基于通用知识的软件设计安全性评估①

戚荣波, 杜 晶, 杨 叶

(中国科学院 软件研究所, 北京 100190)

摘 要: 安全性作为软件系统的重要属性, 越来越受到人们的重视. 在软件开发的早期对安全性进行评估, 对软件的质量控制和成本控制有着重要意义. 当前的软件安全性评估主要依靠专家评审, 结果的客观性及准确性常常受到专家主观意见的影响. 通过使用通用知识作为评估依据, 提出一种可以对 UML 顺序图形式的软件设计文档进行自动化分析的方法, 可以发现软件设计中潜在的安全性漏洞. 该方法可以减少结果中的主观性, 同时, 通过基于该方法的辅助工具的使用, 可以大大提高评估效率.

关键词: 软件安全性; 通用知识; 攻击模式; 软件安全性评估; UML 顺序图

Knowledge-Based Security Evaluation in Software Design

QI Rong-Bo, DU Jing, YANG Ye

(Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: As an important property of the software system, software security has drawn more and more attention. Security evaluation in the early phase of the software development is important to software quality and cost control. Current expert-based review and inspection methods can be error-prone and subjective. In this paper, we have proposed a knowledge-based security evaluation approach which can detect vulnerabilities in the UML sequence diagrams. With common knowledge as the evaluation reference, we can reduce the subjectivity in the result. We also have developed a support tool which can automatically perform most of the work in the method and improve the efficiency.

Key words: software security; common knowledge; attack pattern; software security evaluation; UML sequence diagram

软件安全性是软件系统的一个重要属性[1]. 软件安全性的评估工作, 对软件安全有着重要意义. 软件安全性存在于软件的整个生命周期, 但在软件开发后期解决问题的代价往往要远大于前期. 因此, 在软件开发早期对软件设计进行安全性评估, 对软件安全性有重要意义.

在软件开发早期, 当前主流的评估方法是对软件设计进行专家评审[2], 这类方法往往耗时长, 代价高. 同时, 由于依靠专家判断, 结果的客观性往往会受到影响. 此外, 专家资源作为一种稀缺资源, 也对软件的安全性评估增加了难度和成本.

针对这些问题, 本文提出了一种在软件设计阶段进行的安全性评估的方法. 此方法以 UML 顺序图形式的软件设计文档为评估对象, 对软件攻击模式的通用知识进行分析建模以作为安全性评估的依据, 从而对软件设计的安全性进行客观的评估. 同时, 我们也实现了相应的评估工具, 对此方法中的大部分过程实现了自动化.

1 研究现状

1.1 评估依据

当前软件安全性评估的主要依据及参考是一些公开的安全性标准. ITSEC[7]标准是第一个单独对安全性评估建立的标准, 主要针对的是许多欧洲国家的计算机系统和产品. 在此之后, Common Criteria 标准[8]被建立以取代 TCSEC[9]以及 ITSEC, 它包含全球公认的评

① 基金项目:国家自然科学基金(61073044,71101138,61003028);国家重大科技专项(2012ZX01039-004);北京市自然科学基金(4122087)
收稿时间:2013-04-09;收到修改稿时间:2013-05-07

估标准, 目前被广泛使用. RFC 3871[10]定义了一系列针对大型 IP 网络设施的操作性安全需求, 它使得网络运营商可以和设备供应商沟通他们的安全性需求. MITRE 通过枚举基线安全性数据以提高安全性的可测量性, 通过提供标准化的语言作为工具来进行准确的沟通, 并且通过建立通用知识库来鼓励用户分享信息[19,20]. ISO 27000 系列标准[11]和 DO178-B[12]分别是信息系统和机载软件系统的特定领域的标准. 它们也经常被用于相应领域的系统安全性的评估.

1.2 评估方法

目前对软件安全性进行评估的主流方法是专家评审, 通常使用一些安全性的标准作为评估的依据. 范根检查法[13]是 IBM 在 70 年代开发出的一种小组评审方法, 至今仍被广泛应用. 它通过验证过程的输出是否符合过程指定的出口条件来进行评估, 以实现软件质量控制. 架构评审也是目前行之有效的一种方案[14], 可以发现软件系统中潜在的, 特别是与质量属性相关的一些问题. 这些方法往往流程复杂, 耗时长, 代价高, 评估结果带有主观性.

也有一些方法是从代码层面对软件源代码进行分析, 静态代码分析和渗透测试[15]就是两种典型的针对源代码进行审查和验证的方法, 也有一些研究是将安全性融入开发方法[3,4]. 这些方法往往可以使用一些支持工具来协助完成. 不过这由于这些方法一般只能在软件开发的后期阶段进行, 发现缺陷或问题后, 进行修复的代价会比较大.

还有一些方法主要针对软件设计或软件架构. 一些学者通过建立系统的状态转移模型, 并对模型进行分析和计算, 得出量化的安全性结果[16]. 另一些学者使用了 SysML 活动图描述系统和攻击, 并通过对攻击模型和系统模型的交互进行分析, 以获得量化的安全性结果[17]. 类似的基于系统建模的评估方法的研究有很多[5,6]. 不过这些方法往往模型复杂, 难以理解, 不易于实际操作.

1.3 软件安全性相关的知识

软件安全性相关的知识, 可以分为三个类别: 规范型、历史型以及诊断型[18].

规范型的知识是指从软件架构的不同层面中抽象出来的原则或指导性建议, 它主要用于在开发安全性软件时提供“该做什么”以及“不该做什么”的建议.

历史型的知识主要是历史风险. 这类知识一般详细记录了软件开发工业中发现过的问题以及问题所造成的影响. 当遇到新的类似问题时, 经常会用这类知识来提供参考帮助.

诊断型的知识包括攻击模式和软件漏洞等. 这类知识一般用于识别和处理安全性攻击相关的问题. 其中, 攻击模式是在特定环境下普遍发生的攻击的泛化表示, 软件漏洞是指软件中易遭攻击的弱点.

MITRE 维护的通用攻击模式列表(CAPEC), 描述了利用软件漏洞进行攻击的方法以及应对的措施. 同时, 通用攻击模型列表还可以和同由 MITRE 维护的通用软件弱点列表(CWE)相关联. 其在软件安全性评估和验证上被广泛采用.

2 基于通用知识的安全性评估方法

通过融合整个软件过程中产出的各种证据, 针对整个软件开发生命周期, 中国科学院软件所提出了一些基于证据的软件过程评估方法[21,22]. 这些方法基于可信度的度量模型, 客观地度量软件过程的可信度, 建立了面向整个软件开发生命周期的基于证据的软件可信度量框架.

基于此框架, 特别针对软件安全性, 我们以通用知识作为评估依据, 在软件的设计阶段, 对软件设计文档进行评估. 通过对通用知识的使用, 可以使评估过程减少对专家判断的需求, 在降低成本的同时也可以保证结果的客观性.

本文使用的通用知识包含通用攻击模式以及通用软件弱点. 其中, 攻击模式用于分析软件设计是否存在安全性威胁, 软件弱点用于生成软件漏洞列表.

软件设计文档的形式有多种. 其中, UML 文档相对于一般文本描述的文档, 表达更为直观, 并且逻辑性更加清晰. 顺序图作为 UML 图的一种, 常用于表示系统典型的执行场景. 而针对软件的安全性攻击, 一般正是针对系统执行过程中的弱点而进行的. 因此, 顺序图更易于使用攻击模式来分析软件设计中的安全性漏洞. 本方法采用了 UML 顺序图形式的软件设计文档作为评估对象. 其它形式的设计文档, 也可以参考本方法进行处理.

对于顺序图和攻击模式, 本方法分别定义了相应的设计模型和威胁模型, 通过对它们的建模分析, 可以发现软件设计中存在的安全性漏洞. 整个方法的整体框架如图 1 所示.

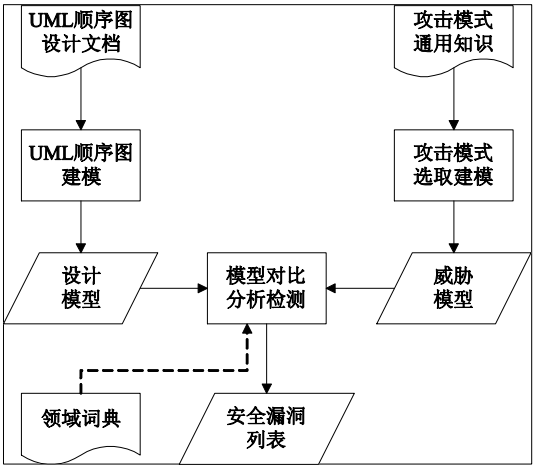


图 1 评估方法的整体框架

从图 1 中可以看出, 本方法的主要工作包括三部分, 分别是对 UML 顺序图的建模, 对攻击模式的选取和建模, 以及模型的对比分析. 下面分别从这三个部分来对方法进行介绍.

2.1 UML 顺序图的建模

我们设计了一个四元组的模型来表示软件设计.

定义 1. 设计模型: (主体, 行为, 资源, 反馈).

每条设计模型表示软件设计中用户或模块对某项资源的访问以及系统因此所产生的反应. 其中, 主体表示访问的发起者, 一般是用户或一些系统的外部模块; 行为表示访问的具体内容; 资源表示访问所涉及的具体资源; 反馈表示系统因为访问而产生的反应.

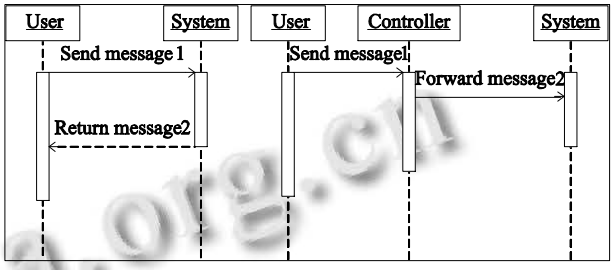
我们需要将 UML 顺序图建模成定义 1 中的设计模型, 需要提出相应的转换规则.

顺序图描述的是对象之间的动态协作, 通过消息的发送来表示交互. 消息一般具有发送方, 接收方以及消息内容等属性. 而消息之间也具备时序关系和逻辑关系. 对顺序图的建模, 实质上是对顺序图中消息的建模.

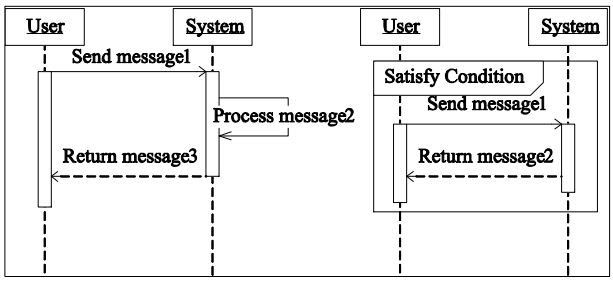
我们将顺序图中常见的基本交互场景分为图 2 所示的几种模式, 并对这几种模式提出了转换的规则.

值得说明的是, 因为表达的灵活性不利于自动化的处理, 在实际建模前, 需要对顺序图进行一定的预处理. 首先删除图中注释类的内容, 将对象的名称规范化, 将消息和条件的内容统一写成动宾结构. 此外, 由于版本的不同以及建模工具实现方式的不同, 顺序图可能还包括一些不常用的其它元素或表达形式, 如消息间的 OR 约束等, 这些都需要转换成图 2 中的形式.

图 2 的 4 种模式中: (a)表示发送消息后, 对象处理完成后立即发送返回消息; (b)表示发送消息后, 对象的处理过程需要向另一个对象发送新的消息; (c)包含了自调用的消息; (d)包含了条件约束下的消息序列.



(a) 返回消息 (b) 传递消息



(c) 自调用消息 (d) 条件约束

图 2 常见的顺序图模式

对于这 4 种模式, 首先分别提取其中对象的名称、消息内容和约束条件. 然后, 通过对消息内容和约束条件进行词性标注和成分分析, 提取其中的动词短语和宾语. 最后, 根据表 1 中的规则, 将其转换成相应的设计模型中的各个元素.

表 1 顺序图常见模式建立设计模型的规则

对应模式	设计模型			
	主体	行为	资源	反馈
(a)	User	Send	Message1	Return message2
(b)	User	Send	Message1	Forward message2
(c)	User	Send	Message1	Process message2
	System	Process	Message2	Return message3
(d)	User	Satisfy	Condition	Send message1
	User	Send	Message1	Return message2

与设计模型类似, 我们设计了一个三元组的模型来表示攻击模式.

定义 2. 威胁模型: (攻击者, 行为, 资源).

每条威胁模型表示攻击模式中攻击者对系统资源的

攻击行为. 其中, 行为表示具体的攻击内容, 资源表示攻击所使用或涉及的资源. 因为攻击模式中往往包含多条步骤, 所以一条攻击模式可能对应多条威胁模型.

本方法使用来自 MITRE 的通用攻击模式文档 CAPEC. 这份攻击模式的列表文档对已知常见的攻击模式进行了组织整理. 同时, MITRE 还整理了与之相关的通用软件弱点文档 CWE, 并在两个文档之间建立了关联. 每种软件弱点, 都可以检索出其可能导致的攻击模式, 反之亦然. 因为两份文档都使用了 XML 进行编排, 因此可以很方便的读取软件弱点和攻击模式的相关元素.

在实际具体项目中, 并非所有攻击模式都与项目相关或者值得关注. 因此, 需要对攻击模式进行筛选, 以缩小范围精确目标, 从而提高结果的相关性和准确性.

筛选主要是通过关键字的检索来实现的. 首先, 根据项目的相关领域和实际需求来选取相应的关键字. 然后, 通过关键字在软件弱点列表中的检索来获得相应的软件弱点. 这些软件弱点是项目可能存在并且值得关注的软件弱点. 再根据软件弱点和攻击模式之间的关联关系, 得到相应的攻击模式.

为了提高自然语言处理结果的有效性, 需要对攻击模式的描述进行预处理, 将一些复杂的句式转换成简单句的形式.

通过自然语言处理技术, 可以提取简单句中的相应成分. 抛弃一些修饰性的句子成分, 如定语、状语等. 将主语、谓语与宾语分别对应到威胁模型中的攻击者、行为与资源上, 就可以建立相应的威胁模型. 表 2 是 3 种攻击模式建模的示例. 其中, 前两条攻击模式只包含一个步骤, 因此只对应一条威胁模型; 第 3 条攻击模式包含多个步骤, 因此对应多条威胁模型.

表 2　对攻击模式建模的示例

攻击模式	威胁模型		
	攻击者	行为	资源
An attacker may try certain common usernames and passwords.	attacker	try	username and password
An attacker modifies the HTTP Verb.	attacker	modify	HTTP verb
An attacker spiders websites.	attacker	spider	website
An attacker brute force guesses usernames.	attacker	guess	username
An attacker brute force guesses function names.	attacker	guess	function name

2.3 模型的对比分析

当序列图设计和攻击模式列表全部建模完成时, 可以得到设计模型的集合与威胁模型的集合. 为了确认每一条威胁模型是否可能对软件设计造成威胁, 需要检查这条威胁模型是否被某条设计模型作了正确地防护.

首先, 对威胁模型的资源项进行检索, 找出与这条威胁模型的资源项匹配的所有设计模型. 如果无法找到相应的设计模型, 那么这条威胁模型的资源没有被系统设计所考虑到, 有可能被攻击. 因此, 需要将这条威胁模型加入列表.

然后, 在上一步中找出的所有设计模型中, 对威胁模型的行为项进行检索, 找出所有与这条威胁模型的行为项匹配的设计模型. 如果无法找到相应的设计模型, 那么这条威胁模型的行为没有被系统设计所考虑到, 也需要将其加入列表.

最后, 在上一步中找出的所有设计模型中, 依次检查其反馈项是否是防护性质的操作, 如拒绝服务、关闭连接等. 如果存在这样的设计模型, 便认为这个威胁模型被成功地防护; 否则, 这个威胁模型也需要加入到列表中.

对每条威胁模型都执行上面的操作, 最终, 可以得到一个威胁模型列表. 这个列表代表了当前设计可能发生的安全性攻击. 在实际建模过程中, 我们会在威胁模型中留下所对应的攻击模式的 ID. 通过将这些 ID, 我们可以将这个威胁模型列表转换成攻击模式的列表, 并进一步转化为软件弱点的列表. 具体的检查过程如图 3 所示.

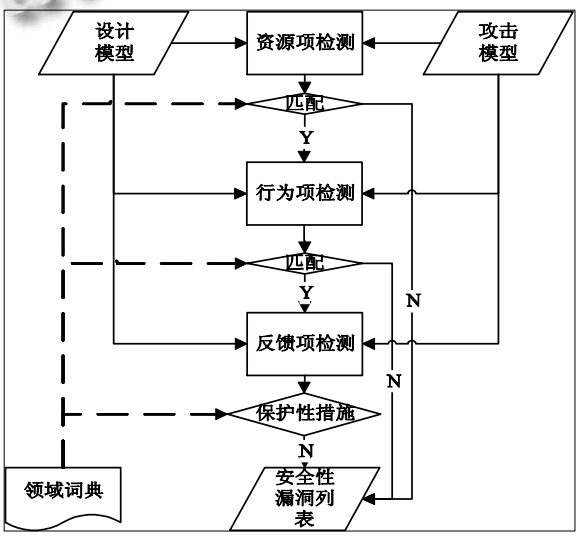


图 3　模型对比分析的过程

在检索与匹配等过程中, 因为实际情况中常常使用不同的词语来表示相同的含义, 同时也存在一些语句是以否定句的形式存在. 因此需要建立一个定义了与项目以及通用知识相关的同义词和反义词的领域词典, 以适应词语的多样性.

3 工具实现及应用

3.1 工具实现

为了提高评估方法的效率, 需要开发相应的辅助工具来实现整个评估过程的自动化. 对于工具的开发, 我们提出了以下几点功能性需求.

① 可以编辑相应的领域词典, 用于评估过程中的筛选和匹配;

② 可以导入 UML 顺序图形式的软件设计文档, 并对文档进行自动化建模;

③ 可以导入软件弱点和攻击模式的通用知识文档, 可以对相应的自然语言描述进行人工的预处理操作, 并自动对处理后的内容进行解析并建模;

④ 可以输入相应的关键词, 从而对攻击模式进行筛选. 筛选后的列表可以用人工选取的方式进一步筛选;

⑤ 自动化地对威胁模型和设计模型进行对比分析, 并最终生成相应的软件漏洞列表;

⑥ 提供图形化向导式的界面, 方便使用.

我们选择了 Java 作为开发语言, 主要因为 Java 的平台无关性, 以及用 Java 编写的自然语言处理的公用库相对丰富. 在经过对比使用后, 我们最终选取了斯坦福大学的 Stanford Parser 工具作为自然语言处理的引擎.

因为不同的 UML 建模软件, 往往使用不同的文件格式对顺序进行持久化. 为了提高工具的通用性, 我们约定文件均采用 XMI 标准的格式. 对于使用 XMI 标准以外的格式来描述的顺序图, 有丰富的第三方工具, 可以将其转换成 XMI 标准的文件.

在开发过程中, 我们将功能分为三个模块, 分别是 Entity, Tool 以及 View. 其中 Entity 模块主要定义了通用知识文档的结构以及模型的定义. Tool 模块包含了整个评估过程中的大部分工作, 主要包括文档读取和建模、自然语言处理、模型对比分析以及领域词典的管理. View 模块包含了整个图形界面的设计, 包括导入设计文档和通用知识文档、设计关键词、编辑领域词典、筛选攻击模式、输出评估结果等界面.

3.2 应用分析

图 4 是一个真实的网站项目的登陆界面的 UML 顺序图. 为了便于处理, 该图作了一定程度的简化和预处理.

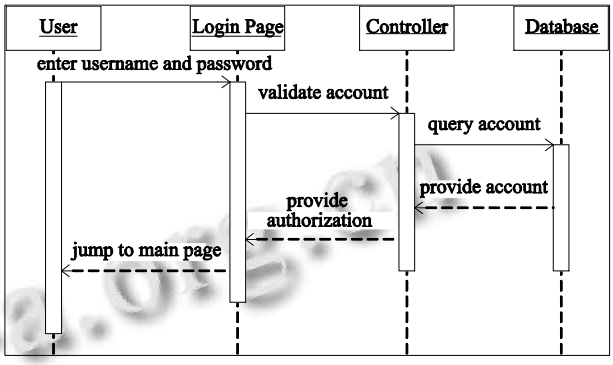


图 4 登陆界面功能的 UML 顺序图

在工具中导入这个顺序图, 对其建模处理, 可以得到图 5 中设计模型的列表. 该列表中每条记录表示一条设计模型. User, Action, Asset, Reaction 四个字段分别表示主体、行为、资源和反馈.

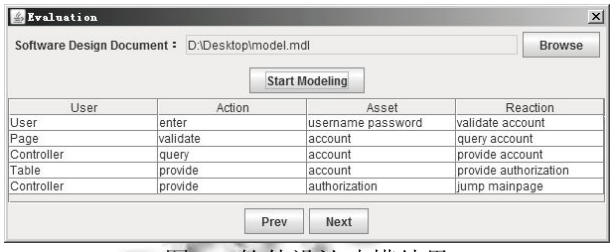
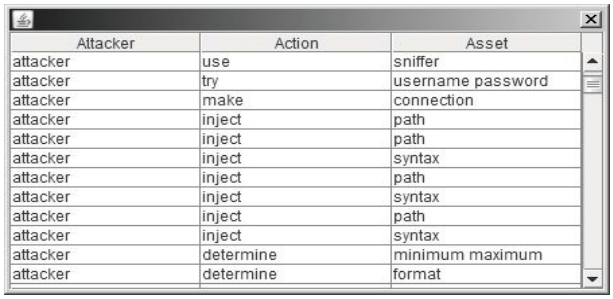


图 5 软件设计建模结果

在下面的步骤中, 需要导入相应的通用攻击模式文档和软件弱点文档. 在通过关键词筛选过相应的攻击模式后, 对这些攻击模式进行建模, 可以得到图 6 中威胁模型的列表. 其中, 每条记录表示一条威胁模型, Attacker, Action 和 Asset 三个字段分别代表攻击者、行为以及资源.



Attacker	Action	Asset
attacker	use	sniffer
attacker	try	username password
attacker	make	connection
attacker	inject	path
attacker	inject	path
attacker	inject	syntax
attacker	inject	path
attacker	inject	syntax
attacker	inject	path
attacker	inject	syntax
attacker	determine	minimum maximum
attacker	determine	format

图 6 攻击模式建模结果

最后，对设计模型和依据模型进行对比分析，可以得到该设计存在的软件漏洞列表，如图 7 所示. 其中，每条记录表示一条软件漏洞，前两个字段表示最后得到的攻击模式，后两个字段表示这个攻击模式所对应的软件弱点.

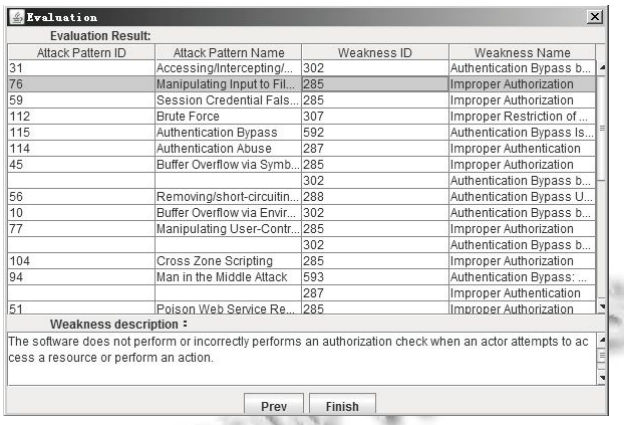


图 7 模型对比分析结果

在工具的辅助下，整个评估过程由一人完成，共花费了 40 分钟. 其中建立领域词典花费了 20 分钟，对顺序图的预处理花费了 2 分钟；对攻击模式的筛选花费了 5 分钟；对攻击模式文本的预处理花费了 10 分钟. 值得注意的是，对攻击模式文本的预处理是可以复用的，只有第一次评估需要进行. 可以看出，本方法的效率是很高的.

在这个项目上，我们最终得到了 17 条安全性漏洞，通过与项目开发人员的沟通和讨论，其中有 9 条安全性漏洞是原先的软件设计中没有考虑到的，有 5 条安全性漏洞是软件设计中已经考虑到的，有 3 条安全性漏洞是与本项目无关的. 可以看出，本方法是有效的，但在准确性上需要提高. 这一方面受限于自然语言处理技术本身的准确性，同时也和领域词典可能不够完善有关.

4 结语

安全性评估，特别是在软件开发的早期阶段进行的安全性评估，对于软件系统是非常重要的. 目前主流的专家评审方法往往耗时长，代价高，结果的客观性往往受专家判断的影响. 而一些模型分析的方法往往有不易理解，建模困难等问题.

本文提出的评估方法，以攻击模式的通用知识作为评估依据，以 UML 顺序图作为评估对象，使用了自然语言处理技术来帮助建立模型，通过模型的对比分析来检测软件设计中的漏洞. 模型简单，易于理解. 同时，通过使用辅助工具，可以自动化地实现方法中的大部分工作，从而提高评估效率.

根据在实际项目中的应用情况，本方法可以快速有效地发现软件设计中的安全性漏洞，但也会发现一些错误的漏洞，在结果的准确性方面还需要进一步的提高.

在下一步的工作中，我们会进一步研究自然语言处理技术，提高模型建立的准确性；同时，也会考虑对其它形式的软件设计文档进行支持.

参考文献

1 陈火旺,王戟,董威.高可信软件工程技术.电子学报,2003,31(z1):1933–1938.

2 王敏,刘波,刘东培.安全可靠软件系统的设计与实现.光电技术应用,2004,18(1):50–53.

3 任地成,王丽君,潘瑞.融入安全的软件开发方法 SDL 的研究.计算机应用与软件,2010,27(7):280–282,293.

4 王志皓,赵保华,赵婷.基于 SDL 的软件安全测试方法研究.电力信息化,2012,10(11):8–11.

5 龚军,张菊玲,吴向前,刘胜全.信息系统安全风险评估在校园网中的应用.计算机应用与软件,2011,28(3):285–288.

6 吴迪,连一峰,陈恺.一种基于攻击图的安全威胁识别和分析方法.计算机学报,2012,35(9):1938–1950.

7 Office for Official Publications of the European Communities. Information Technology Security Evaluation Criteria, version 1.2.1991.

8 The National Information Assurance Partnership. The Common Criteria for Information Technology Security Evaluation. http://www.niap-ccevs.org/Documents_and_Guidance/cc_docs.cfm.1996.

9 US DoD 5200.28-STD,Trusted Computer System Evaluation Criteria.1985.

10 Jones G.RFC 3871. http://www.ietf.org/rfc/rfc3871.txt.2004.

11 International Standards Organization. ISO 27000, First Edition. 2005.

12 RTCA SC-167, EUROCAE WG-12. Software Considerations in Airborne Systems and Equipment Certification. 1992.

13 Fagan ME. Advances in software inspections. IEEE Trans-

(下转第 80 页)

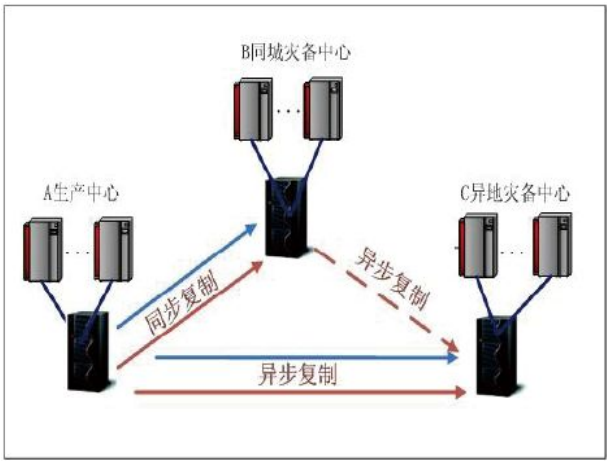


图 12 多跳与分发相结合的模式

3 结束语

基于存储系统的容灾方案, 无论采用同步模式还是异步模式, 均需要一个前提条件: 两边必须部署同品牌、同级别的存储设备. 这一特点与成本预算有着直接的关联关系. 在现有可行方案中, 基于 SAN 的容灾技术可以从底层化解同构问题. 为了将存储设备从数据同步的压力中解放出来, 可在 SAN 交换机上或 SAN 交换机这一层采用专用的硬件设备, 完成数据同步策略管理及异构设备的支持. 其不足之处是增加了底层数据传输层次和维护难度.

参考文献

1 郭继君.大集中模式下存储容灾系统设计方案.华南金融电脑,2002,(9):42–46.
2 施蓓莉.基于 HP 主机平台加 EMC/EVA 存储的银行业务系统灾备实现.上海交通大学,2008:4–16.
3 杨柳明.论商业银行容灾系统建设.广西金融电脑,2008,(2):56–59.
4 王德军,王丽娜.容灾系统研究.计算机工程,2005,31(6):43–46.
5 刘迎风.容灾技术及其应用.计算机应用研究,2002,19(6):11–15.
6 徐鹏,薛建新.数据中心容灾系统研究.计算机工程与设计,2007,28(33):22–27.
7 许福忠.银行交易系统灾备技术研究与应用.国防科技大学,2006:7–17.
8 叶嘉铭,胡晓勤.多数据库容灾系统的设计与实现.计算机工程与设计,2012,33(12):241–245.
9 蔡皖东,何得勇.一种网络容灾系统的设计与实现.计算机工程,2004,30(7):116–118.
10 于新华,刘川意.远程容灾系统设计框架.计算机工程与应用,2006,42(24):31–34.
11 武鲁,李钟华.基于集群服务器的容灾系统的副本管理研究.计算机应用研究,2006,23(6):76–78.
12 陈汶滨,吕曼曼.容灾备份系统研究.计算机安全,2009,(7):70–71.
13 施云龙,韩广琳.浅谈银行系统柜面服务器集群的容灾备份方案.福建电脑,2006,(5):69–71.
14 杨景发.商业银行四地容灾系统的设计与实现.上海交通大学,2011:6–43.

(上接第 6 页)

actions on Software Engineering, SE-12(7): 744–751.
14 Maranzano JF, Rozsypal SA, Zimmerman GH. Architecture reviews: practice and experience. IEEE Software, 22(2): 34–43.
15 Arkin B, Stender S, McGraw G. Software penetration testing. IEEE Security & Privacy, 2005, 3: 84–87.
16 Madan BB, Gogeva-Popstojanova K, Vaidyanathan K. Modeling and quantification of security attributes of software systems. Proc. Int'l Conf. Dependable Systems and Networks. 2002.
17 Ouchani S, Jarraya Y, Ait Mohamed O. Model-based systems security quantification. Privacy, Security and Trust(PST), 2011 Ninth Annual International Conference on. IEEE. 2011. 142–149.
18 Barnum S, McGraw G. Knowledge for software security. Security and Privacy, IEEE, 2005, 3(2): 74–78.
19 MITRE.Common Attack Pattern Enumeration and Classification,Version 1.7.1.http://capec.mitre.org/.2012.
20 MITRE.Common Weakness Enumeration.Version2.4.http://cwe. mitre.org/.2013.
21 杜晶,杨叶,王青.基于证据的可信软件过程评估方法.计算机科学与探索,2011,5(6):501–512.
22 Yang Y, Wang Q, Li M. Process trustworthiness as a capability indicator for measuring and improving software trustworthiness. Trustworthy Software Development Processes. Springer. Berlin, Heidelberg. 2009. 389–401.