Model-driven Security Testing of SAML Single Sign-On System

Hou Weitao[1,2,a], Li Menghao[1,b], Liu Jian[1,2,c,*] and Huo Wei[1,2,d]

[1]Key Laboratory of Network Assessment Technology & Beijing Key Laboratory of Network Security and Protection Technology, Institute of Information Engineering, Chinese Academy of Science, Beijing 100093, China

[2]School Of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

{[a]houweitao, [b]limenghao, [c]liujian6, [d]huowei}@iie.ac.cn

*Corresponding author

Keywords: Security Assertion Markup Language, Single Sign-On, security testing, Fuzz.

Abstract. According to investigation, existing works of security testing for Single Sign-on systems (SSO) on the Security Assertion Markup Language (SAML) are based on partially automatic code review methods, which lead to a low level in effectiveness and reusability. In response to these limitations, a new automatic model-driven security testing framework is proposed. This method utilizes a broker-agent to obtain input traces automatically. Different from most previous methods which are applied to OAuth or openID protocols, in our method a customized fuzzy testing engine is designed to SAML protocol. This engine includes special mutation strategies and an abnormal monitor mechanism. Based on this approach, we have developed a prototypical tool called SSOFuzzer and evaluated it with several SSO reference systems, such as onelogin and myOneLogin. The experimental results show that compared to semi-automatic tools like SAMLRaider, SSOFuzzer can accelerate the generation of test cases by 12.4 times. SSOFuzzer also found four unknown security flaws and one known security flaw from our benchmark systems.

模型驱动的SAML单点登录系统安全测试技术

侯伟涛[1,2,a]，李孟豪[1,b]，刘剑[1,2,c,*]，霍玮[1,2,d]

[1]中国科学院信息工程研究所 网络测评技术重点实验室 网络安全防护技术北京市重点实验室 北京 100093 中国

[2]中国科学院大学 网络空间安全学院 北京 100049 中国

{[a]houweitao, [b]limenghao, [c]liujian6, [d]huowei}@iie.ac.cn

[*]通讯作者

关键词: 安全断言标记语言；单点登录系统；安全测试；模糊测试

中文摘要. 针对目前基于安全断言标记语言实现的单点登录系统自动化安全测试程度低、可重用性较低的问题，提出一种模型驱动的安全测试框架。该框架主要从以下三个方面展开研究。首先，设计基于中间人代理的自动化测试框架，该框架用于自动获取输入，从而节省安全人员的测试时间；其次，设计了更加有效的定制化的模糊测试方案，包括定制化的变异策略和异常的监控方式；最后，在此基础上，实现一个完整的基于安全断言标记语言实现的单点登录系统安全测试工具——SSOFuzzer，并对onelogin、myOneLogin等主流参考实现的系统进行实际测试。在这些实际测试中，在相同的时间内，SSOFuzzer生成的测试用例的数量是半

自动化检测工具SAMLRaider的12.4倍，除此之外，SSOFuzzer共发现了4个未知的安全缺陷和1个已知的安全缺陷。

1. 引言

单点登录系统(Single Sign-On,SSO)是管理网络中分散的用户身份信息、实现一次认证就可以跨域访问的机制。安全断言标记语言（Security Assertion Markup Language，SAML）是一种基于可扩展标记语言（Extensible Markup Language，XML）的面向Web服务的架构，可以用在不同的安全域来交换认证和授权数据。SAML作为工业标准受到广大企业的大力支持，像VMware、IBM、谷歌等国际大厂商都采用其标准。单点登录系统包含三类实体，三类实体分别是用户（User），服务提供方（Service Provider，SP）以及身份提供者（Identity Provider，IDP）。一个典型的单点登录系统的交互过程分为7个步骤，整体的交互流程如图1所示。分别是：（1）用户访问SP；（2）SP生成认证请求，然后把认证请求发送给用户；（3）用户将其重定向给IDP；（4）IDP收到请求后，向用户请求用户的凭据（账户和密码）；（5）IDP收到正确的凭据后，生成断言（权限、属性等信息）并对断言进行签名，最终形成完整的响应，并把响应发送给用户；（6）用户再将响应重定向给SP；（7）SP负责确认断言是否由正确IDP服务器发送，确认无误后，SP验证签名，并为用户所需要的权限进行授权，从而允许用户访问所要访问的资源。

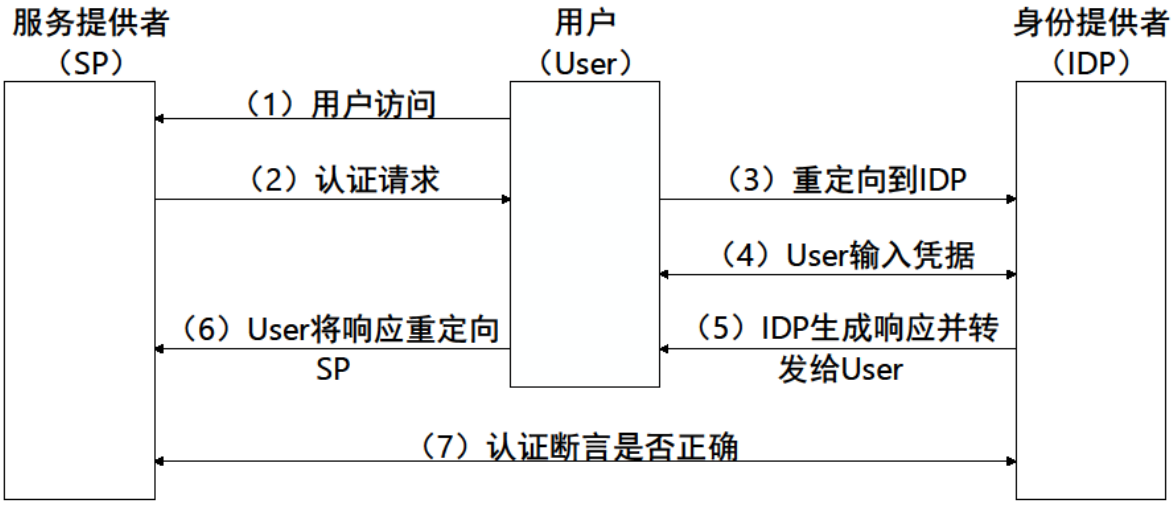

图1 典型的SAML实现的单点登录系统示意图

现有的针对基于SAML的单点登录系统的安全分析技术工作主要有两类：面向模型层的方法和面向实现层的方法。面向模型层的方法采用基于模型检测的技术，将系统规范转换为状态机，然后通过模型检测工具来对其进行检测，例如，文献[1][2]通过分析SAML的解决方案，构建自动化模型，发现了会话劫持、重放攻击和中间人攻击等缺陷；文献[3][5]通过建立SAML的形式化模型，检测出谷歌的单点登录应用中存在安全缺陷，此缺陷可以通过恶意的应用访问未授权的资源。而实现层的工作主要包括有：文献[6]通过对SAML规范中断言的XML签名验证过程的分析，在14个基于SAML框架实现的单点登录系统中发现了11个有签名封装攻击的漏洞，并开发了一套渗透工具——SAMLRaider；文献[7]通过引入恶意的IDP的方式来攻击和分析基于openID实现的单点登录系统，并且成功的绕过它的身份验证；文献[8]采用模糊测试的思想对基于OAuth2.0实现的单点登录系统进行安全测试，同时发现了2类新的漏洞。综上所述，面向模型层的方法由于采用模型检测技术往往只考虑抽象的协议规范，未考虑具体的代码实现，因此无法发现实现上的漏洞；面向实现层的方法，由于缺乏全局的抽象模型，往往不能覆盖目标协议所有的交互路径，从而不能系统性的遍历运行空间。此外，现

有的面向实现层的自动化方法主要集中于OAuth、openID等类型的实现系统，针对SAML单点登录系统主要采用半自动化的代码审计方法，还没有完全自动化测试工具。

由于现有的SAML分析方法主要是半自动化的代码审计，存在效率低、可重用性不高的问题。本文提出了一种模型驱动的SAML单点登录系统的模糊测试方法，从而帮助安全人员提高工作效率。本文首先分析SAML协议的交互流程和信息表示机制，主要通过两个方面来解决现有工具中存在的问题：一是建立基于中间人代理思想的单点登录系统模型，基于此模型实现自动化的捕获消息，并将捕获的消息进行变异，然后测试变异的消息能否触发异常；二是定制化的模糊测试方案，设计专有的变异因子和异常监控的方式，系统化遍历协议的交互空间，自动化判断访问请求是否异常。本文的主要贡献有：

1) 本文设计基于中间人代理的自动化测试框架，该框架用于自动获取输入，并将输入作为种子进行变异，从而节省安全人员的测试时间；

2) 本文设计针对SAML的访问请求的定制化模糊测试方案，包括定制化的变异策略和异常的监控方式；

3) 在此基础上本文实现一个针对SAML单点登录系统的安全测试工具——SSOFuzzer。使用SSOFuzzer并对onelogin、myOneLogin等主流SAML单点登录系统进行测试，实验表明与现有的半自动化方法相比在同等测试时间下SSOFuzzer将产生的测试用例数提高了12.4倍，此外实验共发现5个安全缺陷，其中4个是未知的安全缺陷，1个已知的安全缺陷。

2. 模型驱动的模糊测试

模糊测试的核心是样本构造和异常监控，样本构造依赖于输入和变异因子，而输入又是产生大量测试用例的基础,变异因子是产生畸形消息的保证。本节将介绍如何自动化生成大量的测试用例以及异常监控的方式。

2.1 基于中间人代理的登录模型

基于SAML的单点登录系统的输入是三类实体之间的网络包，由于这些网络包有其特定的结构，不能随意构造，因此，本文提出一种基于中间人代理的登录模型来获取实体之间消息。

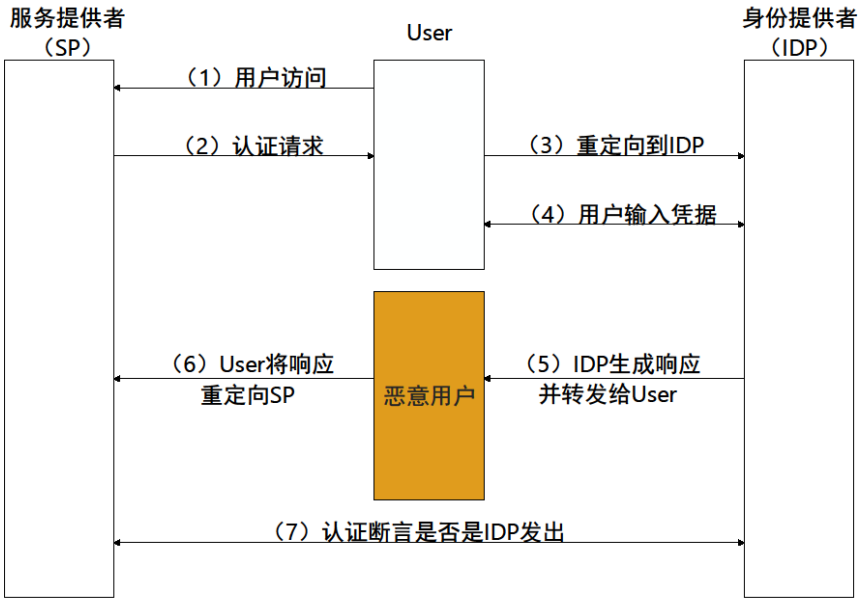

图2 登录模型示意图

首先，针对基于SAML实现的单点登录系统，本文将提出针对性的假设模型，以指导对系统空间的遍历。初步设想的假设模型包括：1）A模型是一个正常的账户，它将遵循着正常的指令，然后发送请求给目标系统；2）B模型是一个模拟攻击者，它不遵循正常的指令执行，它用于发现基于SAML单点登录系统中存在的安全缺陷。

其次，A模型扮演正常用户登录的角色，而B模型扮演中间人的角色，用于对A模型的消息的捕获、解析以及篡改。B模型可以获得所有通过用户的消息。本文将以B模型截获断言为例来说明，如图2所示。用户截获断言的处理过程是用户将IDP生成的断言发送给User，然后User将其重定向给SP。因此，A模型的中User可以用（Sn， request， response）来表示其组成，B模型也用相同的元组表示。其中Sn是表示其具体是操作哪一个操作的标签，request代表了截获断言中请求，而response代表了request发送给IDP后，IDP返回的数据。

2.2 模糊测试

本节分为3小节，第1小节介绍本文定义的4类变异因子，第2小节介绍异常监控的方式，第3小节介绍测试用例的生成与测试。

2.2.1 变异因子

变异因子定义的好坏直接决定能否产生异常，因此，变异因子的选取与设计至关重要。通过分析发现，在三类实体之间交互的消息是基于XML构造的。因此针对实体之间的交互，本文定义4类变异因子，这4类变异因子主要集中在XML消息的解析以及XML消息的逻辑处理上。4类变异因子分别是：

type1-MU_IN：

向XML插入恶意的内容，例如，在XML插入闭合标签</saml2p:Response >、</saml2:Issuer >以及JavaScript或者SQL脚本;

type2-MU_ARP：

随机的替换和重复XML消息中标签内容，例如替换断言中<subject>标签，替换的标签随机选择，用于验证消息解析模块中的安全隐患；

type3-MU_ARM：

将恶意断言（Assertion）插入到消息中随机的位置，用于验证消息逻辑处理上的安全隐患；

type4-MU_HM：

对消息发送的header进行变异，如果header中的字段携带恶意的内容或者字段本身出错，可能在解析消息的时候绕过错误内容，而这部分变异主要基于传统的变异因子扩充而来。

2.2.2 异常监控

一个完整的模糊测试工具主要包括两部分，一个是样本的构造;另一个是异常监控，接下来将详细介绍异常监控。

在介绍异常监控开始之前，本文引入两个定义，分别是预定义行为和预期行为：

定义1. 本文将由畸形消息导致的40x状态(400,404等)的响应定义为预定义行为。

定义2. 本文将由2.2.1节中的变异因子变异产生的畸形消息发送给目标系统后，系统返回的响应与预定义行为的相同，则称之位预期行为，反之，则不是预期行为。

目前主流的模糊测试工具中，进程监控是异常监控的主要方式，例如American Fuzzy Lop，Libfuzzer等模糊测试工具均采用进程监控的方式，但由于本文针对的目标系统涉及到三方实体的交互，在实际中无法监控服务器端进程，因此，本文采用分析目标服务器响应的方式来监控异常。监控的主要思想是：首先，本文将变异后的消息都看成是恶意，因此由它产生的响应均为出错状态（40x状态），但是如果在实际的测试中，返回的响应是非40x状态的，就认为其是非预期行为，可能是一个安全缺陷，并将其记录到日志文件中。

2.2.3 测试用例的生成与测试

测试用例的生成依赖于输入和变异，测试用例可由2.1节构造的登录模型捕获的输入与2.2节介绍的4种变异因子共同作用生成。测试用例的生成与测试过程是：首先，用户给定目标系统的地址——目标系统的统一资源定位符（Uniform Resource Locator，URL），并指定要测试的登录状态；然后，SSOFuzzer从输入中自动获取消息，并对截获到的消息进行变异，从而生成新的测试用例，将生成的测试用例发送给目标系统并收集目标系统的响应；最后，SSOFuzzer分析目标系统返回的响应，如果是非预期行为，则将其记录到日志文件中，反之，则舍弃。

3．SSOFuzzer 实现

基于第2节中提出的模糊测试方法，本文实现了一个针对SAML2.0单点登录系统的自动化测试工具—SSOFuzzer。工具采用python进行开发，主要分为三个模块：捕获消息模块（Capture_Module，CM），消息变异模块（Mutate_Module，MM）以及分析模块（Analyze_module,AM），其整体的结构如图3所示。

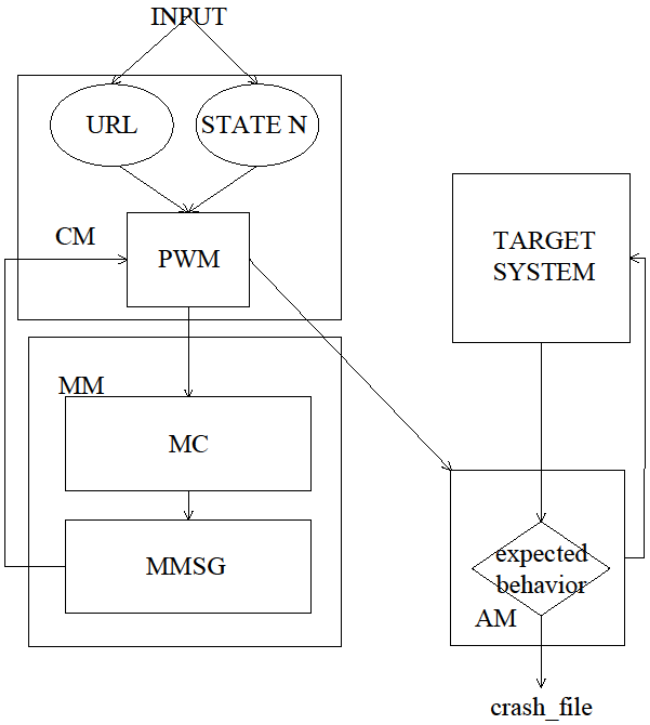

图3 SSOFuzzer 整体框架示意图

捕获消息模块主要是和目标系统进行通信，完成解析和封装消息的功能。捕获消息模块分为三个子模块，分别是提取目标URL提取子模块、消息的状态提取子模块以及消息解析/封装子模块（Parse_Wapper_Module，PWM）。

消息变异模块是实现变异策略的核心模块， PWM将捕获的消息解码后，发送给消息变异模块，变异模块随机选择变异策略后对其进行变异，并将变异后的消息发送给捕获消息模块的PWM子模块。最后，由PWM子模块将变异后的消息进行编码和封装然后发送给分析模块；

分析模块主要负责对消息进行标记，并根据测试系统的响应判断是否出现异常行为。当分析模块收到捕获消息模块发送的过来的消息后，进行标记，然后将其转发给目标系统，目标系统生成相应并将响应发送给分析模块。分析模块将响应和预期行为进行对比判断是否存在一个安全缺陷，若响应不符合预期行为，则是一个异常，并将其记录到日志文件中；反之，则舍弃。

4. 实验分析与结果

在本节中，针对已商用的基于SMAL实现的单点登录系统进行测试评估。测试机器的配置是：2个英特尔的E5-2407 2.20GHz CPUS，8GB内存。对比实验采用SAMLRaider。

4.1 时间效率对比

实验1，工具的时间效率和稳定性。测试时间为2小时（7200秒），每个测试用例一次完整的测试时间包括：测试用例的生成时间、封装时间、传递时间以及响应时间。SSOFuzzer工具不依赖于人，因此其时间主要的花费为测试用例的生成时间和响应时间；而SAMLRaider依赖于人的参与，因此其时间主要花费在测试用例的获取上，具体结果如表1所示。

表1 2小时内产生的测试用例数目

工具	时间（秒）	测试用例	完整测试的平均时间（个/秒）
SSOFuzzer	7200	7589	0.95
SAMLRaider	7200	613	11.75

由表1可知，在同等时间内SSOFuzzer 产生的测试用例是SAMLRaider产生的测试用例的12.4倍，并且每个测试用例从生成到测试完成的时间上前者只有后者的1/12，因此，SSOFuzzer完成一次完整的测试的效率要远远高于SAMLRaider，这样就可以帮助安全分析人员在有限的时间内，尽可能多的产生测试用例，用来测试更多的应用。

4.2 测试用例的丰富度

实验2，由表2可知，SSOFuzzer 包含4种类型变异因子产生的测试用例，而SAMLRaider仅仅包含类型3产生的测试用例。这是由于SAMLRaider是专门针对SAML的签名进行验证，而SSOFuzzer不仅仅对签名部分进行验证，也包括消息解析的其他的部分，例如，header是否能携带恶意的内容、注入其他的脚本（JavaScript、SQL等）等内容。结论，通过实验对比发现，SSOFuzzer 能产生的测试用例种类更多，覆盖面更广。

表2 测试用例的丰富度

工具	type1-MU_IN	type2-MU_ARP	type3-MU_ARM	type4-MU_HM
SSOFuzzer	✓	✓	✓	✓
SAMLRaider	×	×	✓	×

4.3 漏洞发现

实验3，统计SSOFuzzer 测试应用中产生的漏洞。在本实验中，SSOFuzzer共发现4个未知漏洞，1个已知的缺陷，如表3所示；在JAVA-SAML共发现了两个未知的漏洞，但是它没有发现已知的漏洞；在PYTHON-SAML中发现了1个未知的漏洞，发现了1个已知的漏洞；然而，在PHP-SAML中只找到了一个未知的漏洞，但它没有发现已知的漏洞。此外，JAVA-SAML、PYTHON-SAML和PHP-SAML都发现了令牌重放的漏洞。

表3 异常发现

目标	不合法字符串	重放	已知漏洞
JAVA-SAML	✓	✓	×
PYTHON-SAML	×	✓	✓
PHP-SAML	×	✓	×

结论：由上述实验1,2,3可知，由于SSOFuzzer 采用基于中间人代理的设计，可以实现自动化测试，可以大大节省安全人员的测试时间，同时由于SSOFuzzer 比SAMLRadier产生的测试用例种类更多，覆盖的测试面更广。此外，最重要的是SSOFuzzer 的变异源于已知文献中的攻击模型，因此SSOFuzzer 能发现应用中存在的漏洞。

5．结束语

本文针对基于SAML实现的单点登录系统，设计实现一个自动化的测试框架，使用此框架不仅仅可以节省安全人员时间，同时也能发现SSO应用中漏洞。未来还会继续增加新的模块和维护自动化测试工具，例如，适配更多的协议实现的单点登录系统，另外还会丰富变异因子的种类，增加新的变异因子。

致谢

本文为国家自然基金项目（61572481、61602470），北京市科委项目（D161100001216001），中科院先导项目（XDA06010703）的阶段性成果之一。

References

[1] Groß T. Security analysis of the SAML single sign-on browser/artifact profile[C]//Computer Security Applications Conference, 2003. Proceedings. 19th Annual. IEEE, 2003: 298-307.

[2] Groß T, Pfitzmann B. SAML artifact information flow revisited[C]//In IEEE Workshop on Web Services Security (WSSS). 2006: 84-100.

[3] Armando A, Carbone R, Compagna L, et al. From multiple credentials to browser-based single sign-on: Are we more secure?[C]//IFIP International Information Security Conference. Springer, Berlin, Heidelberg, 2011: 68-79.

[4] Sudhodanan A, Armando A, Carbone R, et al. Attack Patterns for Black-Box Security Testing of Multi-Party Web Applications[C]//NDSS. 2016.

[5] Armando A, Carbone R, Compagna L, et al. Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps[C]//Proceedings of the 6th ACM workshop on Formal methods in security engineering. ACM, 2008: 1-10.

[6] Somorovsky J, Mayer A, Schwenk J, et al. On Breaking SAML: Be Whoever You Want to Be[C]//USENIX Security Symposium. 2012: 397-412.

[7] Mainka C, Mladenov V, Schwenk J. Do not trust me: Using malicious IdPs for analyzing and attacking Single Sign-On[C]//Security and Privacy (EuroS&P), 2016 IEEE European Symposium on. IEEE, 2016: 321-336.

[8] Yang R, Li G, Lau W C, et al. Model-based security testing: an empirical study on OAuth 2.0 implementations[C]//Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. ACM, 2016: 651-662.

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

[10] Zhou Y, Evans D. SSOScan: Automated Testing of Web Applications for Single Sign-On Vulnerabilities[C]//USENIX Security Symposium. 2014: 495-510.

[11] Jan S, Nguyen C D, Briand L C. Automated and effective testing of web services for XML injection attacks[C]//Proceedings of the 25th International Symposium on Software Testing and Analysis. ACM, 2016: 12-23.

[12] Wang Shanshan. Study on XML Security Authentication protocol and Single Sign-on System[D]. Xidian University,2004.