安全协议实现的安全性分析与评估

贾相堃[1]，苏璞睿[1]，闫佳[1]
（1 中国科学院软件研究所，可信计算与信息保障实验室）

[摘 要] 安全协议随着互联网的发展得到广泛应用。为了确保安全协议在真实网络环境中的实现满足安全需求，达到设计的安全目标，研究人员对安全协议的具体实现作了大量安全性分析与评估的研究工作。本文概要介绍了目前该领域的主要研究成果，提出了安全协议实现的分析与评估所要解决的问题，并从研究对象和研究方法两个维度对相关工作进行了分类和归纳总结，对当前研究存在的问题和未来发展方向进行了讨论。

[关键词] 安全协议实现，安全性评估

安全协议能够为日益频繁的互联网多方通信交互提供安全保障。虽然协议的可认证性、消息完整性、私密性、抗抵赖等安全特性经过严谨的形式建模和验证，但针对其实现的分析与评估存在严重不足。安全协议实现造成的漏洞层出不穷，对互联网安全构成了严重威胁，如国内多家电子商务平台曝出存在支付协议逻辑漏洞，可被利用进行“0 元支付”；多家银行手机客户端与中心服务器的通信协议实现存在缺陷导致用户信息泄露；尤其是近期，被广泛使用的 OpenSSL 开源库（SSL 协议的一个实现）频繁曝出高威胁漏洞。因此，针对安全协议实现进行安全性分析和评估具有很高的现实意义，也是学术研究的热点问题。

作为互联网通信标准的安全协议在设计阶段经过严格的形式化模型验证，在简化的抽象模型层次上可以确保协议的诸多安全特性，但是在具体的软件代码实现中会不可避免地引入难以控制和察觉的错误，尤其是开发人员在缺乏相关安全知识的情况下更容易造成协议实现的安全隐患。此外，还存在开发人员故意设置后门、部署系统和依赖的第三方函数库存在漏洞等情况，安全性更难以保证。因此，安全协议实现的安全性分析与评估主要解决如下问题：1、协议实现与协议模型规范一致性的检测，如协议实现的状态转换是否符合协议设计规范；2、协议模型没有详细规范而在具体实现中引入的安全问题，如协议只规定使用 RSA 算法，但是对协议安全性影响重大的算法密钥长度没有约定，从而造成巨大安全隐患；3、部署环境和依赖的第三方函数库存在的漏洞对协议实现安全性的影响，如 Heartbleed 内存越界读漏洞会造成基于 OpenSSL 库的服务器的根证书私钥泄露。针对上述安全问题，目前在安全协议的安全性分析与评估过程中，主要以协议实现的源代码或者协议实现的二进制程序为主要分析对象，涉及协议建模和模型验证、软件测试、软件动态分析等关键技术。其中基于源代码的研究主要以形式化分析和模型检验方法为主，能够进行协议实现的合规性的检测；基于二进制程序的研究主要以软件测试和二进制程序分析方法为主，能够发现协议规范无法对具体实现进行细粒度描述和约定而造成的大量其他安全问题。

基于形式化分析和模型检验等方法进行协议实现安全性分析与评估，通常以程序源代码、协议的模型化描述为分析对象。Csur Project[1] 利用信任断言把 C 语言程序和 Dolev-Yao 模型[2] 下的消息进行关联，再利用定理证明器，证明断言所生成语义实现的正确性。Chaki 等[3] 利用模型检验技术验证了 OpenSSL 协议实现的安全性，形成 ASPIER 系统。Bhargavan 的团队[4] 将 F# 语言实现的协议转化为 applied-π 演算，并用 ProVerif[5] 进行安全性验证，而后综合运用 ProVerif 和 CryptoVerif[6]，对 TLS 协议进行了符号证明和计算证明[7]。最近该团队将协议的逻辑不变式、密码算法的前置 / 后置条件表达为 RCF 逻辑公式，通过类型检查工具 F7 进行类型检查[8]。Dupressoir[9] 则是利用定理证明器 VCC[10] 对 C 代码进行验证，扩展了方法的通用性。随着计算能力的快速提升，目前基于各类形式化分析方法的协议安全性验证已经能够对较为复杂的协议模型进行较为完备的模型安全性检验，但是仍然无法满足规模日益庞大的协议实现代码的分析要求。

软件测试也是发现协议实现存在的各类软件漏洞和安全缺陷的重要方法。2004 年，Arunchandar Vasan 等人[11] 采用软件测试中的错误注入（fault injection）方法自动生成错误的协议数据单元（PDUs），通过分析不同协议实现的返回信息，评估协议实现的鲁棒性。Guoqiang Shu 等人[12] 在黑盒测试中引入机器学习方法中的 L* 算法，将其与符号参数扩展的有限状态机（FSM）相结合，验证了 Dolev-

作者简介：苏璞睿，1976 年生，中国科学院软件研究所研究员、博士生导师，可信计算与信息保障实验室副主任，中国密码学会安全协议专业委员会委员。现主要从事网络安全与信息对抗方面的研究，主持了国家自然科学基金、国家 863 计划、国家科技支撑计划等十余项国家级科研项目。

Yao 攻击模型下安全协议的机密性。Yating Hsu[13] 在该方法的基础上，进一步减小了 FSM 的状态空间，提高了基于协议行为模型的模糊测试（Fuzz）的覆盖率和有效性。SecFuzz[14] 则在黑盒模糊测试中通过动态内存分析进行安全缺陷的检测，成功提高了针对安全协议的模糊测试的效率。软件测试是目前针对安全协议实现的最为广泛使用的分析和评估方法，能够不依赖于源代码甚至协议模型进行协议实现合规性检测，但是与通常软件测试面对的问题类似，仍然要解决测试输入空间太大的问题。

基于二进制程序的软件分析方法同样是协议实现安全性分析的重要手段。Udrea 等人 [15] 开发的二进制代码静态分析工具 PISTACHIO，通过源代码和 RFC 规范进行基于规则的比较来实施安全性评估。此外，针对大量无法获得源代码的商业协议或商用软件，逆向分析、动态分析成为分析的必要手段和前提条件。Polyglot[16]、Discovery[17]、AutoFormat[18]、Reformat[19]、Dispatcher[20] 等系统通过动态二进制代码分析，自动提取协议的语法结构和语义信息，在网络程序漏洞挖掘、协议实现的安全性分析与评估中发挥了巨大的作用，也是协议实现自动化 Fuzz 的基础。例如 AUTHSCAN[21] 综合利用白盒程序分析和黑盒模糊测试，从协议实现中自动提取认证协议规范，并用 ProVerif 对形成的 TML（Target Model Language）模型进行验证以发现安全问题。面对日益增多的商业闭源软件和其对应的安全协议实现，只有基于二进制程序的程序分析方法才能对其进行有效分析和评估，该方法是协议实现安全性分析和评估的重要发展方向。

加密算法、随机数生成算法等密码运算是安全协议设计不可缺少的部分，在工程实践中以密码库的形式为安全协议实现提供基本功能。但是由于安全问题的复杂性，在协议实现中针对密码库的使用一直存在着隐患。2004 年 EuroCrypt 大会上 Nguyen[22] 通过研究 GPG v1.2.3，发现了一系列密码误用漏洞。Thai Duong 等人 [23] 针对 ASP.Net 框架，详细地介绍了其中存在的不认证加密等漏洞及其攻击方法。Lin-shung Huang[24] 则在对 TLS 服务器的研究中，指出目前存在大量的弱配置现象。在 Android 移动平台上，也存在相同的问题。Manuel Egele 等人 [25] 开发了基于 Androguard[26] 的工具 CryptoLink，分析了 Google Play 上的应用，指出 88% 涉及密码运算的应用都存在错误。Vasant Tendulkar[27] 也指出 Android 平台上 SSL 实现的漏洞，并给出了开发过程中包设置的建议。David lazar[28] 总结了 CVE 信息库中密码实现相关漏洞的机理，进一步证实安全协议实现的复杂性。安全协议实现对程序研发人员的安全编程要求非常高，针对协议实现中密码运算等安全模块的配置和使用的安全性分析和研究也是一个重要的研究课题。

安全协议实现的安全性分析与评估研究已经在合规性检验、安全性验证以及漏洞挖掘方面取得显著进展，但目前的分析方法还难以应对日益增长的协议实现代码规模。大量安全协议实现在没有得到充分有效验证的情况下投入使用，对互联网安全造成巨大的威胁。即使是 SSL/TLS 这样被广泛应用和分析的安全协议，2014 年 Chad Brubaker[29] 和 Zakir Durumeric[30] 分别进行的大规模网络测量和测试表明，其实现依然存在严重的安全问题。因此，该领域仍然会是未来研究的重点。

参考文献：

[1] Goubault-Larrecq J, Parrennes F. Cryptographic protocol analysis on real C code[C]//Verification, Model Checking, and Abstract Interpretation. Springer Berlin Heidelberg, 2005: 363-379.

[2] Dolev D, Yao A C. On the security of public key protocols[J]. Information Theory, IEEE Transactions on, 1983, 29(2): 198-208.

[3] Chaki S, Datta A. ASPIER: An automated framework for verifying security protocol implementations[C]//Computer Security Foundations Symposium, 2009. CSF'09. 22nd IEEE. IEEE, 2009: 172-185.

[4] Bhargavan K, Fournet C, Gordon A D, et al. Verified interoperable implementations of security protocols[J]. ACM Transactions on Programming Languages and Systems (TOPLAS), 2008, 31(1): 5.

[5] Blanchet B. An efficient cryptographic protocol verifier based on Prolog rules[C]//Computer Security Foundations Workshop, IEEE. IEEE Computer Society, 2001: 0082-0082.

[6] Blanchet B. A computationally sound mechanized prover for security protocols[J]. Dependable and Secure Computing, IEEE Transactions on, 2008, 5(4): 193-207.

[7] Bhargavan K, Fournet C, Corin R, et al. Cryptographically verified implementations for TLS[C]//Proceedings of the 15th ACM conference on Computer and communications security. ACM, 2008: 459-468.

[8] Bhargavan K, Fournet C, Gordon A D. Modular verification of security protocol code by typing[C]//ACM Sigplan Notices. ACM, 2010, 45(1): 445-456.

[9] Dupressoir F, Gordon A D, Jurjens J, et al. Guiding a general-purpose C verifier to prove cryptographic protocols[C]//Computer Security Foundations Symposium (CSF), 2011 IEEE 24th. IEEE, 2011: 3-17.

[10] Cohen E, Dahlweid M, Hillebrand M, et al. VCC: A practical system for verifying concurrent C[M]//Theorem Proving in Higher Order Logics. Springer Berlin Heidelberg, 2009: 23-

42.

[11] Vasan A, Memon A M. Aspire: Automated systematic protocol implementation robustness evaluation[C]//Software Engineering Conference, 2004. Proceedings. 2004 Australian. IEEE, 2004: 241–250.

[12] Shu G, Lee D. Testing security properties of protocol implementations–a machine learning based approach[C]// Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on. IEEE, 2007: 25–25.

[13] Hsu Y, Shu G, Lee D. A model–based approach to security flaw detection of network protocol implementations[C]// Network Protocols, 2008. ICNP 2008. IEEE International Conference on. IEEE, 2008: 114–123.

[14] Tsankov P, Dashti M T, Basin D. SECFUZZ: Fuzz–testing security protocols[C]//Automation of Software Test (AST), 2012 7th International Workshop on. IEEE, 2012: 1–7.

[15] Udrea, O., C. Lumezanu and J.S. Foster. Rule–based static analysis of network protocol implementations. in Proceedings of the 15th conference on USENIX Security Symposium – Volume 15. 2006. Vancouver, B.C., Canada: USENIX Association.

[16] Caballero J, Yin H, Liang Z, et al. Polyglot: Automatic extraction of protocol message format using dynamic binary analysis[C]//Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2007: 317–329.

[17] Cui W, Kannan J, Wang H J. Discoverer: Automatic Protocol Reverse Engineering from Network Traces[C]// USENIX Security. 2007: 199–212.

[18] Lin Z, Jiang X, Xu D, et al. Automatic Protocol Format Reverse Engineering through Context–Aware Monitored Execution[C]//NDSS. 2008, 8: 1–15.

[19] Wang Z, Jiang X, Cui W, et al. ReFormat: Automatic reverse engineering of encrypted messages[M]//Computer Security – ESORICS 2009. Springer Berlin Heidelberg, 2009: 200–215.

[20] Caballero J, Poosankam P, Kreibich C, et al. Dispatcher: Enabling active botnet infiltration using automatic protocol reverse–engineering[C]//Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009: 621–634.

[21] Bai G, Lei J, Meng G, et al. AUTHSCAN: Automatic Extraction of Web Authentication Protocols from Implementations[C]//NDSS. 2013.

[22] Nguyen P Q. Can we trust cryptographic software? Cryptographic flaws in GNU Privacy Guard v1. 2.3[C]// Advances in Cryptology–EUROCRYPT 2004. Springer Berlin Heidelberg, 2004: 555–570.

[23] Duong T, Rizzo J. Cryptography in the web: The case of cryptographic design flaws in asp. net[C]//Security and Privacy (SP), 2011 IEEE Symposium on. IEEE, 2011: 481–489.

[24] Huang L S, Adhikarla S, Boneh D, et al. An Experimental Study of TLS Forward Secrecy Deployments[J].

[25] Egele M, Brumley D, Fratantonio Y, et al. An empirical study of cryptographic misuse in android applications[C]// Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013: 73–84.

[26] Desnos A. Androguard: Reverse engineering, malware and goodware analysis of android applications... and more (ninja!) [J].

[27] Tendulkar V, Enck W. An Application Package Configuration Approach to Mitigating Android SSL Vulnerabilities[J].

[28] Lazar D, Chen H, Wang X, et al. Why does cryptographic software fail?: a case study and open problems[C]// Proceedings of 5th Asia–Pacific Workshop on Systems. ACM, 2014: 7.

[29] Brubaker C, Jana S, Ray B, et al. Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations[J].

[30] Durumeric Z, Kasten J, Adrian D, et al. The matter of Heartbleed[C]//ACM Internet Measurement Conference (IMC). 2014.