系统与软件安全浅析

蔡霞　李晓燕　陈基雄　林云　(华中师范大学计算机科学系　武汉 430079)

摘要:本文用系统工程的概念论述了系统与软件安全,并集中论述了如何将系统与软件安全分析集成到系统开发的各个生命周期中。

关键词:系统安全　软件安全　危险　失误

一、概述

数字计算机具有日益增长的多面性和能力,并能以更少的耗费获得改善的性能和更高的效率。人们一直认为引进计算机也将提高安全。但当用计算机代替可以获得更高可靠性等级的电子机械设备时,安全性反而可能会降低。

即使计算机可以提高安全性,但最终是否在系统安全上提高了仍然是不一定的,当技术提高时,同时增加的有关安全的潜在的问题不能完全被认识到。尽管如此,计算机仍被运用于控制一些危险性系统。对于安全性至关重要的电子机械设备的制造技术和方法的理解有助于软件新技术的开发,以及这些新技术与硬件和工具的衔接。开发应用于系统范围内的全面而完整的技术和工具,已成为研究的热点。

于是系统与软件安全已经成为令人关注的重要问题。

1. 安全(Safety)

安全是"防止那些可能引起死亡、伤害、职业病或设备及财产的损伤或损失",是在任何环境防止危险的程度的一种衡量。显然,安全是一个相对的概念。几乎任一个产生利益的系统都包含不可信赖的因素。例如:安全火柴。往往尝试去消除危险时,常常会使危险转移,而没有真正地清除危险,于是这个问题便复杂化了。

2. 系统与软件安全

安全独立出来后,必须用危险(Hazard)或者是可导致一个失误(Miscap)的系统状态来定义和描述。危险(Hazard)是会导致或可能引起一个失误的已存在的或潜在的条件。失误(Miscap)可认为是由任意可能组合的一系列连续的并发的相关事件。在满足危险发生的条件的情况下,这一系列事件的发生会导致系统失去控制且产生损失(如多米诺骨牌事件理论,即一个倒,全部倒)。

对于系统而言,危险有两个方面的含义:(1)系统会进入一个危险状态的可能性,即有可能引起一个 miscap 的条件;(2)导致一个 miscap 的危险,这时还要分析导致 miscap 的危险的严重性。前者称为危险可能性,后者称作危险的关键性。系统的危险可能是由于硬件错误,软件设计错误,在系统各个部分之间的接口(通信和实时)问题,人为操作或维护错误,或环境压力而引起的。

系统安全是一个在操作效果、时间、和耗费的约束下,在系统生命周期内,运用科学的管理和工程的原则来确保足够安全的系统工程子原则。

系统的状态由系统的各个部分的状态组成。通常计算机是作为系统的控制者,因而对于目前的系统状态有直接的影响。故讨论软件安全是非常有意义的。

软件安全是在软件生命周期中,运用系统安全工程的技术原则来保证软件采取正确的措施增强系统安全,确保使系统安全性降低的错误已被消除或控制在一个可接受的危险级别内。系统安全涉及系统所有的危险状态。而软件安全确保在系统上下文内执行软件时不会产生不可接受的危险。与硬件安全一样,在开发阶段早期可先识别危险,然后通过建立需求和设计去消除或控制这些危险来获取软件安全。

系统安全的目标:在实际的生产或操作前设计一个可接受的安全标准给系统。

系统安全工程的任务:第一步,系统安全工程运用科学和工程原则优化安全,并通过分析、设计和管理进程来识别和控制危险。危险分析包括识别和评估危险所涉及到的系统危险和关键性等级。下一步是清除设计中达到不可接受的等级并且已识别的危险,如果不能消除,则应减少相应的危险以达到一个可接受的水平。

二、系统安全分析和建模

系统安全分析开始于接受工程时并贯穿于整个系统生命周期的始终。在不同的阶段执行的分析是基本危险分析(PHA)、系统危险分析(SHA),以及操作和支持危险分析(OSHA)。系统与软件安全分析在系统与软件生命周期不同的阶段所执行的分析如图 1 所示。

系统安全建模和分析的目的是表明系统是安全的

(即使存在恶意操作和错误)。为了证实在错误出现时一个复杂系统的安全性,必须尽可能列出可以引起危险效果的所有错误序列,并证明导致足够危险的错误序列出现的可能性足够小。后者的分析接近于企图以所有可能的序列组合去分析输出的可能性。系统安全分析过程通常开始于定义什么是危险,并且通过危险定义去找出产生错误事件的所有组合,然后计算出事件发生的可能性并估计出结果。

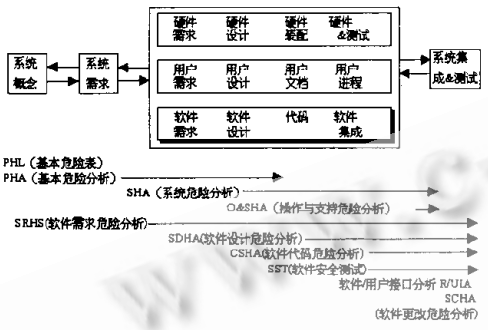

图 1　系统与软件安全分析

因而在任何一个安全系统的第一步是识别危险并根据关键性和可能性(即危险)来给它们分类,这称作基本危险分析(PHA)。PHA 使用基本危险表(Prelinminary Hazard List,简称 PHL)来识别系统的危险领域。PHL 建立于系统的概念定义生命周期阶段期间。后来的系统与软件危险分析都将用到 PHL。

PHA 开始于早期系统生命周期阶段考虑需求时,结束于组件级的系统危险分析能够开始时。PHA 包括检查软硬件控制间的接口、会导致系统失误(Miscap)的软件、控制软件的命令和响应的系统设计规则及软件错误安全设计考虑等。

一旦危险被识别出来,对它们指定危险关键性和危险可能性。危险关键性包括由于危险可能导致的最坏的失误(Miscap)的定性测量。在一个系统的早期设计阶段定性地根据关键性识别和分类是足够的。接着,定性的可能性等级可以重新分配给危险。

典型的定性的可能性的目录可能包括频率(经常发生的),偶然率(系统周期中见发生几次),些微率(在项目周期中有时发生的),极少率(不太可能发生,但还是有可能),极端少率(发生可能性不能与 0 混淆)和物理上的不可能。定性可能性评估常用危险发生率来表示。

系统危险分析(System Hazard Analysis,简称 SHA)是两重性的,一方面, SHA 识别那些由于它的功能(或缺乏某项功能)而导致一个危险的系统的组件。SHA 的这方面考察组件操作或失败是怎样影响系统。它开始于充分设计这些组件时,当设计成熟时更新。假如软件影响一个组件,软件生命周期的每一个产品对于 SHA 的这部分都是一个输入。它针对具体的软件生命周期过程中提出软件的问题,并随着系统或软件设计的变化更新。

另一方面,SHA 也集中于组件接口并将系统作为一个整体来分析系统危险,从而识别危险。SHA 检查系统操作和失败模式效果是怎样影响系统和它的组件的安全的。当系统设计成熟时系统级的 SHA 开始,结束于设计修订,并一直修改直到系统设计完成。操作和支持危险分析(O&SHA)识别和评估由于人为执行的操作引起的危险。它开始于系统测试和综合周期阶段之前(应足够早到提供输入给设计)。O&SHA 识别对消除和减轻危险有必要的安全需求。它也包括识别更改所需要的需求和一个软件控制的操作所必要的警告和警急过程。

一旦基本的危险分析完成,软件安全分析便可以开始了。软件安全分析在软件生命周期内的不同阶段的工作如图 1 所示。软件安全建模和分析技术用于识别软件危险和对于安全至关重要的单一或多错误序列,确定软件需求(包括实时需求、分析和测试软件安全),对安全性至关重要的系统部分进行软件安全分析和证实。

三、软件安全分析和建模

如图 1,软件安全分析和建模的工作很多。其中的三个比较重要的部分是软件需求分析、证实和确证、测试。下面将依次介绍这三个部分:

1. 软件安全需求分析

确定安全需求是一件十分困难的事情,这是软件问题的一个源泉且可能对于安全性来说是最重要的。这是由于不充分的设计预见、不正确的设计态度和特殊的错误及需求分析本质上的困难造成的。

当功能需求集中于系统该作什么时,安全需求必须包括系统不该做什么——包括消除和控制系统的危险以及 miscap 事故发生时限制损失的方式。安全需求的重要部分是软件和系统如何安全地发生错误和可容忍的程序错误说明。在有限的上下文里,已经可以运用一些技术识别和分析软件安全需求,如错误树分析(FTA),软件错误树分析(SFTA),实时时序逻辑(RTL),时序 Petri 网(TPN)等。本文以 FTA 为例来说明。

FTA 开始于基本危险分析已经开始时。对于每一个危险事件都必须构造一个独立的错误树。FTA 假设一个已识别出的危险已经发生,然后反过来查找危险的起因。一个基本的过程用于假设危险已发生并用于确定可能的原因集合。树的根是要分析的危险事件即损失事

件。损失事件的必要的前提条件构成树的下一级,并用"与"或"或"关系来描述。每个子节点用相似的方式扩展。扩展这个系统错误树直到它的所有叶子含有最低级别的不用再深入分析的基本事件(即事件的可能性可以计算出来)或由于某种原因不能分析下去。

例如一个火车的十字路口系统的一个功能是当火车穿过十字路口时打开警示灯并让灯开着直到火车过去。由此画出的 FTA 图如图 2 所示。

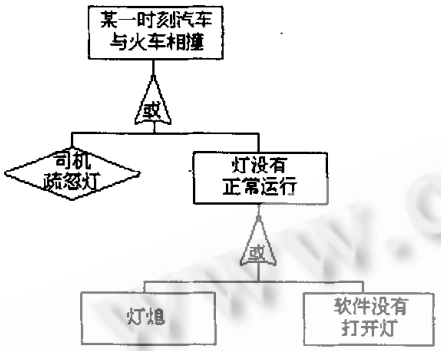

图 2 系统错误树分析

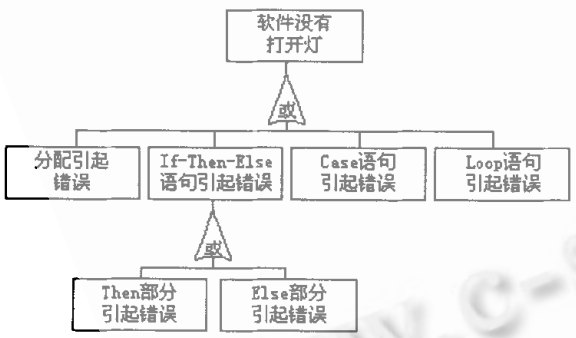

图 3 软件错误树分析

软件错误树分析(SFTA)起源于 FTA(Fault Tree Analysis)。在两者间最显著的不同是 FTA 分析硬件而 SFTA 分析软件。SFTA 可以用于连接 FTA(硬件系统),这样使得软件错误树被组合进来分析整个系统。这是有意义的,因为许多危险是由一个软件错误和硬件错误的组合而引起的。总之,SFTA 可以用来确定软件安全需求,侦测软件逻辑错误,识别多种错误序列涉及会导致危险的系统的各个部分(硬件、人和软件)。

2. 软件安全的证实和确证

安全的证据包括下述选择:

(1)表明一个错误不可能发生,即软件不会处于一个不安全的状态且不会指挥系统进入一个不安全的状态。

(2)表明假如一个软件错误发生了,它不是危险的。

Anderson 和 Witty 在 1978 年已经提供了一个早期的尝试去说明安全证据的含义,即等价性定义。即假如一个原始的需求用 P 表示,那么可以选择一个这样的需求 Q 来代替它:

(1)任何一个程序 P 能执行,同样 Q 也能执行;

(2)Q 描述了程序可接受的行为。

需求 Q 的选择是为了便于更简单的正确性证明,这样的证据称为(充足)相当的证据。安全的证实和确证中可以应用到几种技术是:软件错误树分析(SFTA),软件一般模式分析(SCMA),隐蔽软件分析(SSA)等。

四、系统与软件安全设计

一旦危险系统状态已经识别出来且软件安全需求确定。系统必须最小化危险且满足这些需求。不可能仅靠分析和证实确保一个系统的安全。

安全设计的选择 1:防止或最小化危险发生率。

安全设计的选择 2:假如危险发生,使用自动化的安全设备控制危险发生率。

安全设计的选择 3:提供有关危险的警告设备,过程,和有助于人为反应的训练。

安全设计不能是一个事后的思考,必须在一开始就应该考虑到。

安全设计的两个一般性的原则:(1)设计时提供的证明措施必须具有最小化的所需的证实量和尽量简化的证实过程;(2)考虑提高安全性的设计时必须仔细地计算出来增加的有关的复杂性。

参考文献

[1] MIL - STD - 882B, "System Safety Program Requirements", Department of Defense, 30 March 1984

[2] NASA - STD - 8719.13A, "NASA Software Safety Standard", September 15, 1997

[3] Nancy G., Leveson, "Software Safety: Why, What and How", computing surveys, Vol. 18, No. 2, June 1986

[4] 龚正虎等编著,《协议通信工程学》,国防科技大学出版社。

[5] Leveson, Nancy G., Stephen S. Cha and Timonthy J. Shomeall, "Safety Verification

(来稿时间:1999 年 6 月)