第 38 卷 第 18 期
Vol.38 No.18

计 算 机 工 程
Computer Engineering

2012 年 9 月
September 2012

• 软件技术与数据库 •　　文章编号：1000—3428(2012)18—0057—04　　文献标识码：A　　中图分类号：TP311.1

一种扩展的 Android 应用权限管理模型

鲍可进，彭 钊

(江苏大学计算机科学与通信工程学院，江苏 镇江 212013)

摘 要：现有 Android 移动操作系统不支持用户自由分配已安装的应用权限。为解决该问题，提出一种细粒度的 Android 应用权限管理模型。该模型在保证系统安全性的前提下，对现有 Android 应用权限机制的框架层和应用层进行修改和扩展，使用户可以通过 GUI 界面按需分配系统中已安装的应用权限。实验结果表明，该模型能满足用户的 Android 应用权限管理需求，并且系统性能损失较小。

关键词：安卓；移动操作系统；移动安全；权限控制；安全机制；组件间通信

An Extended Android Application Permission Management Model

BAO Ke-jin, PENG Zhao

(School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang 212013, China)

【Abstract】As existing Android system does not support users to customize the permissions of applications in it, this paper extends the existing Android application permission management mechanism, and proposes a fine-grained Android application permission management model compared to the existing application permission management ability. It retains the existing Android security mechanism, and users can freely customize the permissions of applications that are installed in it through the friendly GUI. Experimental results show that the model can provide users a practical and fine-grained Android application permissions management function, and it has less loss of the system performance.

【Key words】Android; mobile operating system; mobile security; permission control; security mechanism; inter-components communication

DOI: 10.3969/j.issn.1000-3428.2012.18.015

1 概述

随着近年来移动互联网的高速发展，作为移动互联网主力终端的智能手机也迅猛发展，逐渐融入日常生活，成为同 PC 一样不可或缺的互联网终端产品。但是随着智能手机的普及，智能手机的安全问题也逐渐突显，据中科院最新调研报告显示，当前 68.6%的手机用户正面临移动安全威胁，而市场研究机构 IDC 指出，智能手机的销售量已超过 PC，将成为犯罪分子攻击的下一个目标。

智能手机的安全威胁很大一部分来自个人隐私的泄密及各种恶意扣费软件，这些恶意软件大多在用户不知情的情况下后台执行，究其原因是恶意应用的权限滥用。在这样的需求下，本文对现有 Android 操作系统进行扩展，在保证其现有安全性的前提下，实现一个轻量级的细粒度的应用权限管理模型，使用户能在一定限制条件下按需对 Android 系统中所安装应用的权限进行分配。

2 相关研究

近年来，随着 Android 平台的快速发展，关于 Android 安全的相关研究也越来越多。文献[1-2]对 Android 系统的安全机制进行了全面深入的剖析，指出现有 Android 系统的不足并提出可行的安全增强方向，该文献对本文模型的提出起到一定的指导意义。文献[3]提出一个 SAINT 模型，该模型允许应用开发者设置应用在安装时和运行时的权限限制，与本文模型一样弥补了现有 Android 系统应用安装后权限不可更改的缺陷，但本文模型把运行时的权限限制交给用户设置，而不是开发者，从用户的角度来讲更加实用。文献[4]提出与本文很类似功能的模型 Apex，该模型允许用户设置安装时和运行时的应用权限，并支持基于时间、地理位置等条件触发权限策略的改变，与本文模型相比，该模型为用户提供的专门针对某一特定应用的权限编辑能力有限且较繁琐。文献[5]通过结合 Android 平台数据安全的特性与平台提供的广播机制，针对短信、电话、网络这 3 种数据与资源的安全访问建立动态的监控的模型，从而在一定程度上保证了这些脆弱资源的安全。

3 现有 Android 应用权限管理机制

3.1 Andoird 简介

Android 是 Google 协同开放手持设备联盟(Open Handset Alliance)在 2007 年 11 月 5 日发布的一种基于 Linux 内核的开源操作系统，主要用于移动便携设备。该系统架构由 4 个层次组成，从底至顶分别是 Linux 内核层、硬件抽象层、Android 框架层和应用程序层[6]。本文的相

作者简介：鲍可进(1958－)，男，教授，主研方向：嵌入式系统，智能控制；彭 钊，硕士研究生

收稿日期：2011-11-07　　**修回日期：**2012-01-10　　**E-mail：**pengzhao.lh@gmail.com

关工作主要集中在框架层(扩展的应用权限审核机制)和应用程序层(用户编辑权限的交互应用)。

3.2 Android 安全机制

Android 安全机制体现在 2 层：Linux 内核级别和 Android 框架层[1]。Linux 内核层为 Android 安全机制提供最底层的支持：每一个应用安装后，Linux 将为其分配一个固定永久不变的 UID(User Identification)(其中，普通 Android 应用程序的 UID 从 10 000 开始分配，10 000 以下是系统进程 UID)，Linux 内核保证每个 UID 对应的应用只在自己的进程和数据空间内运行；Android 框架层实现了应用级的强制访问控制(Mandatory Access Control, MAC)，体现为每个应用的权限在授权后是固定的、不可由用户自主更改的。应用权限由应用开发者在 AndroidManifest.xml 配置文件中定义。

权限的授予只在应用安装时发生，用户在安装过程中一次性授权 AndroidManifest.xml 中的权限请求，其后不可再更改。在应用运行过程中，任何未在 AndroidManifest.xml 中定义的权限操作都会被系统拒绝。Android 系统内置 100 多种可供应用申请的权限，详细的列表可参见 Android 开发者官方网站。

3.3 现有 Android 应用权限管理机制的弊端

虽然现有的 Android 系统提供了完善的安全机制，但是在应用权限的管理方便方面还存在以下不足：

(1)“All-Or-None”的应用授权模式：虽然在应用的安装过程中安装器会弹出询问窗体让用户决定是否同意所有的权限申请，用户如果选择同意就会授权所有的权限申请，如果选择拒绝就会中止安装，从而无法使用该应用的任何功能。这样用户无法根据自己的需求自由定制授权应用的权限。

(2)一旦应用安装完成，授予的权限将不可再更改。用户唯一去掉已授权权限的方法就是卸载该应用。

(3)ADB(Android Debug Bridge)是 Android SDK 中提供的一个使手机与 PC 通信的通用调试工具，ADB 以 root 身份登录 Android 系统，通过 ADB 安装应用的过程中它会在不告知用户的情况下默认授权应用的所有权限请求，同样安装后用户不可更改授予的权限。

这些缺陷为系统实用性和安全性带来了很大的问题，比如，用户可能只想使用某个应用的某几个功能，而不想让它持有可能泄漏自己隐私的诸如短信读取、邮件读取等功能；再如用户对某一应用的信任度会随着时间的推移而变化，他后来可能会想限制或者放宽其权限。而现有的权限机制无法满足用户的这些需求。

4 扩展的 Android 应用权限管理模型

4.1 模型简介

针对现有 Android 系统在应用权限管理上的不足，对现有的系统进行了扩展。对 Android 安全的扩展可以在 Linux 内核层集成现有的 Linux 访问控制框架[7]，或是扩展现有的 Android 框架层。模型实现正是基于后者，在现有的 Android 权限检查机制前扩展了一个权限检查机制，实现了一种在应用权限控制上较现有 Android 应用权限控制机制更细粒度的模型。该模型在保证系统现有安全性的前提下，提供给用户自由按需分配已安装应用权限集合的能力，并提供应用权限白名单和黑名单的功能。

在现有 Android 安全机制下，应用在安装完成后，用户无法再编辑其所授予的权限，基于模型扩展后的系统允许用户任意编辑(增加、删除、修改或分组)应用的权限集合，当然，用户可编辑的权限集合必须是应用在安装时被授予权限集合的一个子集，只有这样才能保证在不损失现有系统安全性的前提下，对现有系统的安全性和实用性做出改进。另外，白名单功能允许用户把自己足够信任的应用放入白名单分组，白名单分组内应用发起的所有调用请求都无须经过该模型的权限审核机制验证而直接允许；同理，黑名单分组中应用的任何调用请求都会被该模型拒绝。

该模型在设计过程中遵循“与现有系统兼容，注重性能与用户体验”的设计原则。例如，出于对执行时效率和存储空间的考虑，模型只存储用户拒绝的权限，而不是像其他相关模型一样[4]，直接存储系统中所有应用的所有权限，这使得模型的存储数据少得多，不但提高了查询效率，而且节省了数据存储空间。

4.2 模型架构

该模型的整体架构如图 1 所示，大虚线框内为模型对现有 Android 权限审核机制的扩展，虚线箭头线 A 为系统扩展前的权限审核流程。从图 1 可以看出，该模型是在现有 Android 权限审核机制之前插入一个新的权限审核机制。该模型横跨应用层和 Android 框架层 2 个层次，其中在应用层实现了用户与模型交互的 UI 交互界面，用户通过该交互界面编辑应用的权限集合；在 Android 框架层实现了权限审核、权限管理与存储、动作执行等需求，主要由权限检查器、权限管理器、权限存储器、动作执行者 4 个模块构成。

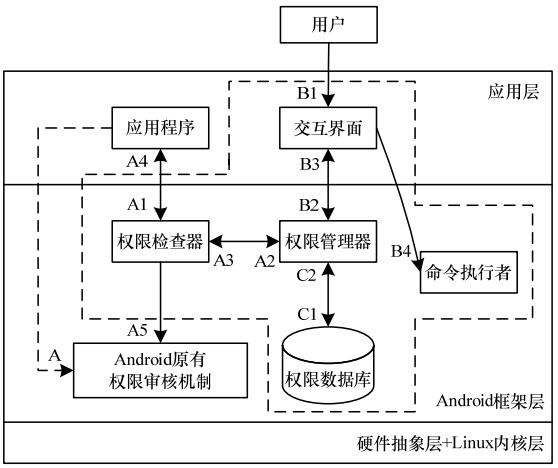

图 1 细粒度 Android 应用权限管理模型架构

应用程序：Android 系统中实现某些特定功能的软件称为应用程序(Application)，或简称应用。应用程序是对目标组件调用请求的发起者(箭头 A1 或者虚线箭头 A)，下文对目标组件的调用请求简称为请求。

交互界面：交互界面是一个基于 Android 框架开发的处在应用层的应用程序，该应用可以让用户通过一个人性化的 UI 界面，系统中任意应用的授权权限集合进行增加、删除、修改、分组等操作(箭头 B1、B2、B3)。用户在交互界面的编辑结果将通过权限管理器保存到权限数据库(箭头 B1->B2->C1)。交互界面还有一个额外的功能就是发送应用进程中止命令给动作执行者(箭头 B4)。

命令执行者：动作执行者可以响应交互界面下达的命令，结束当前正在运行的任意应用。设计这样一个模块主要是考虑到这样的情况：某一时刻用户通过用户界面删除应用 A 的权限 P，但是这时 A 正在运行且正使用 P 权限，权限管理器发现这一情况后就会发送命令给动作执行者，命令其结束应用 A，动作执行者执行结束 A 的命令。

权限管理器：权限管理器主要负责与权限数据库相关的操作请求，出于安全性考虑，权限管理器是本模型中唯一可访问权限数据库的模块。权限管理器内部包含对权限数据库的增、删、改等基本操作(箭头 C1、C2)，可响应权限检查器和交互界面的权限查询、编辑请求(箭头 A2、A3)。

权限检查器：权限检查器是模型中应用权限审核机制的入口，当应用程序发起应用是否拥有请求的权限(箭头 A2、A3)，然后根据查询结果决定直接拒绝该请求(箭头 A4)还是将该请求继续传递给 Android 现有的权限审核机制(箭头 A5)。

Android 现有权限审核机制：Android 系统现有的独立的、完整的应用权限审核机制。模型为了保证与现有 Android 系统的兼容，保留了 Android 现有的权限审核机制，请求在通过扩展的权限审核机制审核后，将被继续往下传递，进入 Android 现有的权限审核机制。

4.3 具体实现

在 Android 系统中，应用对目标组件发出调用请求后，Android 的权限审核机制会在该请求真正执行之前被强制触发：ActivityManagerService 类接收到请求后设置相关的环境变量，并调用其内部 checkPermission 方法检查请求发起者的权限。所以，本文模型就是在 checkcheck Permission 方法的最开始部分放置一个引导函数，这样每次系统检查请求者权限的时候，都会首先被引入扩展模块，由扩展模块暂时接管权限的审核。

模型中处在 Android 框架层的扩展模块是对现有 Android 源码的修改和扩展，应用层的交互界面是一个基于 Android 框架的应用程序，各模块使用 Java 语言编码来实现。

权限检查器：作为模型中整个扩展模块的入口，权限检查器是一个放置在 checkcheckPermission 方法头部的钩子函数，这样可以保证第一时间截获权限审核请求。权限检查器根据截获的请求设置相关环境变量后与权限检查器交互以获得权限审核结果。如果请求权限审核未通过，权限检查器就通过 SecurityException 类抛出带有审核拒绝详细信息的异常，SecurityException 类抛出的异常最终会被请求的应用所接收并触发错误提示窗口呈现给用户；如果权限审核通过，权限检查器将把其截获的请求原封不动的继续往下传递给现有的 Android 权限审核机制。

权限数据库：在实现上，权限数据库采用 Android 运行时中集成的 SQLite 作为数据库支持。SQLite 是一款非常流行的嵌入式数据库，它支持 SQL 查询，并且只用很少的内存。权限数据库中分别存储了一张用于记录被用户拒绝的应用权限的表和一张实现应用黑白名单分组的表。

权限管理器：权限管理器是整个模型中唯一对权限数据库具有访问能力的模块，模块内部实现了所有的数据库查询修改操作，所以这里的权限管理器很像软件设计方法中 3 层架构中的数据层，或称为数据库驱动。

交互界面：基于 Android 框架开发的应用，由 3 个 Activity 组成，分别是用于列表显示系统中所有安装应用的 ListActivity、用于编辑应用权限的 EditActivity、实现黑白名单分组编辑的 GroupActivity。所有对模型的权限编辑都在这 3 个可视的 Activity 内完成。另外，当用户在 EditActivity 内删除某个应用的权限后，交互界面就发送一条命令给动作执行者，命令动作执行者结束该应用进程。

动作执行者：接收到交互界面发来的应用结束命令后，动作执行者调用 activityManager 类中的 getRunning Services 和 getRunningTasks 方法检查待结束的应用当前是否处于运行状态，如果正在运行，则调用 activityManager 类中的 kill BackgroundProcesses 方法结束该应用；如果没有在运行，则不执行任何动作。

5 实验验证

5.1 实验环境

本文模型基于目前较新的 Android2.3.3 源码修改实现，修改后的源码按照 Android 官方文档中的说明编译成镜像[8]，这里不再详述。为了保证实验环境的普遍性，采用 Android SDK 自带的模拟器作为实验平台，实验分别对比模拟器装载原生 Android2.3.3 镜像和前面编译的扩展后的 Android2.3.3 镜像的功能和性能差别。为便于表述，下文分别简称为扩展前和扩展后。

5.2 功能测试

为了对扩展后的 Android 系统进行功能验证，使用 Android 平台上的某款词典作为测试应用，该应用在安装时申请了存储、网络通信、手机通话等权限，主要对网络通信权限的控制能力进行测试。网络通信权限(android.permission.INTERNET)授予手机完全的互联网访问权限，该词典在查询单词网络释义时需要这个权限。

实验方案：(1)首先在上述用户交互应用的权限编辑界

面中去除改词典的网络通信权限，然后查询某个单词的网络释义。由于查询网络释义需要网络通信权限，而该权限已经被去除，因此正如所期望的，得到来自扩展模块权限审核失败的提示信息，见图 2。(2)在交互应用的黑白名单编辑界面把改词典加入白名单，其又能正常查询网络释义。(3)在交互应用的黑白名单编辑界面把改词典加入黑名单，其又无法使用网络释义功能。

图 2 未通过本文模型权限审核的界面

另外，从图 3 可以看出，该词典申请了手机通话权限，但是作为一款词典应用，这样的权限请求让用户可疑，所以，可以在图 3 所示的权限列表中去掉其可疑的手机通话权限，这样就能确保用户的隐私不被泄漏。这也是本文模型设计的初衷。

图 3 本文模型提供的应用权限编辑界面

5.3 性能测试

本文模型是在 Android 现有权限审核机制基础上的扩展，在权限审核时运行状态只比现有机制增加了权限检查器、权限管理器、权限存储器 3 个模块，所以，预估无论在内存还是 CPU 消耗上，并不会比现有系统有大幅增长。从表 1 可以看出，资源消耗增长确实是微乎其微，增长只占到剩余内存的 0.8%左右，几乎可以忽略不计，本文不再详述。本文主要对系统扩展前后的权限审核机制时间消耗做出实验对比。

表 1 扩展前后系统剩余内存对比 MB

系统	剩余内存
扩展前	14.25
扩展后	14.13

实验方案：由于网络存在很多不稳定因素，因此时间消耗对比测试并没有像上面的功能测试一样测试网络通信，而是重新编写一个简单的测试应用，该测试应用连续 1 000 次重复地在手机 SDCard 中创建一个空文件、并立刻删除，1 000 次任务执行完成后，应用记录第 1 次文件创建开始到最后一次文件删除完成的总耗时，进而计算出每个创建、删除周期的平均耗时。在 SDCard 中创建、删除文件需要用到权限 WRITE_EXTERNAL_STORAGE(存储卡写权限)，对比实验分别在权限数据库中存在 0 条规则、20 条规则、50 条规则、100 条规则 4 种条件下进行，并在每种条件下对比 3 个不同系统环境的测试应用时间消耗，这 3 个系统环境分别为：原生 Android 系统，基于本文模型扩展后用户授予测试应用存储卡写权限的 Android 系统(简称扩展有写权限 Android)，基于本文模型扩展后用户不授予测试应用存储卡写权限的 Android 系统(简称扩展无写权限 Android)。最后用测得的时间消耗除以重复次数 1 000 次即可得到每个文件访问周期(创建、删除文件各一次)的时间消耗，所得的对比数据如图 4 所示。

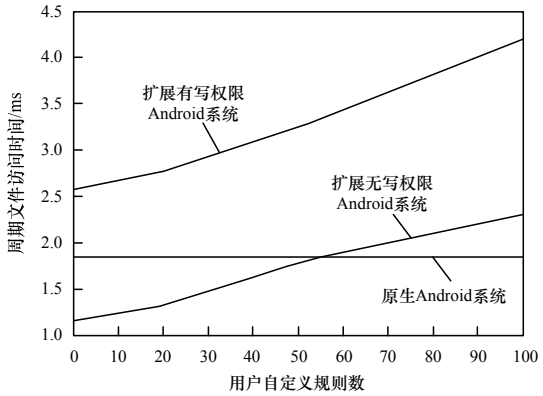

图 4 3 种测试环境下的周期文件访问时间比较

从图 4 可以看出，原生 Android 系统的访问消耗不会受用户自定义权限规则数的影响；在扩展有写权限 Android 环境下，随着权限规则数的增加，时间消耗几乎呈线性增长，这是因为随着权限规则数的增加，权限管理器对权限数据库的检索时间会相应增长；在扩展无权限 Android 环境下，在权限规则数不多时，周期文件访问时间出现了比原生 Android 系统还少的情况，主要原因是当权限规则数很少时，权限管理器对权限数据库的检索时间比原生 Android 系统的权限审核时间短，而权限检查器收到权限管理器的查询结果后直接跳过 Android 现有的权限审核机制，返回拒绝的响应给应用请求。但是随着权限规则数的增加，这一情况又会有所变化。

虽然图 4 上显示规则数增多时，周期文件访问时间会大比例增长，但是实际上，这样的时间增长从数值上来说是很小的，如 100 条权限规则时，扩展无写权限 Android 系统比原生 Android 时间消耗增长 4.197–1.840=2.357 ms，这个时间增长从用户角度来讲，几乎是忽略不计的。

(下转第 64 页)

(2)测试漏洞的时间

为保证测试的效率，测试时一般以 1 000 个畸形数据文件为单位，分组进行测试，当前组测试完毕后，如无漏洞则进行下一组测试，有漏洞程序自动记下导致程序异常的畸形数据报文文件。在测试过程中，对于包含校时数据报文以及气象信息报文的这 2 类软件没有出现异常，但对于站号坐标报文及目标信息报文则出现了程序异常情况。以目标信息报文为例，如表 2 所示。

表 2 测试站号坐标报文漏洞所消耗的时间 s

测试文件编号	有无漏洞	消耗时间
0~1000	无	21.277
1001~2000	无	21.239
2001~3000	无	21.243
3001~4000	无	21.256
4001~5000	无	21.251
5001~6000	无	21.243
6001~7000	无	21.275
7001~8000	有(7168.txt)	3.271
	有(7424.txt)	7.355
8001~9000	无	21.283

4.3 测试结论

从目前对 4 类军用报文软件初步测试来看，系统已经具备了畸形数据创建、漏洞发掘等功能。目前系统创建畸形数据性能较好，速度基本能够满足用户需求。被测软件有 2 类软件被测出漏洞。另外，依据出错的畸形数据文件，使用 OllyICE 对报文软件(有源代码条件下)进行分析，发现这些错误主要集中于整形溢出、缓冲区溢出以及特殊字符串处理的函数中，如 strcpy、strcat、atoi、atof 以及 Format 等函数，证明了系统的有效性和实用性。

5 结束语

本文在分析军用报文格式及其软件特点基础上，利用 Fuzzing 技术设计并开发了军用报文软件漏洞发掘系统 MTSFuzzer。初步测试表明，该系统能够发掘报文软件漏洞，可作为进一步提高军用软件可靠性的安全检测工具。今后将针对长报文畸形数据构建效率以及基于 XML 格式类报文软件的漏洞发掘问题进行研究。

参考文献

[1] 吴志勇, 王红川, 孙乐昌, 等. Fuzzing 技术综述[J]. 计算机应用研究, 2010, 27(3): 829-831.

[2] 李宗蕾. Fuzzing 工具的设计与实现[D]. 北京: 北京邮电大学, 2010.

[3] 邵 林, 张小松, 苏恩标. 一种基于 Fuzzing 技术的漏洞发掘新思路[J]. 计算机应用研究, 2009, 26(3): 1086-1088.

[4] 胡文涛. 基于 Fuzzing 的网络服务型软件的漏洞挖掘研究[D]. 上海: 上海交通大学, 2008.

[5] SPIKE Proxy[EB/OL]. (2009-06-15). http://www.immunitysec.com/resources-freesoftware.shtml.

[6] Sutton M, Greene A, Amini P. 模糊测试——强制性安全漏洞发掘[M]. 李 虎, 译. 北京: 机械工业出版社, 2009.

[7] 黄 奕. 基于模糊测试的软件安全漏洞发掘技术研究[D]. 合肥: 中国科学技术大学, 2010.

[8] 王继刚, 曲慧文. 软件漏洞发掘与安全防范实战[M]. 北京: 人民邮电出版社, 2010.

编辑 陆燕菲

(上接第 60 页)

6 结束语

本文针对现有 Android 系统在应用权限管理机制上的缺陷，提出了细粒度的 Android 应用权限管理模型。该模型是对现有 Android 安全机制所存在缺陷的有效扩展，能够在保证现有系统安全性的前提下，允许用户自由定制系统中已安装应用的权限，且从实验效果来看，与原系统相比性能消耗增加甚微，很好地达到模型的预期效果。下一步将对本文模型继续进行功能与易用性上的扩展与完善，使其可以部署到现有 Android 手机中。

参考文献

[1] Android in Wikipedia Site[EB/OL]. (2008-04-13). http://en.wikipedia.org/wiki/Android_(operating_system).

[2] Enck W, Ongtang M, McDaniel P, et al. Understanding Android Security[J]. IEEE Security and Privacy Magazine, 2009, 7(1): 50-57.

[3] 陈汉章, 张玉清. 访问控制框架及其在 Linux 中的应用研究[J]. 计算机应用研究, 2007, 24(4): 217-219, 222.

[4] Nauman M, Khan S. Design and Implementation of a Fine-grained Resource Usage Model for the Android Platform[J]. International Journal on Artificial Intelligence Tools, 2010, 8(4): 440-472.

[5] 蔡罗成. Android 后台监听实现机制浅析[J]. 信息安全与通信保密, 2010, (6): 39-41.

[6] Shabtai A, Fledel Y, Kanonov U, et al. Google Android: A Comprehensive Security Assessment[J]. IEEE Security and Privacy Magazine, 2010, 8(2): 35-44.

[7] Ongtang M, McLaughlin S, Enck W, et al. Semantically Rich Application-centric Security in Android[C]//Proceedings of the Annual Computer Security Applications Conference. [S. l.]: IEEE Press, 2009.

[8] Android's Official Building Tutorial by Google[EB/OL]. (2010-07-16). http://source.android.com/source/building.html.

编辑 陆燕菲