文章编号：2096-1472(2017)-01-55-05

基于OAuth2.0协议的安全授权模型研究

王婷婷[1]，赵松泽[2]
（1.北京自如信息科技有限公司，北京 100873；
2.北京师范大学附属实验中学，北京 100032）

摘　要：本文在OAuth2.0授权码模型的基础上做出改进，采用HLPSL语言对授权码模型进行形式化建模，建立OAuth2.0协议授权码模型形式化模型，找到授权码模型出现安全漏洞的根本原因是客户端凭证可以被攻击者窃取。结合惰性无限状态方法和惰性攻击者优化方法对形式化模型分析和验证。提出OAuth2.0安全授权码模型，并分析和验证其在理论上无安全漏洞。通过的研究，本文可以提供一套安全的OAuth2.0授权协议模型，对目前安全要求高的开放平台的授权是有指导意义的。

关键词：OAuth2.0；安全；授权码模型；形式化
中图分类号：TP309　　**文献标识码**：A

The Research of the Security Authorization Model Based on OAuth2.0 Protocol

WANG Tingting,ZHAO Songze
(1.*Beijing Ziru Information Technology Co.Ltd.,Beijing* 100873,*China*;
2.*The Experimental High School Attached to Beijing Normal University,Beijing* 100032,*China*)

Abstract: Based on the OAuth2.0 protocol authorization code model,the paper adopts HLPSL to create a formal model on the authentication code model.The root cause of the security vulnerability on the authorization code model is that attackers can steal the client certificate.Through the analysis and verification of formal model by using the method of infinite state of inertia and the optimization method of inert attackers,the paper proposes the OAuth2.0 security authorization code model,and analyzes and verifies its zero security vulnerability in theory.Through the research,this paper provides a secure OAuth2.0 authorization protocol model,which is a guide for the authorization of the open platform with high security requirements.

Keywords:OAuth2.0;security;the authorization code model;formal

1 引言(Introduction)

近些年来，随着OAuth2.0协议的广泛应用，其安全性受到了人们的重点关注。2012至2014年期间，腾讯、新浪微博、Twitter、Facebook、Google等国内外大量知名网站因使用OAuth授权的开放平台而受到安全威胁，曾数次紧急公布修复方案。目前最主流的开放平台授权协议OAuth的授权模型的安全性一直受到企业应用者的关注，实质上是保证不同角色在交互过程中提供安全可信任的服务。OAuth2.0授权码模型是功能最完整、流程最严密的授权模式，也正因此授权码模型被广泛应用，但是频频曝出安全漏洞问题。从20世纪80年代以来，国内外出现了很多对安全协议进行形式化分析的方法，根据方法的性质不同，也出现很多不同的分类方法，本文根据形式化分析方法的表达能力分将其大致分为基于模型检测的安全协议分析方法、基于模型检测的安全协议分析方法和基于证明的安全协议分析方法。

OAuth在“客户端”与“服务提供商”之间，设置了一个授权层(authorization layer)。“客户端”不能直接登录“服务提供商”，只能登录授权层，以此将用户与客户端区分开来。“客户端”登录授权层所用的令牌(token)，与用户的密码不同。用户可以在登录的时候，指定授权层令牌的权限范围和有效期。“客户端”登录授权层以后，“服务提供商”根据令牌的权限范围和有效期，向“客户端”开放用户储存的资料。

2 基于模型检测的安全协议形式化分析方法研究(Research on formal analysis method of security protocol based on model detection)

基于模型检测的安全协议分析和验证方法的基本思路是利用有限状态机理论，通过定义状态集合及状态迁移函数为安全协议建立模型；通过穷尽搜索状态空间来判断一些特殊的状态是否可达，或者是否可以生成一条特殊的状态转移路径，并以此检测该模型是否具备期望的安全性质。通用的做法是把安全协议看成一个分布式系统，单个主体涉及的协议执行部分称为局部状态，所有局部状态构成系统的全局状态；协议执行过程中，主体收发消息的动作会引起局部状态的改变，进而也引起全局状态的改变；在安全协议全局状态上定义安全属性或不变关系，则安全协议是否满足安全目标

定价与系统可达的每个全局状态上安全属性或不变关系是否都能得到满足。

安全协议的模型检测分析方法取得了很大的成功，其优点是自动化程度高，可以借助自动分析工具来完成分析过程而不需要用户的参与，并且安全协议存在缺陷时能够自动生成相应的攻击实例。模型检测方法是基于Dolev−Yao模型。

本文采用基于模型检测的安全协议形式化分析验证方法，并使用OFMC形式化检测工具，OFMC是AVISPA平台的主要检测工具，也被称为On−the−Fly模型检测器，基于惰性攻击者思想，结合两种分析验证方法来优化和证明协议的正确性和完整性，第一种是使用惰性数据类型建立高效的无限状态空间；第二种是将符号技术和惰性Dolev−Yao攻击者模型最优化相结合的方法。

2.1 惰性无限状态方法

惰性无限状态方法的关键思想是明确形式化的无限树结构，作为惰性编程语言的一种数据类型。这样会产生一个有限的和可计算的模型，并且可以在需求驱动的方式下，用来生成树的任意前缀。可以通过搜索无限状态树种的一个攻击状态来找到一种攻击。On−the−Fly模型检测器OFMC使用迭代加深的方式搜索该无限状态树，当发现攻击，OFMC返回攻击路径，导致攻击状态的信息交换的顺序。所以就是一个协议不安全的半决策程序：当攻击存在的时候程序终止。此外，当采用惰性攻击者处理攻击者产生的无限消息集合时，搜索程序会终止有限的几个会话。

惰性无限状态方法有几个优点，它通过启发式方法和其他搜索，并从自身开始搜索减少搜索过程来分离协议的语义。通过迁移系统产生的无限状态树给定了协议的语义，启发式方法可以看成是树型结构的转化器，输入是无限状态树，输出是相对小或更加受限的树型结构，然后搜索该结果树。尽管语义、启发式方法和搜索都是独立的阐述，但是惰性无限状态方法以高效、模型驱动的方式将他们协同在一起。此外，有很多高效的编译器，使用惰性函数式编程语言例如Haskell、OFMC编译器就是使用这个语言。

2.2 惰性攻击者优化方法

惰性攻击者是一种优化方法，大大减少了搜索树的空间。方法使用符号表示来避免明确例举Dolev−Yao攻击者生成的可能信息，通过存储和操作必要生成的约束。表示方式的判定是需求驱动的方式，因此攻击者被称为是惰性的。

(1)约束

当例举攻击者可以发送的所有消息时，Dolev−Yao攻击者引出搜索树的一个巨大分支。惰性攻击者方法认为消息的某一部分的真实值通常与接受者是无关的。因此，当接收者不需要进一步分析特定信息部分的值时，在搜索期间可以延迟决定关于攻击者实际选择的部分的哪个值用变量来替换，并且记录攻击者能生成消息的约束。我们表达这一信息使用形式化约束from(T,IK),T是攻击者通过已知的信息IK(入侵者知识)生成的表达式集合。

定义 语义约束from(T，IK)是满足约束中变量替换为σ的集合，即

[from(T,IK)]={σ|ground(σ)∧ground(Tσ∪IKσ)∧Tσ⊆DY(IK σ)}

如果T⊆V，称约束from(T,IK)是简单约束，并且记为(from(T,IK))。

约束集是一组有限的约束，并且它的语义是其元素的语义的交集，即[{c1,..,cn }]=⊓ni=1[ci]，如果[C]!=∅，约束集C是可满足的。如果所有的约束都是简单约束，则约束集C是简单约束集，并且记为simple(C)。

(2)约束规则

惰性攻击者方法的核心是规约给定约束集为等价的约束集，要么是不可满足的要么是简单约束(每个简单的约束集是可满足的)。这种规约的执行使用下面的生成和分析规则，这些规则描述约束集的可能转换。

$$\frac{from(m_1 \cup m_2 \cup T, IK) \cup C, \sigma}{from(\langle m_1, m_2 \rangle \cup T, IK) \cup C, \sigma}\ G^{l}_{\text{pair}},$$

$$\frac{from(m_1 \cup m_2 \cup T, IK) \cup C, \sigma}{from(\{|m_2|\}_{m_1} \cup T, IK) \cup C, \sigma}\ G^{l}_{\text{scrypt}},$$

$$\frac{from(m_1 \cup m_2 \cup T, IK) \cup C, \sigma}{from(\{m_2\}_{m_1} \cup T, IK) \cup C, \sigma}\ G^{l}_{\text{crypt}},$$

$$\frac{from(m_1 \cup m_2 \cup T, IK) \cup C, \sigma}{from(m_1(m_2) \cup T, IK) \cup C, \sigma}\ G^{l}_{\text{apply}},$$

$$\frac{(from(T, m_2 \cup IK) \cup C)\tau, \sigma\tau}{from(m_1 \cup T, m_2 \cup IK) \cup C, \sigma}\ G^{l}_{\text{unif}}\ (\tau = mgu(m_1, m_2),\ m_1 \notin \mathcal{V}),$$

$$\frac{from(T, m_1 \cup m_2 \cup \langle m_1, m_2 \rangle \cup IK) \cup C, \sigma}{from(T, \langle m_1, m_2 \rangle \cup IK) \cup C, \sigma}\ A^{l}_{\text{pair}}\ (\{m_1, m_2\} \setminus IK \neq \emptyset),$$

$$\frac{from(m_1, IK) \cup from(T, m_2 \cup \{|m_2|\}_{m_1} \cup IK) \cup C, \sigma}{from(T, \{|m_2|\}_{m_1} \cup IK) \cup C, \sigma}\ A^{l}_{\text{scrypt}}\ (m_2 \notin IK),$$

$$\frac{from(m_1^{-1}, IK) \cup from(T, m_2 \cup \{m_2\}_{m_1} \cup IK) \cup C, \sigma}{from(T, \{m_2\}_{m_1} \cup IK) \cup C, \sigma}\ A^{l}_{\text{crypt}}\ (m_2 \notin IK),$$

$$\frac{from(m_1, IK) \cup from(T, m_2 \cup \{m_2\}_{m_1} \cup IK) \cup C, \sigma}{from(T, \{m_2\}_{m_1^{-1}} \cup IK) \cup C, \sigma}\ A^{l}_{\text{crypt}^{-1}}\ (m_2 \notin IK).$$

表明这种规约并不会改变集合的结果，大概说来，[C]=[Red(C)]，相关的一类约束集C。

生成规则或分析规则r的形式$\frac{C', \sigma'}{C, \sigma}\ r$，C和C′规约集合和σ和σ′替换。它表达的是(C，σ)可以衍生出(C′，σ′)，表示为(C，σ) r(C′，σ′)。

综上所述，我们总结为检查协议是否是安全的，通过生成可达惰性状态，过滤攻击状态和约束是可满足的。

3 OAuth2.0授权码模型形式化分析与验证(Formal analysis and verification of OAuth2.0 authorization code model)

3.1 OAuth2.0授权码模型

授权码模型包含资源拥有者、用户代理、第三方应用、授权服务器和资源服务器这五个角色。通常，用户代理就是用户所使用的浏览器，授权服务器和资源服务器都是服务器

提供方的内部服务器机群，因此站在整体架构的角度来观察服务提供方，仅仅存在一系列的服务程序接口，通过这些接口，第三方应用可以请求授权服务器的授权许可，也可以在获取授权许可后去申请资源，因此本文将授权服务器和资源服务器进行合并，隐藏了服务提供方的服务器机群之间的交互细节，对外部只提供授权服务器这一个角色。

当第三方应用希望获取服务提供方的资源时，需要在服务提供方进行信息的注册，当服务提供方自动化审核通过第三方应用的注册信息后，服务提供方会生成与这个第三方应用相关联的应用ID和应用密钥，并将这一认证信息发回给第三方应用，第三方应用将认证信息保存在自己的服务器中的数据库中。当第三方应用完成预注册请求后，便可以开始通过授权码模型向服务提供方发出授权请求和访问共享资源请求。

授权码模型的架构如下图所示，其中包括资源拥有者(用户)、用户代理、第三方应用和授权服务器四个角色。根据授权码模型在各个阶段所完成的任务不同。可将其划分为四个阶段：请求授权阶段、授权码与令牌的交换阶段、访问共享资源阶段、令牌更新阶段。

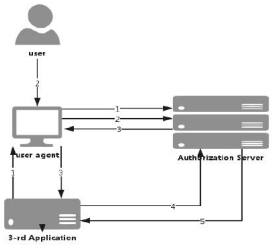

图1 授权码模型架构图

Fig.1 Authenticator model frame

由架构图可以看出用户代理主要进行消息转发，即将第三方发送的请求通过信道1转发给授权服务器，将用户的认证信息通过信道2转发给授权服务器，以及将授权服务器的授权应答通过信道3转发回第三方应用。当第三方应用获取授权请求后，之后的信息传输只通过信道4和信道5在第三方应用和授权服务器这两者之间进行传输，由于不再经过用户代理进行信息传输，这部分信息对于用户来讲是透明的，因此授权码模型也被认为最安全的模型。

根据OAuth2.0规范内关于授权码模型的描述，本文给出了描述这个模型时序图，如图2所示。

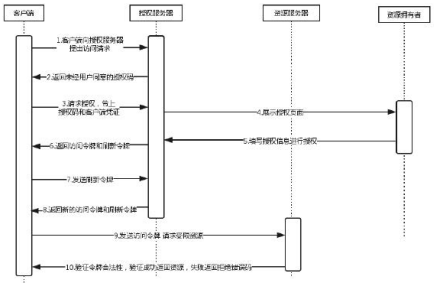

图2 OAuth2.0协议流程时序图

Fig.2 The OAuth2.0 protocol sequence chart

3.2 形式化建模的总体方案

采用AVISPA平台形式化语言HLPSL描述的OAuth2.0授权码形式化建模的总体方案可以分解为三个步骤：

(1)抽象出基本角色，并定义角色；

(2)描述变迁；

(3)描述安全目标。

HLPSL是基于角色的一种语言，我们需要根据OAuth2.0协议的行为定义不同角色，然后针对每个角色阐明他们的特定数据以及行为，角色之间通过消息的传递来进行交互，变迁就是指这里的消息的传递。

模型由很多角色(Role)及目标(Goal)组成，每个角色有输入参数，局部变量以及行为。授权码模型有四个角色参与，分别是第三方客户端、资源拥有者、授权服务器、资源服务器。根据授权码形式化需求的描述，秘密性就是要保证客户端访问凭证、授权码、访问令牌和刷新令牌的私密；认证性更加关注实体认证，保证客户端是个合法的参与者。

完整的模型除了参与协议的不同角色之外，还含有一些特殊的角色：会话(session)，指构成会话流的组合角色，这些会话可以调用不同的角色，构建完整的会话流；环境(environment)定义了模型的主体，包括这个环境中的基本变量、角色、攻击者知识、正常会话及攻击会话。

3.3 建立OAuth2.0授权码形式化模型

根据上一章节给出的AVISPA形式化模型架构，本小节将通过HLPSL语言搭建OAuth2.0协议的授权码模型。我们用形式化的方法描述授权码模型的流程，协议如下所示：

C、O、A、S分别是四个参与的主体。其中C表示第三方客户端(client)，O表示资源的拥有者(resource owner)，A表示授权服务器(authorization server)，S表示资源服务器(resource server)。

Ka、Kac代表对称密钥，仅在协议的主体之间共享。其中Koa表示资源拥有者(O)和授权服务器(A)之间的通信密钥，Kca表示客户端(C)和授权服务器(A)之间的通信密钥。

Ka、Kc是由主体创建的数据，Ka是授权服务器创建的授权许可，Kc是客户端维护的客户端凭证。

Kt(Token)、Kr(Refresh)表示授权服务器生成的访问令牌和刷新令牌。

T表示资源服务器上受保护的资源。

整个流程的形式化描述表示如下：

(1)C->A:{Napp}_Kac

(2)A->C:{Ka}_Kac

(3)C->A:{Ka,Kc}_Kac

(4)A->O:{Ka,Kc}_Kao

(5)O->A:{Ko}_Kao

(6)A->C:{Kt,Kr}_Kc

(7)C->A:{Kr}_Kr

(8)A->C:{Kt,Kr}_Kc

(9)C->S:{Kc}_Kt

(10)S->C:{T} Kc

3.4 OAuth2.0授权码模型验证与分析

使用AVISPA平台的验证工具Span来进行协议的安全测试，通过ofmc后台分析对该模型进行了攻击验证,最后运行的结果如图3所示。

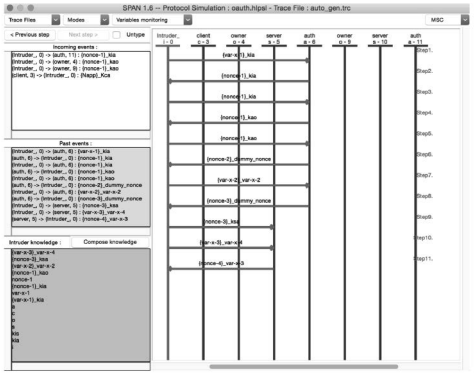

图3 使用 AVISPA 对 OAuth2.0 验证结果

Fig.3 OAuth2.0 verification results through AVISPA

(1)(Intruder_,0)->(auth,6):{var-x-1}_kia:攻击者向授权服务器发起授权请求。

(2)(auth,6)->(Intruder_,0):{nonce-1}_kia:授权服务器返回未授权的授权码。

(3)(Intruder_,0)->(auth,6):{nonce-1}_kia:攻击者向授权服务器发送授权码和通过非法途径获取的客户端凭证。

(4)(auth,6)->(Intruder_,0):{nonce-1}_kao:授权服务器引导用户授。

(5)(auth,6)->(Intruder_,0):{nonce-1}_kao:用户以为是正常的第三方客户端，成功授权。

(6)(auth,6)->(Intruder_,0):{nonce-1}_kao:授权服务器将得到的授权信息发送给盗取了客户端凭证的攻击者。

(7)(auth,6)->(Intruder_,0):{nonce-1}_kao:攻击者向授权服务器发送刷新令牌请求。

(8)(auth,6)->(Intruder_,0):{nonce-1}_kao:授权务器更新访问令牌和刷新令牌，又发给了攻击者。

(9)(Intruder_,0)->(server,5):{nonce-3}_ksa:授权服务器抄送资源服务器相关的访问令牌。

(10)(Intruder_,0)->(server,5):{var-x-3}_var-x-4:攻击者通过持有的访问令牌向资源服务器发起资源访问请求。

(11)(Intruder_,0)->(server,5):{var-x-3}_var-x-4:服务器验证访问令牌合法后返回了受限资源。

(12)i->(i,17):T(6):攻击者成功得到了受限资源。

如果第三方客户端的客户端凭证被攻击者通过非法手段获取,就可以通过相应的攻击模式，骗取资源拥有者的授权，从而获取受限的资源。

4 安全授权码模型的建立(Establishment of security authorization code model)

4.1 安全授权码模型

安全授权码模型的建立是参考授权码模型存在的安全漏洞，即客户端凭证会被攻击者窃取，导致的授权服务器无法判断客户端的合法性，误认为是合法的客户端，并允许其访问用户资源，本章通过改进客户端认证流程，假设客户端与其后台服务器是独立的，增加客户端后台服务器角色(Client's Back-End Server)，每个客户端向授权服务器请求授权之前，需要向客户端后台服务器请求客户端编号(Device ID)，客户端编号是由客户端后台服务器来唯一确定的，并且带有时效性，过期会失效，攻击者是不能获取客户端编号的；当客户端向授权服务器请求授权时，不仅要携带授权码和客户端凭证，还需要携带客户端编号(Device ID)；授权服务器验证授权码和客户端凭证的合法性之后，识别发出请求的客户端，并向客户端后台服务器(Client' Back-End Server)请求设备编号(Device ID)，授权服务器接收到客户端后台服务器返回的Device ID，与客户端请求token时携带的Device ID对比，设备编号相同认为请求的客户端是合法客户端，会返回客户端token和刷新token，不相同则认为是非法客户端，拒绝客户端的请求。安全授权码模型如图4所示。

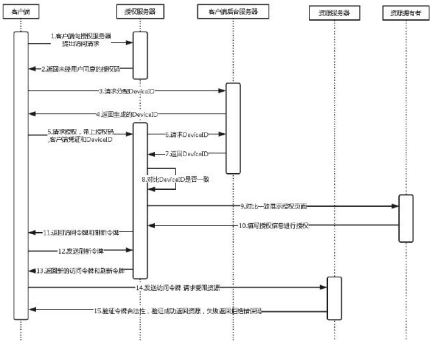

图4 安全授权码模型

Fig.4 Security authorization code model

我们用一种形式化的方法描述上图中的流程,协议如下所示：

C、O、A、S、B分别是五个参与的主体。其中C表示第三方客户端(client),O表示资源的拥有者(resource owner),A表示授权服务器(authorization server),S表示资源服务器(resource server)，新增B表示应用的后台服务器(background server)。

Kao、Kac、Kcb、Kab代表对称密钥，仅在协议的主体之间共享。其中Kao表示资源拥有者(O)和授权服务器(A)之间的通信密钥，Kca表示客户端(C)和授权服务器(A)之间的通信密钥。Kcb表示客户端(C)和应用后台服务器(B)之间的通信密钥，

Kab表示验证服务器(A)和后台服务器(B)之间的通信密钥。

Ka,Kc,Kdid是由主体创建的数据，Ka是授权服务器创建的授权许可，Kc是客户端维护的客户端凭证,Kdid是应用后台服务器授予客户端的授权凭证。

Kt(Token),Kr(Refresh)表示授权服务器生成的访问令牌和刷新令牌。

T表示资源服务器上受保护的资源。整个流程的形式化描述表示如下：

(1)C->A:{Napp}_Kac

(2)A->C:{Ka}_Kac

(3)C->B:{Kdid}_Kcb

(4)B->C:{Kdid}_Kcb

(5)B->A:{Kdid}_Kab

(6)C->A:{Ka,Kc}_Kdid

(7)A->O:{Ka,Kc}_Kao

(8)O->A:{Ko}_Kao

(9)A->C:{Kt,Kr}_Kc

(10)C->A:{Kr}_Kr

(11)A->C:{Kt,Kr}_Kc

(12)C->S:{Kc}_Kt

(13)S->C:{T} Kc

4.2 验证与分析

我们使用AVISPA平台的验证工具Span来进行改进后的协议的安全测试，通过ofmc后台分析对该模型进行了攻击验证,最后运行的结果如图5所示。

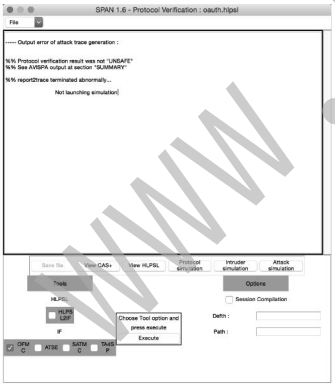

图5 使用AVISPA改进的OAuth2.0验证结果

Fig.5 Improved OAuth2.0 verification results through AVISPA

工具提示“result was not unsafe”表示该工具没有找到攻击模式可以破解改进的授权码模型。说明该模型在一定程度上是安全的。

5 结论(Conclusion)

本文首先对OAuth2.0功能最完整、流程最严密的授权码模型，采用严格的数学方法——形式化方法进行分析验证，分析验证过程中使用HLPSL语言对授权码模型进行了十分完善的形式化建模，使用安全协议自动化验证工具对协议进行分析，对运行结果分析影响OAuth协议安全的关键因素；然后对授权模型进行改进，研究并建立安全授权码模型，并对其进行形式化分析；最后对授权码模型和安全授权码模型对比分析，对安全授权码模型的再一次检验。本文研究重点是建立OAuth2.0的安全授权码模型。

参考文献(References)

[1] 严海星.OAuth协议的形式化建模与验证.华东师范大学,2015.

[2] Google,(2012)Using OAuth2.0 to Access Google APIs.

[3] Available:http://developers.google.com/accounts/docs/OAuth2.

[4] Available:https://dev.twitter.com/docs/auth/pin-based authorization.

[5] S.Pai,et al.Formal Verification of OAuth 2.0 using Alloy Framework[J].Communication Systems and Network Technologies(CSNT),2011 International Conference on.IEEE,2011:655-659.

[6] D.Jackson.Alloy:a Lightweight Object Modeling Notation[J].ACM Transactions on Software Engineering and Methodology (Tosem),2002,11(2):256-290.

[7] Available:http://alloy.mit.edu/alloy/index.html(2014).

[8] C.Bansal,K.Bhargavan,S.Maffeis Discovering Concrere Attacks on Website Authorization by Formal Analysis[J].Computer Security Foundations Sysposium(CSF),2012 IEEE 25th. IEEE,2012:247-262.

[9] R.Milner.Communicating and Mobile System:the Pi Calculus. Cambridge University Press,1999.

[10] B.Blanchet,B.Smyth.Proverif 1.85:Automatic Cryptographic Protocol Verifier[J].User Manual and Tutorial,2011.

[11] G Bai,et al.AUTHSCAN:Automatic Extraction of Web Authentication Protocols from Implementations[J]. NDSS,2013.

[12] S.Chari,C.S.Jutla,A.Roy.Universally Composable Security Analysis of OAuth v2.0.[J].IACR Cryptology ePrint Archive,2011:526.

作者简介：

王婷婷(1986-)，女，硕士，中级工程师.研究领域：信息安全.

赵松泽(1998-)，男，高中生.