International Journal of Scientific and Technical Advancements
ISSN: 2454-1532

Privilege Escalation Attack on Android and Its Defences

Akshay Bhardwaj[1], A J Singh[2]
[1, 2] Department of Computer Science, Himachal Pradesh University, Himachal Pradesh, India
Email address: akshay117@gmail.com, aj.hpucs@gmail.com

***Abstract*—** Android is most usually utilized stage for cell phones today which gloats of a propelled security model having MAC and sandboxing. These components permit engineers and clients to confine the execution of an application to the benefits allocated. The misuse of vulnerabilities of the project is bound inside the benefit limits of an applications sandbox. Benefit acceleration assaults have developed complex as the utilization of android frameworks have expanded. Various types of components have given some kind of rest to the designers however the security highlight taking care of by the engineers has not helped much. In this paper we talk about the nuts and bolts of the benefit heightening assault and the different strategies used to counter and keep this issue.

***Keywords*—** privilege escalation; attacks; sandboxing; android; security.

I. INTRODUCTION

The prevalence of cell phones and the unfathomable number of the comparing applications makes these stages alluring to assailants. Presently, different types of malware e x i s t f o r cell phone stages; counting android. Most advanced mobile phones depend completely on application sandboxing and favored access for security. Applications are detached and allowed special consents as it were. The application performs activities which are expressly permitted in the application's sandbox. Android checks comparing authorization assignments at runtime. Consequently, an application is not permitted to get to favored assets without having the right consents.

In this paper we demonstrate that Android's sandbox model is adroitly defective and really permits benefit acceleration assaults. This is not an execution bug, yet rather a major defect. In Section 2 we talk about the distinctive Android security instruments and quickly clarify how the benefit acceleration assault can be completed bypassing the sandboxing highlight. In Section 3, we demonstrate the benefit heightening assault. In Section 4, we talk about the related work for the avoidance of this sort of assaults and the different models. In Section 5, we break down the different countermeasures and attractive quality of the arrangements. In Section 6, we finish up in light of perceptions.

II. ANDROID SECURITY MECHANISMS

Here we talk about the Android security components in a word. Optional Access Control (DAC): The DAC component depends on documents (protests) and process (subjects) which access rules. The standards are set and determined to have better get to control system. Sandboxing: Android is a benefit isolated working framework. Sandboxing secludes applications from each other and from framework assets. Framework documents are possessed by either the "framework" or "root" client, while different applications have own extraordinary identifiers.

Authorization Mechanism: Applications may pronounce custom sorts of consent marks to confine access to claim interfaces. Required consents are unequivocally indicated in a Manifest record and are affirmed at establishment time taking into account checks against the marks of the applications pronouncing these authorizations and client affirmation. At runtime, the reference screen checks whether the use of this segment has imperative authorizations.

Segment Encapsulation: Application parts can be indicated as open or private.

Application Signing: Android uses trust based permission mechanism which is verified by third party. But it need not be signed by a certificate authority. It is just a self signed certificate. The certificate is included in its APK file such that the signature is can be validated at install time.

III. PRIVILEGE ESCALATION ATTACK ON ANDROID

Fig 1 delineates a case of benefit acceleration assault on Android. In the figure, there are three applications running in their own particular DVMs. Application 1 has no consents. The segments in application 2 is not monitored by any consents, they are available by segments of some other application. Thus, both segments of application 1 can get to segments 1 in application 2. Application 2 has authorization P1, Therefore, both segments of utilization 2 can get to part 1 of use 3 which is secured by authorization P1.

From the fig we watch that segment 1 of utilization 1 is getting to part 1 of use 2. Be that as it may, it does not have authorization P1, so it is not permitted to get to segment 1 of use 3. Then again, application 2 has authorization P1. Thus, part 1 of utilization 2 is permitted to get to segment 1 of application 3. Along these lines, despite the fact that part 1 of utilization 1 is not permitted to get to segment 1 of application 3, it can get to it by means of segment 1 of use 2. Along these lines, the benefit of use 2 is raised to application 1 for this situation. Keeping in mind the end goal to keep this assault, segment 1 of use 2 ought to uphold that segments getting to it must have authorization P2. This should be possible at code level or by guarding part 1 by authorization P2. Nonetheless,

Akshay Bhardwaj and A J Singh, "Privilege escalation attack on android and its defences," *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 63-66, 2016.

this depends on application designers to play out the requirement at the right places. This is a blunder inclined methodology as application engineers may not be security specialists. [2]

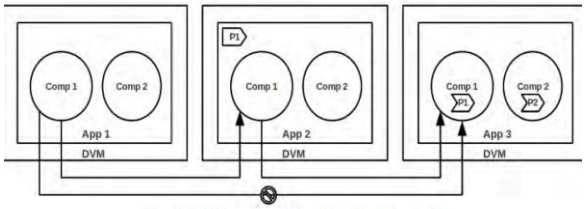

Fig 1: Privilege Escalation Attack on Android

The benefit acceleration assault on Android was initially proposed by Davi et al. [1] in which they showed an case of the assault. They demonstrated that a bona fide application misused at runtime or a malevolent application can heighten allowed consents. Be that as it may, they didn't recommend any guard for the assault in the paper. The most significant works are security augmentations to Android security engineering, in particular Saint [12] and Kirin [6, 7], as they could give a few measures against benefit acceleration assault. Holy person is a strategy augmentation which permits application engineers to characterize complete access control rules for their segments.

Holy person gives a component to guarantee that the guest has in any event the same consents as a callee, as a fundamental condition to avert benefit heightening assaults. Be that as it may, Saint accept that entrance to parts is certainly permitted. It gives certain security against benefit heightening assaults as the application can control which applications can get to it. Be that as it may, this put the weight of authorizing security to application designers which is blunder inclined as the greater part of them are not security specialists. Here we see a comparability with the methodology embraced in C/C++ dialects to delegate limits checking to engineers. Notwithstanding numerous years of examination, assaults that adventure too far out mistakes in C and C++ projects are still common: New programming bugs ceaselessly show up permitting foes to perform runtime abuses.

Accordingly, we accept, comparably it is a mistake inclined way to deal with depend on engineers to characterize right Saint arrangements or to characterize them by any means. Kirin is an application confirmation administration to moderate malware at introduce time. Kirin is apparatus that investigates Show records in the APK of the applications to guarantee that conceded consents conform to a framework wide approach. I t a n a l y s e s p e r m i s i o n s t h a t require unsafe mixes of authorizations [7] or it can break down a superposition of authorizations allowed to all applications introduced on a stage [6]. In any case, their methodology can't recognize applications defenseless against benefit heightening assault. The last approach permits location of uses defenseless against benefit accelerations assaults as it gives a photo of potential information streams crosswise over applications. By the by, as it breaks down potential information streams (as inverse to genuine information streams) furthermore, can't pass judgment on about nearby security authorizations made by applications (by method for reference screen snares), it experiences false positives. Along these lines, it is valuable for manual examination, yet can't give solid choices for programmed security implementations.

Enck et al. [8] depict Android security instruments in points of interest. Blazes [3, 4] gives direction on creating secure applications on the Android stage. Schmidt et al. [14] review instruments which can increment gadget security furthermore demonstrates case of Trojan malware for Android [13]. In [11] Nauman et al. proposed p e rmi s i o n system permitting clients to support a subset of consents the application requires at establishment time, furthermore force imperatives for every consent. Chaudhuri [5] presents a center formal dialect taking into account sort examination of Java develops to depict Android applications dynamically and to reason about their security properties. Shin et al. [18] formalize Android consent system by speaking to it as a state-based model which can be ended up being secure with given security necessities by a hypothesis prover. Barrera et al. [2] propose a philosophy to examine consent use by different applications and gives aftereffects of such an investigation for a determination of 1,100 Android applications. Mulliner[10] presents a strategy for powerlessness investigation (programming bugs) of SMS executions on various versatile stages including Android. . Shabtai et al. [16, 17] give a extensive security evaluation of Android security instruments and recognize high-hazard dangers, however try not to consider a danger of a benefit acceleration assault we depict in this paper. A late piece based benefit heightening assault [9] demonstrates to pick up root benefits by abusing a memory related helplessness dwelling in the Linux piece. Interestingly, our assault does not require helplessness in the Linux part, yet rather depends on a traded off (helpless or malevolent) client space application.

Besides, Shabtai et al. [15] demonstrate to embrace the Linux Security Module (LSM) structure for the Android stage, which mitigates part based benefit heightening assaults, for example, [9]. Jakobsson et al. [19] proposed a product based authentication way to deal with recognize any malware that executes or is enacted by intrudes. Taking into account memory-printing of customer gadgets, it makes it outlandish for malware to cover up in RAM without being identified. TaintDroid [20], in light of corrupt examination, tracks the stream of protection delicate information. At the point when the information are transmitted over the system, clients are told to recognize getting into mischief applications. QUIRE [21] is a security arrangement that can protect against benefit heightening assaults by means of confounded delegate assaults. To address this issue, when there is an Inter Process Communication (IPC) demand between Android applications, QUIRE [21] permits the applications to work with a lessened benefit of its guest by following the call chain of IPCs. Chan [36] et al. proposed a defenselessness checking framework to identify considerate applications which neglect to authorize the extra keeps an eye on authorizations conceded.

Akshay Bhardwaj and A J Singh, “Privilege escalation attack on android and its defences,” *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 63-66, 2016.

IV. PRIVILEGE ATTACKS MEASURES AND CONSIDERATIONS

Table 1: Privilege escalation Attack Analysis

Name of Measure	Type	Technique Used	Effective in	Not Effective in
A Vulnerability checking system [2]	Checking permissions	AndroidManifest.xml file used to define permissions to the application	Classifying which applications are vulnerable to attacks	i. Cannot Detect all kinds of privilege escalation attacks ii. Code level checking is missing
Kirin [6][7]	Checks security critical vulnerable links		Focuses on directly reachable interfaces	Transitive permission attacks still possible
Saint [12]	Application isolation and protection	Fine grained Access Control Model	Prevention of browser attack [9]	i. Additional security features to the application ii. Developers defining permissions are more error-prone
Porscha [23]	Application isolation and protection	Policy-oriented secure content handling	Improvement on model proposed by Saint	i. Data without proper policy tagging can pass through ii. Attacks based on control flows
TaintDroid [20]	Detection and checking of privileges	Dynamic Taint Analysis	Addresses data flows	i. Covert channel exploiting sensitive information [24] ii. Attacks based on control flows iii. Performance penalty is very high
Apex [11]	Application isolation and protection	Deny/accept permission at install time	User friendly and makes Android very flexible	i. Relies on user knowledge ii. Transitive permission attacks still possible
CRePE [25]	Deny/accept permissions granting	Context-Related Policy Enforcement for Android	Can use it as company policy and prevent attacks	i. only few functionalities are blocked ii. Transitive permission attacks still possible
QUIRE [21]	Prevention of attack especially confused deputy attack	Non System centric system policy	It addresses attacks that exploit vulnerable interfaces of trusted applications	i. Failure to detect and prevent colluding unknown attacks ii. Covert channel exploiting possible
IPC Inspection [26]	Prevention of attack especially confused deputy attack		No policy framework so fast and better results. Can be used to detect and prevent at both install time and runtime	i. Failure to detect and prevent colluding unknown attacks ii. Covert channel exploiting possible iii. Only Control channels covered. Data channels can be exploited iii. Neglects permissions classified as normal iv. Less general than other prevention mechanisms v. Not compatible with legacy Android systems
ComDroid [27] Stowaway [28]	Checks security critical vulnerable links	Static Analysis Tool	It warns the developer from broadcasting privacy sensitive data	i. Failure to detect and prevent colluding unknown attacks
XmanDroid [22]	Detection and prevention	System centric system policy	i. Prevents attacks on runtime ii. Detects transitive permission usage over any number of hops iii. Handles exceptional cases (e.g. pending intents and dynamic broadcast receivers)	i. Failure to detect and prevent unknown attacks ii. False detection rates are higher iii. Gets more complex when rate increases

V. CONCLUSION

Non-special applications can heighten authorizations by summoning ineffectively outlined higher-favored applications that don't adequately ensure their interfaces. Albeit as of late proposed augmentations to Android security systems [6,12] mean to address the issue of inadequately planned applications, they experience the ill effects of pragmatic deficiencies. Holy person [12] gives a way to secure interfaces of uses, yet depends on application engineers to characterize Saint arrangements accurately, while Kirin [6] can distinguish information streams permitting benefit acceleration assaults, however brings about false positives.

From the investigation we can suggest that Android's sandbox model neglects to limit limits against runtime assaults as the authorization framework does not check transitive benefit utilization. The majority of the strategies neglect to address intriguing assaults despite the fact that few of them are sufficiently close [22].Looking forward to systems that can deal with a wide range of benefit heightening assaults giving improved security keeping engineers free from considering about Android security issues.

REFERENCES

[1] L. Davi: A. Dmitrienko: A.-R. Sadeghi: M. Winandy (2010); Privilege escalation attacks on Android, ISC.
[2] Patrick P. F. Chan: Lucas C. K. Hui:S.M. Yui (2011); A Privilege Escalation Vulnerability Checking System for Android Applications, IEEE
[3] J. Burns (2008): Developing secure mobile applications for Android.
[4] J. Burns. Black Hat (2009); Mobile application security on Android.
[5] A. Chaudhuri (2009); Language-based security on Android, ACM SIGPLAN, pages 1–7.
[6] W. Enck: M. Ongtang: P. McDaniel (2008); Mitigating Android software misuse before it happens. Technical Report, Pennsylvania State University.
[7] W. Enck: M. Ongtang: P. McDaniel (2009); On lightweight mobile phone application certification, ACM CCS ’09, pages 235–245.
[8] W. Enck: M. Ongtang: P. McDaniel (2009); Understanding Android security, IEEE Security and Privacy, 7(1):50–57
[9] A. Lineberry: D. L. Richardson: T. Wyatt (2010); These aren’t t h e permissions you’re looking for, BlackHat
[10] C. Mulliner (2009); Fuzzing the phone in your phones, Black Hat USA
[11] M. Nauman: S. Khan: X. Zhang (2010); Apex: Extending Android permission model and enforcement with user-defined runtime constraints,ASIACCS ’10, pages 328–332. ACM
[12] M. Ongtang: S.McLaughlin: W. Enck: P. McDaniel (2009); Semantically rich application-centric security in Android. In ACSAC ’09, pages 340–349. IEEE Computer Society.
[13] A-D. Schmidt: H.-G. Schmidt: L.Batyuk: J. H. Clausen: S. A. Camtepe: S. Albayrak: C. Yildizli (2009); Smartphone malware evolution revisited: Android next target?, Malware 2009, pages 1–7.
[14] A.-D. Schmidt: H. G. Schmidt: J. Clausen: K. A. Yuksel: O. Kiraz: A. Camtepe: S. Albayrak (2008); Enhancing security of linux-based Android devices, Lehmann.
[15] A. Shabtai: Y. Fledel: Y. Elovici (2010); Securing Android powered mobile devices using SELinux, IEEE Security and Privacy, 8:36–44.
[16] A. Shabtai: Y. Fledel: U. Kanonov: Y. Elovici: S. Dolev (2009); Google Android: A state-of-the-art review of security mechanisms, CoRR, abs/0912.5101.
[17] A. Shabtai: Y. Fledel: U. Kanonov: Y. Elovici: S. Dolev: C. Glezer (2009); Google Android: A comprehensive security assessment. IEEE Security and Privacy, 8(2):35–44x
[18] W. Shin: S. Kiyomoto: K. Fukushima: T. Tanaka(2010); A formal model to analyze the permission authorization and enforcement in the Android framework invited paper. In SecureCom 2010
[19] M. Jakobsson: K.-A. Johansson (2010); Retroactive detection of malware with applications to mobile platforms,” HotSec’10, pp. 1–13.
[20] W. Enck: P. Gilbert: Sheth. A. N(2010); Taintdroid: An information-flow tracking system for real-time privacy monitoring on smartphones, 9th USENIX Symposium on Operating Systems Design and Implementation
[21] M. Dietz: S. Shekar: Wallach(2011); Quire: lightweight provenance for smartphone operating systems, USENIX Security Symposium
[22] Sven Bugiel: Lucas Davi: Alexandra Dmitrienko: Thomas Fischer: Ahmed-Reza Sadeghi (2011); XManDroid: A New Android Evolution to Mitigate Privilege Escalation Attacks, Technische University.
[23] M. Ongtang: K. Butler: P. McDaniel (2010); Porscha: Policy oriented secure content handling in Android. In ACSAC'10:
[24] R. Schlegel: K. Zhang: X. Zhou: M. Intwala: A. Kapadia: X. Wang (2011); Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones, NDSS, pages 17-33.
[25] M. Conti: Nguyen: B. Crispo (2010); CRePE: Context-related policy enforcement for Android, ISC 2010
[26] A. P. Felt: H. Wang: A. Moshchuk: S. Hanna: E. Chin (2011); Permission re-delegation: Attacks and defenses. USENIX Security Symposium
[27] E. Chin: A. P. Felt: K. Greenwood: D.Wagner (2011); Analyzing inter-application communication in Android. MobiSys.

Akshay Bhardwaj and A J Singh, “Privilege escalation attack on android and its defences,” *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 63-66, 2016.

ISSN: 2454-1532

[28] A. P. Felt: E. Chin: S. Hanna: D. Song: D. Wagner (2011); Android permissions demystified. TR, B.

Akshay Bhardwaj and A J Singh, “Privilege escalation attack on android and its defences,” *International Journal of Scientific and Technical Advancements*, Volume 2, Issue 4, pp. 63-66, 2016.