VII. RELATED WORK

The most similar work to this study is by McDonnell, Ray,

and Kim [29], who study the use of deprecated API methods

and the adoption of updated API methods in ten open source

Android apps. Unlike our study, which focuses on target API

levels, they focus on the usage of methods changed in new

API levels. Targeting an API level is not dependent on using

methods added in that platform version and apps should target

the most current API level even if they do not use any newly

added methods. Their results say nothing about the security

consequences of outdated apps but do show that developers

are slow to adapt to the changing Android platform.

The target fragmentation problem has been discussed in

relation to specific vulnerabilities in several studies. Thomas

et al. [5] study the changes in Android 17 that closed the

JavaScript Interface vulnerability in depth. Their study focuses

on the slow adoption of new devices and its effect on the

lifetime of the vulnerability but they also find that 22% of

studied apps use the JavaScript Interface and targeted API

levels below 17. Mutchler et al. [4] identify apps that load

untrusted content in WebView and note that the JavaScript

Interface vulnerability puts these apps at risk.

The vulnerabilities mentioned in this paper have been studied without mention of target fragmentation. Lu et al. [30]

build a static analyzer to identify vulnerabilities including

Service Hijacking. Georgiev, Jana, and Shmatikov [15] provide

a tool to prevent attacks through the JavaScript Interface. Chin

and Wagner [13] statically analyze apps and find unsafe use

of file: URLs. Jin et al. [31] build a tool to detect a variety

of XSS-like vulnerabilities in WebView apps.

Because both apps and devices must be updated in order to

take advantage of new security features, target fragmentation

and device fragmentation are linked. Thomas, Beresford, and

Rice [2] study device fragmentation using volunteers who

install their device monitor app. They find that 88% of devices

are vulnerable to at least one of selected vulnerabilities and

that devices are updated infrequently (1.26 times per year on

average). Zhou et al. [32] find that more than 1,000 of 2,423

factory images can be exploited through misconfigurations of

device drivers. Xing et al. [33] identify how apps can exploit

the OS update process to obtain sensitive system permissions.

Mulliner et al. [34] provide a scalable method for applying

third party patches to vulnerable Android devices.

The app update process has also been studied. Moller et ¨

al. [35] investigate the update patterns of Android users and

find that only half of users install an app update within

one week of the update being published. McIlroy, Ali, and

Hassan [36] mine update data from 10,713 apps and find that

only 1% of apps receive at least one update per week.

Several other studies analyze large datasets of Android apps.

Viennot, Garcia, and Nieh [10] crawl, download, and analyze

1,100,000 apps to obtain statistics about permission and library

distributions as well as identify apps that unsafely embed

credentials. Other studies [37, 38] also study permission and

library usage. Kavaler et al. [39] compare usage of Android

classes and questions asked on StackOverflow.

VII. 相关工作

这项研究最相似的工作是由McDonnell, Ray,and Kim [29],——研究使用过时的 API 方法在十个开源Android 应用程序中采用了更新的 API 方法。与我们的研究不同, 它侧重于目标 API级别, 他们侧重于在新的API 级别方法中更改的用法。针对 API 级别的目标不依赖于使用在该平台版本和应用程序中添加的方法。针对最新的 API 级别, 即使它们不使用任何新的添加的方法。他们的结果说过时的应用程序与安全无关, 但确实表明, 开发人员很慢, 无法适应不断变化的 Android 平台。

对目标碎片问题进行了讨论几个研究中的特定漏洞的关系。Thomaset al. [5] 研究了 Android API 17 关闭JavaScript 接口漏洞的深度的变化。他们重点研究新设备的缓慢采用及其对生存期的漏洞, 但他们也发现, 22% 研究的应用程序使用 JavaScript 接口和目标 API等级低于17。Mutchler et al. [4] 识别加载的应用程序web 中不受信任的内容, 并注意 JavaScript接口漏洞使这些应用程序面临风险。

本文中提到的漏洞已被研究, 而没有提到目标碎片。Lu et al.[30]。

构建静态分析器来识别漏洞, 包括服务劫持。Georgiev, Jana, 和 Shmatikov [15] 提供

一种防止通过 JavaScript 接口进行攻击的工具。 Chin 和Wagner [13] 静态分析应用程序和发现不安全的使用的文件: 网址。Jin et al. [31] 建立一个工具来检测各种web 应用程序中的类似 XSS 的漏洞。

因为必须更新应用程序和设备才能利用新的安全功能, 目标碎片和设备碎片链接。Thomas, Beresford 和Rice [2]研究设备碎片使用志愿者安装他们的设备监视器应用程序。他们发现88% 的设备容易受到至少一个选定的漏洞和该设备不经常更新 (平均每年1.26 次)。 Zhou et al. [32] 在2423工厂中超过1000家图像可以通过错误设备驱动程序。Xing et al.. [33] 识别应用程序如何利用操作系统更新过程以获取敏感的系统权限。Mulliner et al. [34]提供了一种可伸缩的应用方法通过第三方补丁到易受攻击的 Android 设备。

我们还研究了 app 更新过程。Moller et al [35] 调查 Android 用户的更新模式发现只有一半的用户在应用程序更新发布更新的一周内安装了。McIlroy Ali和Hassan [36]从10713个应用的更新数据里发现只有1% 的应用程序每周至少更新一次。

其他一些研究分析了 Android 应用的大数据采集。 Garcia和Nieh [10] 抓取、下载和分析110万应用程序获取有关权限和库的统计信息分布以及识别安全嵌入的应用程序凭据.其他研究 [37, 38] 也研究允许和库使用。Kavaler et al. [39]比较了使用了 Android在

StackOverflow 的相关课程和问题。

VIII. CONCLUSION

Android apps specify a target API level and run in a

compatibility mode on devices with higher API levels. The

compatibility mode can disable important security changes in

the Android platform. We call the problem of apps targeting

outdated API levels the target fragmentation problem. In this

study we analyze a dataset of more than one million Android

apps collected over four years and show that the large majority

of collected apps target outdated API levels. We examine

the practical implications of target fragmentation on seven

security changes to the Android platform and show that target

fragmentation hamstrings new security features.

We believe that applying security changes in this optional

manner is a flawed approach that sacrifices security at the

altar of compatibility. Developers become a new obstacle

to securing apps and users have no means of ensuring that

their apps target the most current API levels. The target

fragmentation problem is further compounded by the coupling

of security changes and non-security changes. We hope that by

shedding light on this problem, developers can become more

aware of the consequences of targeting outdated API levels

and this flawed design can be reexamined and changed so that

there is less opportunity for Android apps to operate without

access to important security features.

VIII. 结论

Android 应用程序指定一个目标 API 级别, 并在具有较高 API 级别的设备上的兼容模式运行。兼容模式可以禁用Android 平台重要的安全更改。我们叫这个问题为针对过时API级别的应用程序目标碎片化问题。我们花了四年里研究收集分析了超过 100万 Android 应用，显示大部分收集的应用的目标是过时的 API 级别。我们研究目标碎片化对Android 平台七个安全变化的实际影响 , 并显示目标碎片化对新的安全功能的削弱。我们认为, 应用可选的安全更改方式是一个有缺陷的方法, 牺牲安全性来确保兼容性。开发商为了保证应用程序和用户的安全成为新的障碍，没有办法确保他们的应用程序的目标是最新的 API 级别。目标碎片化问题进一步复杂化加剧安全更改和非安全更改。我们希望通过在这个问题上, 开发者可以变得更意识到以过时的 API 级别为目标的后果，对这种有缺陷设计可以重新审视和改变, 以便Android应用运行时不会产生重要的安全功能。