞, 攻击者可以在没有 RSA 密钥私钥的情况下, 完成跨协议的攻击, 来解密 SSL 3 或者更高级的 TLS 会话。就算服务器本身不支持 SSL 2, 但是与使用 SSL 2 的服务器共享 RSA 密钥, 也会受到 DROWN 攻击连锁反应的影响。

DROWN 攻击区别于 Bleichenbacher 攻击是它利用 SSL 2 解密 ClientKeyExchange 信息。在攻击的实现细节中, 要使用到 Bleichenbacher 的攻击方法, 这里需要先解决两个问题, 首先是攻击者需要把 SSL 密文转换成有效的 SSL 2 密钥交换信息, 才能应用 Bleichenbacher 攻击, 其次是 Oracle 查询需要严格的检测非填充部分的长度。

根据 Bardou 等人在文献[14]发表的论文介绍, Bleichenbacher 攻击需要大约百万级以上的查询次数。而 DROWN 攻击则利用了一些特殊的攻击技巧来优化攻击效率, 使用 Trimmer 来剪切截获的经过 RSA 加密的明文信息, 然后精心构造 00|02|PS|00|密文消息的结构。然后利用 SSL 协议的出口协议来弱化会话密钥, 攻击者将截断会话密钥至 5 字节。然后进行 Oracle 查询, 一旦成功查询, 便可以对成功篡改的密文做多次平移操作, 从而对未知长明文消息做分段攻击。最后将各个小段的解密信息组合起来, 构成解密明文信息。作者还对 Bleichenbacher 攻击做出了新的改进, 他们发现此侧信道仍存在可被利用的方法, 比如使用同一个剪切过的 RSA 密文对 Oracle 做两次询问。若剪切正确, 在 Oracle 的两次答复会使用正确的短会话密钥返回结果, 让攻击者是否通过成功验证得知剪切是否成功。否则 Oracle-E 的两次答复会使用两个不同的随机数伪装“短会话密钥”, 通过仔细验证收到的返回信息, 攻击者也能得知剪切的内容是否成功。攻击者还利用了 SSL 2 协议的漏洞中的一种“高效设计”, 即允许客户端通过明文指令要求服务器把高位 RSA 加密的密钥置零, 攻击者对此特性加以利用, 能达到将服务器的会话密钥设置为只含有一个字节的信息量。

充分利用这些特点优化攻击, 使得 DROWN 攻击能够通过以下方式进行。收集大量的 TLS RSA 密钥交换信息, 大约 1000 个左右就可以进行解密。把截获的含有 48 字节预主密钥的 TLS 密文截断成多个小段, 然后组建成 RSA PKCS#1 v1.5 编码格式。构造

00|02|PS|00| 小段密文结构，使用改进的 Bleichenbacher 攻击，发起多个 SSL 2 Export_40 连接，并进行多次 Oracle 解密，最后把解密后的明文组建成原来的消息。

研究人员表示他们能够通过 1000 个记录的握手信息, 40000 个 SSL 2 连接和 2^{50} 次离线计算，在使用 2048 字节 RSA 密钥的服务器中，解密 TLS 1.2 握手信息，而且如果使用共享计算资源，可以在 8 小时内完成攻击，成本约 440 美元。他们还指出如果使用特殊的 DROWN 攻击可以降低 SSL 2 连接到 14000 个。

DROWN 攻击威胁到了大量的用户，能够破解会话信息，如果被截获的信息是银行账号或者国家机密信息，危害将不可估计。攻击成本也不高，而且如果攻击对象是普通用户，能预先获取的信息更多，解密过程将更简单，速度更快，成本将更低。DROWN 攻击充分体现了使用最新协议，及时将废弃协议完全停止的重要性，也体现了密钥重用带来的重大危害，使用过时的加密协议和弱加密原语对会话产生严重的威胁。

很多加密方式在最初设计的时候，并没有考虑太多的应用安全性问题。在实际的应用中，使用可靠加密原语和侧信道强化的算法是很有必要的。侧信道的大量信息泄漏为攻击者提供了检测判断的条件，使得攻击者将这些条件作为修改密文后的测试结果。而服务器端返回的错误信息提示越多，攻击者检测就越便利。减少一些非必要的错误返回信息在服务器端是很有必要的，侧信道中泄漏的信息也需要尽可能的减少。SSL 虽然定义了相同的错误消息填充和解密，但是仍然不能避免时间侧信道的攻击。所以类似 CBC 这样的弱加密方案应该被重新审视，应该考虑使用更安全，更有效的加密方式，并且废弃的协议一定要及时完全禁用。通过利用老版本协议，DROWN 这种大规模的攻击极具很严重威胁性。一些低版本的协议严重影响着新协议的安全性，所以推进 SSL 3 的完全禁用也具有很重要的意义。

2.2 基于 RC4 的相关研究

2.2.1 RC4 加密原理

RC4 是 Ron Rivest 在 1987 年设计的加密算法，作为一种较早出现仍被大量使用的高效加密算法。由于对 CBC 模式的攻击大量出现，加密方式的重心开始转移到 RC4，使得 RC4 开始广泛的受到关注，也开始被大量研究人员重视。虽然早在 2001 年, RC4 就被 Fluhrer、Mantin 和 Shamir 等研究人员证明其密钥调度算法存在缺陷[17]，但是当时没有实际的攻击实例。后来的研究人员在分析大量的弱密钥后，发现密钥中的少量关键位置对影响大量初始状态输出位值具有不可忽略的概率作用。也就是说，如果利用某些位置的已知明文(如 HTTP 报头固定值)破解密钥流的一部分内容，可以被放大利用到破解流上的其他对应位置。理论突破到实际应用往往会存在着一段距离，但是随着近几年对 RC4 研究的深入，研究人员们构造了一些能够实际测量的攻击，验证了理论的正确性，也就加速了 RC4 在 TLS 应用上的退役。

RC4 从设计之初到现在应用一直非常广泛，原因就在于其加密方式简单，效率高，并在线上实时通讯加解密领域有着广泛的应用。如图 3 所示, RC4 的加密原理并不复杂，加密算法主要分成两部分。第一部分是密钥调度算法(KSA)，通过密钥 key 来初始化 S 状态。然后第二部分是伪随机数生成算法(PRGA)，通过使用 KSA 初始化后的 S，生成伪随机序列，并更新 S，如图 4 所示(来源于 Google)。

```
RC4 Key Scheduling(KSA)

Input: key K of n bytes
Output: initial internal state S0

j, S0 =0 range(256)
for i in range(256):
    j += S0[i] + K[i % len(K)]
    swap(S0[i], S0[j])
return S0

RC4 Key stream Generation(PRGA)

Input: internal state S0 = KSA(K)
Output: Key stream byte S1

i, j, S1 = 0, 0, S0
while True:
    i += 1
    j += S1[i]
    swap(S1[i], S1[j])
    yield S1[S1[i] + S1[j]]
```

图 3 RC4 加密算法

Figure 3 RC4 Encryption Algorithm

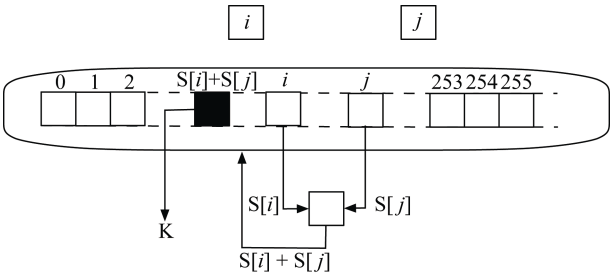

图 4 RC4 解密

Figure 4 RC4 Decryption

AlFardan 等人在文献[18]中系统的分析了 RC4 的安全性和近几年来 RC4 出现的种种状况。当 RC4 被用于 TLS 加密时，AlFardan 设计了实验对 TLS 进

行了纯密文文本恢复攻击。提供了两种明文恢复攻击方法, 而且这些攻击也可实际应用于破解 RC4 加密的 TLS 会话。攻击主要是根据 Mantin 和 Shamir 等人在 RC4 流中发现的两个偏差进行的设计。

2.2.2 单字节偏差攻击

RC4 在密码学中存在最重要的问题就是加密偏差问题。Mantin 等人通过统计分析、贝叶斯分析等方法[19]发现密钥流中的第二字节倾向于为 0 的概率比一般情况要高, 为 1/128 而不是正常的 1/256。

S 初始化结束生成密钥流的过程。最初 *i* 和 *j* 都为 0。用 *X* 表示 $S_0[1]$, 在第一轮中将 *i* 更新为 1, 并将 *j* 更新为 $0 + S_0[1] = X$, 并且交换 1 和 *X* 位置的内容。第一个输出是 $S_1[X + Y]$, 它基本上属于均匀概率分布的任何值。现在假设 $S_0[2] = 0$。在第二轮中, *i* 被递增到 2, 并且 *j* 增加到 $X + 0 = X$。

进行图 5 的交换。对于大约 $1 / N$ 的密钥, 第二轮输出为 0, 概率为 1, 而对于其他的 $1 – 1 / N$ 个密钥, 第二个输出为 0, 概率为 $1 / N$ 均匀分布。导致的结果是第二位置输出为 0 的总概率约为 $2/N$, 为正常可能性的两倍。

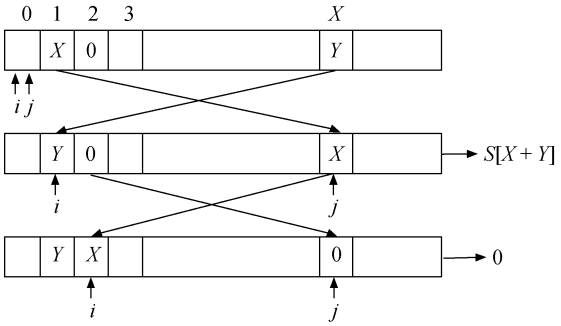

图 5 RC4 加密开始两轮 $S_0[2]=0$

Figure 5 Two Rounds of RC4 Encryption with $S_0[2]=0$

这种类型的偏差在密码学上是非常危险的, 因为只要知道了 RC4 密钥流中的第二字节倾向于 0, 那么就能够猜测出密文的第二字节。其他存在偏差的字节位置也是会出现类似的情况。对 TLS 会话进行攻击, 需要建立多个连接, 获取相同明文被多种不同密钥加密的密文数据。观察第二字节的值, 如果出现频率较高, 那么这个值很可能就对应明文内容中的特定值。

在 AlFardan 等人[18]发表的论文中, 其中一个攻击通过分析 2^{44} 个 RC4 密钥的密钥流, 发现在最开始的 256 个字节中都存在着多种偏差。他们通过改进算法, 分别处理不同位置的偏差。最后的效果能够达到在 2^{32} 个数据样本下, 破解全部 256 字节的内容。而且在被攻击字段存在已知内容的情况下, 数据样本的数量能减少到 2^{28}, 远低于应该存在的 2^{128} 的安全性。

在现实生活中, 这种攻击暂时还是很难实施的。因为最少需要收集 2^{28} 个数据样本, 被动攻击很难在合理的时间内符合要求。就算每秒进行一次连接, 也需要 8 年左右的时间才能收集齐。所以只有进行主动攻击, 通过注入 JavaScript 恶意脚本进行中间人攻击。而且能破解的内容是前 256 字节, 如果浏览器把有效内容放到 256 字节后, 这个攻击就很难取得有效的内容。

2.2.3 双字节偏差

除了单字节偏差, 在 RC4 流中还发现了存在多字节偏差。与单字节偏移不同, 大多数所识别的多字节偏移不只是出现在单一位置, 而是以规律的间隔周期性连续出现在加密流中。

在 AlFardan 等人[18]公布的第二个攻击中, 展示了如何使用双字节偏差来破解明文。利用双字节进行攻击时不需要大量不同的 RC4 密钥加密的样本, 在同一个连接上能够获取多个样本, 从而有效的减少了需要建立的连接数。双字节攻击能够在 $13*2^{30}$ 个数据样本下, 破解 16 字节的明文数据内容。为了使目标 cookie 处于 TLS 会话内容的固定位置, 作者在 HTTP 头中添加了填充, 使得加密的 POST 请求达到 512 字节, 会产生一些额外的开销。按照每小时生成 600 万个密文的速度进行实验, 需要大约 2000 小时来收集数据。这个攻击方式会产生大量的网络流量, 在实际网络中, 这个攻击暂时不能构成威胁, 但是还是证实了双字节偏差确实可以被利用。

2.2.4 攻击改进

在文献[17]中, Fluhre 等人提出不变性弱点(Invariance Weakness), 这是 RC4 加密中的存在的 L 形键模式。它一旦存在于 RC4 键中, 在整个初始化过程中会保持部分置换状态。当用 PRGA 算法处理时, 该完整部分包括置换的最低有效位, 通过流的长前缀确定伪随机输出流的最低有效位, 如图 6 所示。这些存在偏差的流字节与明文字节进行异或运算, 导致从密文到明文的转换存在着偏差。这些模式通常发生在不同数量的 LSB(Least significant bits), 1 个 LSB、2 个 LSB 乃至 7 个 LSB, 会分别导致出现不同类别的弱 RC4 密钥。

SSL 协议中有许多密码套件都使用 RC4 方式进行加密。在 SSL 握手过程中, 信息通信使用 RC4 加密方式为上游和下游通信生成加密密钥。上游密钥用于客户端到服务器通信的加密, 而下游密钥用于服务器到客户端通信的加密。重要的点是加密是有

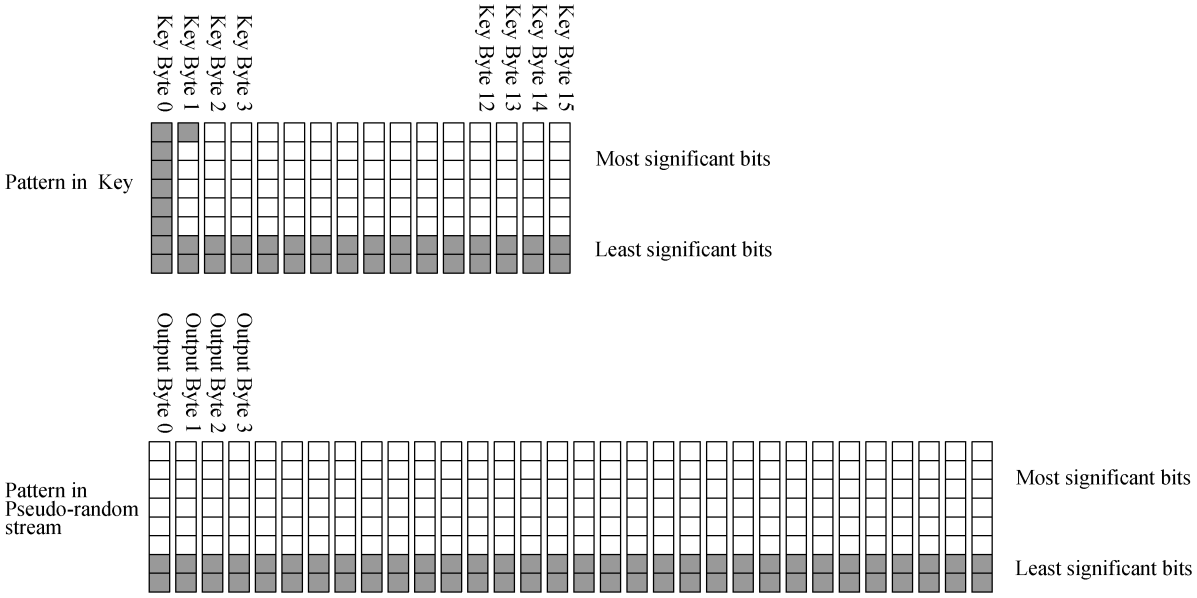

图 6 Invariance Weakness

Figure 6 Invariance Weakness

状态的, 使用第一个密钥流字节来加密第一个消息, 后续的密钥流字节用于加密下一个消息等。考虑到不变性弱点仅在密钥流的前 100 个字节中, 它只能用于受保护的上游流量的前 100 个字节和受保护下游流量的前 100 个字节。假设每个方向上的第一个加密消息是 SSL 握手完成消息(SSL 的典型使用为 36 字节), 则大约 64 字节的密文数据将被保留。上游密钥流的前 36 个字节用于加密 Finished 消息, 下一个字节开始用于加密实际应用数据。

虽然对在 TLS 中 RC4 加密方式进行了很多有效的攻击, 但根据 Garman 等人[20]在 2015 年的统计中发现, 使用 RC4 加密的流量还是占了约 30%的 TLS 流量比例。Garman 等人[20]也在 2015 年 3 月发布他们在 TLS 中对 RC4 的攻击细节, 攻击的重点是用来恢复用户密码。通过应用贝叶斯分析方法将密码的先验信息和收集的密文结合, 转化为密码的后验概率。论文中的结果显示, 对 RC4 的攻击效果将会越做越好, 目前能够做到 2^{26} 次加密用来恢复用户密码具有非常良好的成功率, 比之前 2^{34} 次的效果又要好很多。作者分析了不同参数条件对攻击效果的影响, 如使用先验概率构造的成功率会显著较高; 随着密码长度的增加, 攻击的成功率显著下降; 允许大的密码测试数量值能提高攻击的成功率; 使用 Base64 编码方式, 会增加密码的长度, 但由于编码会引入冗余, 又会有利于攻击, 所以整体效果上也会有利于攻击。

Vanhoef 和 Piessens[21] 也在 2015 年 7 月提出了进一步的改进, 打破 Wi-Fi 保护访问时间密钥完整性协议(WPA-TKIP), 并针对 TLS 协议设计实用的明文恢复攻击。使用统计假设检验的方法, 发现 RC4 密钥流中新的偏差, 并揭示了初始密钥流字节中的许多新偏差。利用这些偏差, 尽可能的减少返回的候选列表。作者还引入了一种生成大量相同数据包的方法来打破 WPA-TKIP, 这些分组通过生成其明文候选列表来解密, 并且使用冗余分组结构来剔除不良候选。从解密的数据包中可以得到 TKIP MIC 密钥, 用于解密和注入数据包。作者通过在受害者的浏览器中运行恶意的 Javascript , 收集 $9*2^{27}$ 个左右的 HTTPS 加密后的请求, 最好的效果能够将攻击时间缩短到约 52 小时。每个请求都是具有相同密钥加密后的密文, 为了限制在 75 小时内完成攻击, 需要在每秒内发出约 4450 个请求。虽然对比 AlFardan 设计的攻击需要 2000 小时要小很多, 但是在距离实际可用于现实攻击还差一点。因为在短时间内发送如此大量的请求会很容易被服务器检测出来, 但这个实验结果确实将对 RC4 的攻击又向实际可行的边缘推进了一大步。

虽然目前对于在 TLS 中使用 RC4 加密的攻击[22]还没达到实际可被利用的程度, 但是它的安全期限已经变得很短了, 在理想的情况下还是能够被破解的。从对 RC4 的攻击历程来看, 只要是存在漏洞的算法, 在被长久的研究中, 总是会被找到攻击方案的。并且攻击方法会被不停的改进, 直至达到现实可用的程度。一般的加密方案都有一定的存活年限的,

而不是一直长存的, 所以现实应用中需要实时地更新加密方式。总体来说, 推进 RC4 在 TLS 上的停用是非常有必要的。在文献[6]中, Vaudenay 强烈地建议 Web 应用程序管理员应考虑在其应用程序的 TLS 配置中禁用 RC 4; 鼓励 Web 浏览器在 TLS 配置中禁用 RC 4。 当然, 最好的方案是按照 RFC 7525 [23]的建议, 使用 AES- GCM 加密方式。但 AES-GCM 只在 TLS 1.2 版本才支持。因此, 推进 TLS 1.2 以及新版本协议 TLS 1.3 的广泛部署, 才是长久之计。

3 协议实现漏洞

在SSL/TLS的协议应用中, 除了CBC模式, RC4之类的协议设计问题, 近几年还发现了一些协议实现上相关的漏洞。K. Bhargavan 团队的相关研究中, 有三次握手攻击, SLOTH 攻击, 降级弹性等具有代表性的研究。SSL/TLS 协议的设计是非常复杂的, 其中的一些 RFC 的定义很难被完美的实现, 导致协议在实现上存在着一些致命的漏洞。近几年有很多研究人员对 TLS 的实现做了深入的研究, 如分析 OpenSSL、MatrixSSL 等, 发现了一些比较有影响力的漏洞, 类似于 HeartBleed、FREAK 等。在发现漏洞的过程中, 也有研究人员开发新的方式, 其中利用自动状态机来发现漏洞的方式就是典型的一种方式。

3.1 三次握手攻击

三次握手攻击, 主要是利用 TLS 协议的 RSA/DH 密钥交换缺陷和会话恢复的缺陷来绕过防护措施。作为 TLS 会话的核心, 会话的主密钥是非常重要的部分, 而它的产生需要经过客户端驱动产生。客户端会生成预主密钥和一个随机值发送给服务器, 而服务器也会生成一个随机值发送给客户端。而且预主密钥是加密传输的, 随后根据客户端和服务器端的三个值生成主密钥。

三次握手攻击在实际进行的时候需要利用恶意网站进行中间人攻击配合。客户端把自己生成的预主密钥和随机值发送给恶意网站, 而恶意网站作为正常的接受方, 可以通过正常方式解密客户端的预主密钥。然后恶意网站再向目标服务器发送请求, 建立新的连接, 并使用从客户端接收到预主密钥和随机值, 并把服务器端的随机值转发给客户端。当握手过程结束的时候, 恶意网站就拥有了和客户端, 服务器相同的主密钥, 并分别和客户端, 服务器建立了连接。但是由于两个连接使用的证书是不一样的, 而且每个连接都拥有不同的 verify_data 值。这就需要利用会话恢复的机制来进行下一步的破解。在会话恢复的时候, 并不要求进行证书的验证, 而且也没有身份的验证。而仅仅通过主密钥进行通讯双方的身份认证, 从而可以在握手结束的时候, Finished 消息将一致, verify_data 相同。这时攻击者再发起重协商, 利用客户端的证书伪造自己的身份, 从而控制两边的连接, 发送任意修改的数据内容, 从而完成攻击。作者也提出一些比较可靠的解决方案, 如禁用重协商, 只启用 ECDHE 加密套件和对所有网站都要求验证客户端证书等, 具体每种解决方案都有一定的适用范围[24]。

3.2 SLOTH 攻击

对于 MD5 和 SHA-1 的碰撞, 很多实践者认为, 它们在这些协议中的使用仅依赖于第二前级映像, 不受碰撞的影响。Bhargavan 等人在文献[26]中则是系统的进行研究并揭示这个论点的薄弱性, 在密钥交换协议上设计出一类新的基于哈希结构的, 有效的冲突查找算法的碰撞攻击。在 TLS 1.2 客户端身份验证上进行演示攻击和 TLS 1.1、IKEv2 和 SSH-2 进行降级攻击, 迫使多个 TLS 库进行更新, 并禁用主流协议中的弱哈希函数。

SLOTH(security losses from obsolete and truncated transcript hashes)是一种攻击者强迫使用弱哈希算法的攻击, 如客户端 TLS 1.2 版本, 强迫使用 MD 5 算法进行加密。在握手的初期, 客户端将 ClientHello 数据包发生给服务器; 数据包中声明了服务器可以使用的签名和加密算法。但是攻击者可以截获该数据包, 并且向客户端发送一个要求更改算法的数据包, 迫使客户端改变。至此, 攻击者开始了冒充目标服务器的攻击过程。位于客户端和服务器端之间的攻击者通过发送 Server Hello、Certificate 和 Server Key Exchange 数据包响应客户端请求。在 Server Key Exchange 中, 攻击者使用 RSA-MD5 算法替换客户端实际指定的算法。客户端接收到“服务器端”的响应, 被迫选择使用弱哈希算法。随后, 客户端再次发送 Client Key Exchange 响应, 握手成功。被成功攻击后, 中间人攻击者就可以冒充服务器, 解密所有加密的流量。SLOTH 也可以反向进行, 致使服务器端被降级攻击。针对这个攻击, 最好的方案是在 TLS 1.2 后的版本中删除 MD5 / SHA-1 等弱哈希算法的支持, 禁用这些弱哈希算法, 而且对 SSH 和 VPN 也需要进行同样的更改[25]。

3.3 弹性降级

降级攻击是攻击者使用一些策略, 让服务器和客户端采用比较弱的协议或者加密套件的一种攻击。TLS、SSH、IPsec 和 ZRTP 的密钥交换协议都是高度可配置的, 并能够支持多种版本的协议、加密算

法和参数。这种多样可选性设计原本是为了让协议能够具备更好的应用范围。Bhargavan 等人在文献[26]中设计了一个正式的框架来研究弹性降级和密钥交换协议的其他安全属性的关系, Bhargavan 等人还剖析和归类了经典攻击和一些新型攻击, 总结其中的原因, 并调查现有的标准防降级弹性的相关内容。然后将这些结果与降级安全性结合起来, 分析几个协议实现的条件。最后设计了一套降低安全性的模式, 并解释如何使用它们来加强现有协议的安全性。

在文献[26]中多次提及到现代协议具有加密可选灵活性, 提供了多种协议和密码模式的可配置选择, 使得在两个对等体之间实际执行的密钥交换取决于交换的协商阶段。在协议实现方面, 灵活可选是不可或缺的特性。不幸的是, 对算法灵活选择性的支持也为降级攻击提供了机会。在这种攻击中, 主动网络攻击者干扰协商, 导致诚实的对等方完成密钥交换后, 使用较弱的加密方式。为了防止对特定协议模式的攻击, 关闭导致其协商的配置也是必要的。

Bhargavan 在文献[27]中对降级弹性的探讨是建立在 miTLS 上的, 考虑 Initiator 和 Responder, 引入了一个降级保护谓词 DP, 可以对成对的配置进行操作, 并且确定了一组配置来降低弹性。还引入了一个函数 Nego, 将两个相反角色的配置映射到协议模式, 在正常情况下应该进行协商。如果符合 DP 的配置开始的两个对等体只能协商 Nego 确定的模式, 那么在有攻击者的情况下, 协议也会降级安全。例如, 对于 TLS 协议, Nego 的具体实例会确定两个 TLS 对等体配置, 会确定进行 TLS 1.2 和密码套件的协商。但是, 如果服务器支持不安全模式, 例如 DHE-EXPORT 密码套件, 则对手可能会迫使其降级到不安全的模式。这表明如果没有防止降级的对策, TLS 1.2 的使用将不具备安全性。另一方面, 如果只具有一种可选模式的协议显然是降级安全的, 文献[27]证实了 TLS 1.2 版本协议并不是降级安全的, 描述了 IKEv2 和 ZRTP 新的降级漏洞。提出相应的配置方案来避免出现这些漏洞, 最后证明 SSHv2 是降级安全的。

3.4 利用状态机检测漏洞

在 TLS 实现中, 还存在着很多细节的问题需要解决。在文献[28]中, Bhargavan 等人研究 TLS 握手的可验证安全性。通过 miTLS 验证了许多协议版本, 配置和密码套件的主流浏览器和服务器的互操作性, 并提供了在应用层可证明的安全性。在 TLS 协议的实现中, 必须处理各种版本协议和相关的扩展, 认证模式和密钥交换的方法, 而不同的组合在客户端和服务器端之间又形成了不同的消息序列。Beurdouche 等人在文献[29]中设计了复合状态机来解决不同协议模式之间的细节问题。通过系统测试一些流行的开源 TLS, 发现了几个在库里面隐藏多年的安全漏洞。

自动状态机是第一个用 C 语言编写, 进行过验证的 TLS 状态机综合实现, 并且可以嵌入到 OpenSSL 中进行使用。利用论文中的方法可以对密码协议库的核心组件进行形式化验证。

经过 20 年的演变, TLS 具有很多版本, 扩展和密码套件, 而且其中的一部分已经不被使用或者已经被确认为不安全。但客户端和服务器端的实现为了保证具有灵活性、互操作性, 在部署时通常会支持部分不安全的密码套件。虽然握手期间的 TLS 会话的特定参数会通过 MAC 值来验证, 但不足以保障双方支持的不安全加密方式被攻击者利用。当客户端或者服务器端仅支持安全协议版本, 不管对等方支持什么不安全的套件, 都只能和安全方保持一致。

在具体实现中, 就 TLS 1.0～TLS 1.2 而言, 有一些细节地方不能被忽视。首先, 协议中消息顺序是精心设计的, 从互操作性和安全性的角度是不能随意更改的, 比如 ServerCCS 消息就必须在 Server Finished 消息之前发生。其次, 客户端和服务器端必须要能区分真正可选的消息, 而不是简单的线性接收和发送, 类似于区分 ServerNewSessionTicket 和当前密钥交换消息。然后, 不能过早地计算会话参数和密钥, 客户端应该在接收到服务器的消息后才确认握手过程完成。还有一些版本的协议, 如 SSL 23, DTLS 都存在一些细节问题。在 SSL 3 中, 客户端可能根本不发送 ClientCertificate 信息, 而 DTLS 则允许服务器使用新的 HelloVerifyRequest 消息来响应 ClientHello。在 TLS 库中, 还有许多网络上不常用的密码协议, 如 PSK, DH anon/ECDH anon, DHE PSK 等, 在使用这些密钥套件时需要一些额外的协商参数。而对于重协商, 在同一个连接上建立多个 TLS 握手, 从逻辑上看非常清晰, 但是实现起来就相当棘手, 也就导致了很多利用重协商漏洞的攻击。

Beurdouche 在用文献[30]中开发的 FLEXTLS 工具来验证这些实现中的漏洞。该工具是建立在 miTLS 上, 是经过验证的 TLS 实现。利用强大的消息传送和密码库, FLEXTLS 能够用来评估协议的漏洞。使用该工具, 发现了一些对 TLS 实现的攻击, 如 SKIP 和 FREAK, 也为 FREAK 和 Logjam 提供了验证演示。在文献[29]中, 也应用了 FLEXTLS 工具来进行评估验证, 测出很多细节漏洞。如 OpenSSL 中 EarlyCCS(CCS 严格意义上不是握手信息, 不出现在

握手记录中, 不受客户端或服务器端状态机控制, 可以出现在 ServerHello 后的任意位置)。中间人通过在 ServerHello 之后向对方注入 CCS 消息, 提前设置弱记录密钥, 然后让他们完成握手, 拦截合法的 CCS 消息, DH Certificate(客户端忽略 ClientKeyExchange, 可能导致 client impersonation attack), Server-Gated Crypto (SGC 允许客户端收到 ServerHello 后重新握手, 但是表明某些扩展是否被使用的信息会在新握手中消失), Export RSA(512 位弱签名), Static DH(使用 DHE 或者 ECDHE 时, 如果证书包含 ECDH 公钥, 而且客户端不接受 ServerKeyExchange, 则会回滚到 Static ECDH, 用服务器证书的公钥, 导致前向安全性丢失)。还有 JSSE 相关问题, 如 Client Flaws(处理特定于某些密码的可选消息, 客户端和服务器状态机都允许跳过消息, 导致 server impersonation attack), Server Flaws(JSSE 服务器存在类似的允许客户端跳过消息)。还有一些关于 NSS, Mono, CyaSSL 和 GnuTLS 相关漏洞。这个状态机发现了很多协议实现中存在的漏洞, 作者也对大部分漏洞进行了验证实验, 实际证明了漏洞的存在, 并提出了针对性的改进建议。

3.5 心脏出血

Heartbleed 是 OpenSSL 实现上存在的漏洞。通过发送特殊信号 Heartbeat 给服务器, 来查看服务器是否在线, 当服务器在线时, 会发送回复信息给主机, 然后允许进行安全通信, 服务器和主机会间断性的发送这个信号来确保对方是否在线。心跳包设计之初是为了能够解决及时检测连接状态问题, 相比 TCP 的 keepAlive 机制具有更大的灵活性, 可以自己来控制检测的间隔和检测的方式[31]。

心脏出血漏洞最先被谷歌安全中心的 Neel Mehta 提出来, 发现代码实现中没有对内容分配的限制。攻击者能够利用 Heartbeat 发送恶意信息给服务器, 强制 OpenSSL 服务器读取任意内存位置, 攻击者还能够控制心跳的大小和结构, 一次可以收到 64kb 的服务器内存数据。经过多次请求, 攻击者能够得到更多的服务器内存信息, 有可能包括大量的邮件地址、密码等信息。OpenSSL 从 1.0.1 版本到 1.0.1f 版本都存在心脏出血漏洞, 可想而知, 数以万计的服务器都会受到影响, 根据 Netcraft 估计, 17% 的服务器都受到了影响, 大约在 50 万台。

当然最致命的是攻击者可以用这种方式来获取加密密钥, 泄漏的密钥允许攻击者解密过去和未来的相关流量, 并且可以随意假冒服务。而且 X.509 证书中的加密和签名提供的保护都可以被绕过, 想要从泄漏中恢复, 需要修补漏洞, 并撤销被泄漏的密钥, 重新发布和分发新密钥。即使这样做, 攻击者仍然可以解密过去拦截的流量, 而且这些更新操作都必须由服务的业主来完成, 更新的过程比较困难。

对该问题的解决最有效的方式是打补丁, 可以选择升级 OpenSSL 版本, 也可以配置 OpenSSL 协议删除对心跳协议的支持。由于攻击者可能获取到密钥, 所以最好替换私钥, 如果要想保证过去的加密信息无法被泄漏的服务器私钥影响, 则是需要部署具有前向保密性的加密方式。

3.6 Freak 和 Logjam 攻击

继 Heartbleed 后, OpenSSL 又被公布发现了 Freak(Factoring RSA Export Keys)漏洞。虽然这个漏洞的影响没有 Heartbleed 影响力那么大, 但是很容易被攻击者利用进行中间人攻击。该漏洞源自于 20 世纪 90 年代, 美国限制出口高强度的加密算法, 限制加密强度最大为 40 位, 密钥交换强度最大为 512 位。从现在的计算速度来看, 这种出口密码套件的安全强度是非常弱的, 可以在几小时内完成破解。要进行 Freak 攻击, 需要克服两个障碍, 首先被注入的消息需要被目标服务器上的强 RSA 签名。然后为了修改 TLS 握手的消息, 攻击者需要伪造 Finished 消息, 使得对消息的修改变得合法化。完成这个中间人攻击[29]:

1. 在 ClientHello 中要求一个标准的 RSA 密码套件;

2. MITM 攻击者将服务器消息改为“Export RSA”;

3. 服务器使用 512 位 Export RSA 进行响应, 并使用密钥进行签名;

4. 根据 OpenSSL 的漏洞, 客户端会接收这个弱加密方式;

5. 攻击者对这个弱 RSA 进行破解, 恢复 RSA 解密密钥;

6. 当客户端将发送预主密钥给服务器时, 攻击者变可以解密它, 恢复出 TLS 主密钥;

7. 攻击完成后, 可以注入任意内容。

实际上不仅是 OpenSSL 直接受到这个攻击的影响, Android 系统很多使用的 OpenSSL 库也都存在风险, 而且随着攻击面的扩大, Apple 的 SSL / TLS(Secure Transport)和 Microsoft 的 SSL/TLS 库(Schannel)也受到了影响。对于 Freak 漏洞, 只有建议各服务提供商移除对出口密码套件的支持, 因为具备出口密码套件的支持是这个攻击成立的必要条件。

在 FREAK 攻击之后, 人们就把注意力集中到存在类似问题的 Diffe-Hellman 密钥交换算法上, 对弱密钥交换机制进行移植, 从而发布了新的漏洞

Logjam。这个攻击复制了 Freak 攻击的原理, 把 RSA_Export 替换成 DHE_Export。和 FREAK 不同的是服务器要愿意使用不安全的 DH 参数, 还需要缓存临时的 DH 密钥, 最后还要求客户端接受这些弱 DH 参数。所以最重要的原因就是浏览器能够接受不安全的 DH 参数。实际上, 攻破 1024 位参数的攻击者能够对 6.56%的 HTTPS 服务器进行被动攻击, 攻破了 10 个通用组的攻击者能够对约 10%的服务器进行攻击。对于 SSH, 攻破一个标准组, 能够使用被动攻击约 360 万的服务器, 而对于 IPsec 则能攻击超过 60%的服务器, 影响力非常巨大。

缓解这个攻击其实也不难, 首先禁用出口套件, 然后升级 DHE 的套件, 确保使用的是 2048 位, 而不是 1024 位, 或者完全禁用 DHE 算法。

SSL/TLS 协议设计是比较复杂的, 在实现过程中很难全部达到 RFC 的标准, 导致在实际应用中存在着很多的问题。随着近几年对 TLS 研究的深入, 很多以前隐藏在实现中的问题, 隐藏在库中的漏洞正在逐步被发现。通过不断的进行发掘、改进、升级, TLS 也越来越能够被大众更加放心的使用。由于 SSL/TLS 协议内容的复杂性, 存在着很多未知的变数。只有通过从在之前版本中的漏洞中不断地进行总结, 才能够逐步改善, 成为更完备, 更安全的通讯协议。通过总结 TLS 各版本中出现的问题, 在 TLS1.3 中做出了修改, 废除了类似于 RC4, CBC 加密等方式, 废弃了弱哈希算法的使用, 也采取了更有效的措施来防止降级。总之, 协议的发展是靠逐渐积累起来, 一点一点推进的, TLS1.3 将更具有更好的完备性和实用性。

4 证书相关问题

数字证书(简称证书)是基于公钥加密机制的一种数据结构, 可以绑定一个实体的身份和其拥有的公钥。证书可以为两个实体间的通信提供安全保障。SSL/TLS 也是通过证书来确保客户端和服务器进行安全, 保密的通信。服务器发送证书, 客户端(如浏览器)会检验证书的有效性。然而证书的安全性验证却一直是一个难题。目前针对证书安全性的研究非常广泛, 近几年有不少的相关研究, 值得学习。

证书相关研究

SSL 使用的证书是由 CA 签发的, 基于可信的第三方来保证通讯的安全。在以前只有少数几个证书颁发机构的时候, 可以通过保证自己的根证书的安全来保证颁发的证书的安全。但是, 随着目前市场上的证书颁发机构增多, 颁发的要求也参差不齐, 导致现实生活中有不少伪造的证书在被使用, 严重威胁到了网络通讯的安全。而且之前也出现过大型 CA 的根证书被攻破的情况, 颁发了不少看似合法的证书。在 2001 年就出现 Comodo 的其中一个注册机构被“完全入侵”的情况, 给 7 个网站签发了 9 张证书, 连 google.com 也受到了影响。现在随着 MD5, SHA-1 碰撞实验的成功, 也标志着伪造合法证书在理论上成为了可能。在推进 MD5, SHA-1 退出市场的间隙期, 也存在着不少安全隐患。对于不合法的证书的吊销并清除市场一直是比较困难的事情, 实施起来有难度。并且在吊销证书到证书被 CRL 或者 OCSP 更新, 也存在着空档期, 在缓存期间, 这些伪造的证书仍能被正常的使用。

近年来, 有不少对检测伪造证书的研究。Lin Shung 等人在文献[32]中, 利用 Flash Player 的插件设计和应用了一套针对全球知名网站 Facebook 的 SSL 中间人攻击检测, 分析了超过 3 百万以上条实时的 SSL 连接, 发现存在 0.2%左右的 SSL 连接是使用的伪造证书。其中有大量的防毒软件, 内容过滤器, 也有一些是恶意软件使用的不合法证书。之前存在大量的商业证书机构都或多或少被欺骗发布过不合法的证书, 这种情况下很多标准的浏览器也无法简单得区分出这些证书。但是更多的情况是用户忽略浏览器给出的警告信息, 继续访问危险网站。有一些研究人员认为应该忽略掉证书过期的问题, 他们表示证书过期是一件很常见的事情, 如果经常给用户提示, 会降低用户对警告的重视性。但是这样会导致一些伪造的过期证书被允许通过, 也会带来严重的安全隐患。作者对收集到的伪造证书进行了特征分析, 其中大部分的伪造证书非常小, 通常没有超过 1KB, 只有很少部分超过 5KB。而且伪造证书链比较短, 链长大部分为一, 通常是一些 self-singed 证书, 极少含有中间证书。通过分析伪造证书的 Subjects, 发现大部分采用的是合法的域名, 只有少部分使用的一些不相关的域名。其中大部分伪造证书都是选择好目标后, 预先生成, 而不是简单复制 Facebook 的合法证书。检查 Issuers 时, 发现大量的伪造证书来源于防毒软件, 防火墙, 广告软件和恶意软件。实际上, Carnavalet 在文献[33]中, 分析了存在于 TLS 代理中的这些用于过滤 TLS 流量的防毒软件和父进程控制应用程序, 并设计了一个集成框架来分析这样的客户端 TLS 代理。通过系统分析发现, 其中一些工具严重影响其主机上的 TLS 安全性, 还发现多个产品容易受到在中间人攻击的情况下完全服务器代理, 并且如果启用 TLS 过滤, 则还会更

可能被攻击。其中的一些工具还误导浏览器, 使其相信这样的 TLS 连接比实际更安全。通过检测发现, 在实际的网络中, 不安全的证书是确实存在的, 而且存在的数量还不少。

网络中存在大量的伪造证书, 导致受 SSL 保护的 HTTPS 看起来似乎也没那么安全了。证书安全是 SSL 协议安全的一个重要部分, 而客户端正确验证服务器端发送的证书的有效性起着至关重要的作用。Chad Brubaker[34]提出了一种大规模测试证书在 SSL 实际应用中的有效性的方法。对真实证书片段增添一些限制和延展, 利用“Frankencerts”合成不同的组合证书, 并进行“差分测试”。也即在实际 SSL 应用中, 发现一个证书被部分机构接收而被部分机构拒绝, 就可以作为差异筛选条件, 提取出来进行查询, 用来发现其中存在的漏洞和不足。差分测试使用 Frankencerts, 涵盖了 208 个区别点, 涉及的 SSL 应用包括 OpenSSL、NSS、CyaSSL、GnuTLS、PolarSSL 和 MatrixSSL 等。发现了在实际应用中存在着很多的问题, 如 MatrixSSL 会接收 X.509 v1 版本的证书, 使得所有使用 MatrixSSL 的应用都能被中间人攻击。攻击者只需要一个有效的 X.509 v1 证书就可以伪装成中间证书发布者, 发布一些假的证书, 并逃过 MatrixSSL 的检查。在 GnuTLS 中, 也存在 X.509 v1 证书的相关漏洞, 由于两个标志的错误匹配, 导致可以接收本地可信的 X.509 v1 证书, 即使是来自恶意服务器的证书。更严重的漏洞是来自于错检和漏检根证书对下层证书的限制。当然攻击能够实现的很重要的部分是用户忽略掉浏览器的提示, 坚持访问不安全的网站, 使得攻击者可以利用大量用户忽略证书过期的问题, 伪造相应证书来完成中间人攻击。

X.509 证书在实际应用中也存在不少的问题, 验证证书的有效性涉及到解析 ASN.1 的数据结构并解释内容信息, 过程比较复杂, 也很容易出错。但是 X.509 证书被大量使用, 从而具有不可替代性。为了应对 X.509 证书的一些结构和应用不足, Antoine 提出了 Cinderella[35]来改善 X.509 证书的使用。通过应用程序接收并验证证书完整性, 有效性和一致性, 取代之前的接收验证证书链的方式。这样省略证书产生更小的信息, 隐藏证书内容具有更强的隐私性, 嵌入一些额外的检测将具有更好的完整性。通过编写 X.509 模版来编写应用程序的新格式, 使用 Geppetto 加密编译器为该策略生成零知识可验证计算方案。并为 RSA-PKCS#1 签名和 ASN.1 解析开发新的 C 库, 提高加密可验证性能。在实际的应用中, 对 TLS 支持细粒度验证策略, 通过撤销检查和选择性披露证书内容, 能有效地将X.509证书转换为匿名证书。Cinderella 利用可验证计算弥补了现有 X.509 基础设施和现代密码学之间的差距, 并且能够使用户在较高级的应用程序中重用现有的证书链和签名机制, 与现有的基础设施完美集成, 不需要直接访问X.509签名密钥, 使其与现有的基于硬件的解决方案兼容。Cinderella 提高 X.509 身份验证和授权决策的灵活性, 表达能力和隐私权, 增加了“加密能力”。其中一个给定的验证策略可以将收集和验证最近的非撤销证据的责任移动到证书持有者, 使验证者的撤销检查更简单和更高效, 比较好的解决了 CRLs 和 OCSP 在线检查撤销状态的难题。

实际上, 对于证书的有效性问题, 研究人员提出了很多加强 SSL 安全的方法, 如采用 HTTP Strict Transport Security(HSTS)来遏制 SSL 剥离;使用 The Public Key Pinning Extension for HTTP(HPKP)允许网站使用 HTTP 标头指定自己的公钥, 并指示浏览器拒绝具有未知公钥的任何证书; 使用 TLS Origin-Bound Certificates(TLS-OBC)来封锁存在的很多中间人攻击插件; 采用 DNS-based Authentication of Named Entities(DANE), 依赖与DNSSEC阻止伪造修改 DNS 记录来使浏览器只接受一些特定的证书; Google 提出并发布的证书透明化项目, 可以实时检测伪造证书。实际上有很多的想法和建议提出来解决证书问题, 但是真正用于实践却比较难。Adam 和 Bates[36]提出了一些能及时解决目前存在的部分代码漏洞的方案, 引入了对 SSL 实现的轻量级改进的 CERTSHIM, 以程序透明的方式来防止SSL漏洞, 充当动态链接验证过程中的透明库。CERTSHIM 具有良好的可扩展性, 能融合 Convergence, DANE 和基于客户端的密钥固定等方法, 并且只带来极少的延迟。利用 CERTSHIM 解决了部分遗留在数据库中的危险代码问题, 也为后面的研究指明了方向, 更是促进了证书验证的发展。

鉴于证书的重要性, 很多研究机构投入了很多的精力来改善证书的现状。对证书的有效性验证一直是研究的重点, 近几年提出了一些比较有效的方案来缓解, 但是还不能完全解决证书的安全性问题。谷歌提出了白名单机制的证书透明化来改善证书的验证问题, 出于谷歌的影响力, 这个方案的普及对于保障证书的安全性将有显著的改善。要彻底解决证书安全性问题, 保障 SSL 通讯还需要进一步的努力, 研究工作大部分是从失败中学习, 总结经验并加以改善, 然后逐步完善成可靠的系统。

5 发展趋势和展望

近几年来, 随着 TLS 协议的逐步完善, 顺应发展的潮流, 工业界也开始大规模的部署 HTTPS, 吸引了更多的研究者, 引发对 TLS 协议更深入的研究。

近年来对 TLS 协议的研究主要是集中在设计和实现上。其中利用状态机分析协议实现的过程中发现了大量隐藏在协议内部的漏洞。从对协议设计上的填充攻击、CBC padding 攻击、lucky thirteen 攻击、poodle 攻击等可以得出, 选用好的加密方式的重要性。由于 TLS 协议的复杂性, 其中一些 RFC 标准并没有被很好的实现, 导致协议实现上的漏洞, 这在 Heartbleed 攻击的漏洞上有很好的印证。而在证书撤销和伪造证书的检测两个问题上, 一直都没有完美的解决方案, 所以很多研究人员也一直在这个方面进行努力。目前, TLS 协议的部署也存在着一些问题, 还需要投入更多的努力。理论是实践的基础, 在对理论进行实践的过程中往往还会遇到很多意料之外的问题, 这些都需要在实际的部署中才能够被发现。

TLS 自身的安全性是保障网络流量的安全传输的前提。总结近几年来对 TLS 1.3 以前版本协议的各项研究, 快速推进 TLS 1.3 版本的研究和部署至关重要。而且从 DROWN 攻击的总结中可以得出, 禁用老版本的 TLS 协议在工程实践部署上还仍需要努力。今后的发展中, 将会逐步禁用带有缺陷的版本, 如 SSL 2、SSL 3、TLS 1.0 等, 并转向大规模部署具有更高安全性的 TLS 1.3。证书方面, 工程部署上还有待提高。由于证书本身极其复杂, 而且存在大量错误配置的情况, 再加上实际环境中存在着大量的过期、伪造证书的威胁, 所以在证书方面的研究和部署上还需要更多的投入。

大量的移动应用在使用 TLS 协议来保障传输安全时也存在很多的问题。这些问题主要集中在使用的加密方式不安全, 对证书的部署存在缺陷, 对校验证书不完全等等。这样的结果导致虽然使用了 TLS 协议, 但却起不到应有的保护效果, 很容易被攻击者利用。这些问题主要是由于开发人员不熟悉 TLS 协议导致的, 所以给开发人员普及一些基本的 TLS 协议常识有助于开发出具有更高安全性的应用。

6 结束语

本文从协议设计、协议实现以及证书的角度对近几年的 TLS 部分研究进行总结。重点介绍了针对 CBC padding 模式设计的一系列攻击和 RC 4 的不安全性, 通过 Lucky Thirteen 攻击、POODLE 攻击提醒及时更新使用更安全的加密方式, 淘汰不太安全的老版本加密算法。Bleichenbacher 攻击、DROWN 攻击以及 POODLE 攻击更是提醒我们, 在工程部署上, 完全禁用 SSL 2 和限制 SSL 3 这些老版本协议的重要性。当然, 协议从一开始设计到最后完善, 需要一个发展的过程。最开始设计的协议存在一些逻辑上的漏洞, 比如三次握手攻击和重协商攻击。但对于这类漏洞, 迁移到新版本的 TLS 协议和完善 TLS1.3 的实现和部署是很好的解决方案。证书方面则是保障 TLS 安全过程的一个核心点, 在工程部署上存在的很多问题, 如证书撤销和伪造证书检测问题都亟待解决, 因此对证书方面的研究也需要进一步的投入。要解决好理论到实践的跨越, 一直都是很棘手的问题, 一个完善的系统往往需要从很多失败中汲取经验和教训, 不停的改进、更新, 最后才能在实用性上做得更好。

参考文献

[1] C. Meyer and J. Schwenk. Lessons learned from previous SSL/TLS attacks a brief chronology of attacks and weaknesses. IACR Cryptology ePrint Archive, 2013.

[2] T. Bellare and C. Allen. The Transport Layer Security (TLS) Protocol Version 1.0. RFC 2246, January 1999.

[3] T. Dierks, and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1, RFC 4346. April 2006.

[4] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246. August 2008.

[5] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, draft-ietf-tls-tls13-19. March 2017

[6] S.Vaudenay. Security flaws induced by CBC padding, application to SSL, IPSEC, WTLS… . In EUROCRYPT, 2002

[7] B. Kaliski. PKCS #5: Password-Based Cryptography Specification Version 2.0 .RFC 2898, Sep 2000

[8] B. Kaliski, and A. Rusch. PKCS #5: Password-Based Cryptography Specification Version 2.1. RFC 8018, Jan 2017

[9] N. AlFardan and K. G. Paterson. Lucky thirteen: breaking the TLS and DTLS record protocols. In IEEE, 2013

[10] B. Möller, T. Duong, and K. Kotowicz. This POODLE Bites: Exploiting The SSL 3.0 Fallback. Google, Sep 2014

[11] B. Moeller, A. Langley. TLS fallback signaling cipher suite value (SCSV) for preventing protocol downgrade attacks. RFC 7507, 2015

[12] D. Wagner and B. Schneier. Analysis of the SSLv3 protocol. In USENIX, 1996

[13] V. Klíma, O. Pokorný and T. Rosa2. Attacking RSA-based sessions in SSL/TLS. In CHES, 2003

[14] R. Bardou, R. Focardi, Y. Kawamoto. Efficient padding oracle at-

tacks on cryptographic hardware. INRIA, 2012

[15] C. Meyer, J. Somorovsky, E. Weiss. Revisting SSL/TLS implementations: New Bleichenbacher side channels and attacks. In USENIX, 2014

[16] N. Aviram, S. Schinzel, J. Somorovsky. DROWN: Breaking TLS using SSLv2. USENIX, 2016

[17] S. R. Fluhrer, I. Mantin and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In Cryptography, 2001

[18] N. AlFardan, R. Holloway and D. J. Bernstein. On the security of RC4 in TLS. In USENIX, Aug, 2013

[19] I. Mantin and A. Shamir. A practical attack on broadcast RC4. In in Computer Science, 2001

[20] C. Garman and K. G. Paterson. Attacks only get better: password recovery attacks against RC4 in TLS

[21] M. Vanhoef and F. Piessens. All your biases belong to us: breaking RC4 in WPA-TKIP and TLS. In USENIX, Aug, 2015

[22] I. Mantin. Bar-Minzva attack: breaking SSL with 13-year old RC4 weakness. In Blackhat, 2015

[23] Y. Sheffer,R. Holz and P. Saint-Andre. Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). IETF RFC 7525. May 2015

[24] K. Bhargavan, A. Delignat-Lavaud and C. Fournet. Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS. In IEEE Symposium on Security and Privacy (Oakland), 2014

[25] B. Beurdouche and K. Bhargavan, A. Delignat-Lavaud. A Messy State of the Union: Taming the Composite State Machines of TLS. In IEEE Symposium on Security and Privacy (Oakland), 2015

[26] K. Bhargavan, C. Brzuska and C. Fournet. Downgrade Resilience in Key-Exchange Protocols. In IEEE Symposium on Security and Privacy (Oakland), 2016

[27] K. Bhargavan and G. Leurent. Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH. In Network and Distributed System Security Symposium (NDSS), 2016

[28] K. Bhargavan and C. Fournet, M. Kohlweiss. Proving the TLS Handshake Secure (As It Is). In CRYPTO, 2014

[29] B. Beurdouche and K. Bhargavan, A. Delignat-Lavaud. A Messy State of the Union: Taming the Composite State Machines of TLS. In IEEE Symposium on Security and Privacy (Oakland), 2015

[30] B. Beurdouche, A. Delignat-Lavaud and N. Koberssi. FLEXTLS: A Tool for Testing TLS Implementations. In USENIX Workshop on Offensive Technologies (WOOT), 2015

[31] Z. Durumeric, J. Kasten and F. Li. The matter of Heartbleed. In ACM, 2014

[32] H. Lin-Shung, A. Rice, E. Ellingsen. Analyzing forged SSL certificates in the wild. In IEEE Symposium on Security and Privacy, 2014

[33] X. Carne Carnavalet and M. Mannan. Killed by proxy: analyzing Client-end TLS interception software. In NDSS, 2016

[34] C. Brubaker, S. Jana and B. Ray. Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations. IEEE Symposium on Security and Privacy, 2014

[35] A. Delignat-Lavaud, C. Fournet and M. Kohlweiss. Cinderella: Turning Shabby X.509 Certificates into Elegant Anonymous Credentials with the Magic of Verifiable Computation. In IEEE Symposium on Security and Privacy, 2016

[36] Bates and Adam M. Securing SSL Certificate Verification through Dynamic Linking. In ACM Conference on Computer and Communications Security (2014).

[37] Xinyu Li, Jing Xu and Zhenfeng Zhang. Multiple Handshakes Security of TLS 1.3 Candidates. In IEEE Symposium on Security and Privacy, 2016

韦俊琳 于2016年在华中科技大学大学生物技术专业获得本科学位，计算机专业获得计算机双学位。现在清华大学计算机专业攻读硕士学位。研究领域为网络空间安全。研究兴趣包括: TLS 协议安全, CDN, 防火墙以及二进制安全。email: weijl16@mails.tsinghua.edu.cn

段海新 清华大学网络科学与网络空间研究院研究员，网络与信息安全实验室(NISL)主任，CCERT 应急响应组负责人，加州大学 Berkeley 访问学者，蓝莲花战队(Blue-Lotus)联合创始人。长期从事网络安全相关的教学和研究，重点关注 DNS、CDN、PKI、Web、TLS 等基础设施和基础协议的安全，研究成果发表在 Oakland、USENIX Security、NDSS、SIGCOMM 等学术会议，在工业界引起广泛关注。

万涛 华为加拿大研究所研究员，长期从事互联网安全领域的研究和开发工作，重点关注 Web, mobile, CDN, Routing, SDN, 5G, PKI, 和 TLS 等领域的安全。在 NDSS, Oakland, USENIX Security 和 ACM TISSEC 等安全会议和安全杂志上发表过文章。