Privilege Escalation Attacks in Android: Their Approaches, Detection and Defense Techniques

Shruti Dengre  
Information Technology  
Indira Gandhi Delhi Technical University for Women  
Kashmere Gate, Delhi, India  
+91-9650596435  
dengre.shruti@gmail.com

Rishabh Kaushal  
Asst Professor, Information Technology  
Indira Gandhi Delhi Technical University for Women  
Kashmere Gate, Delhi, India  
+91-9810774114  
rishabh.kaushal@gmail.com

ABSTRACT
Android has become one of the most popular mobile phone operating system which is progressively deployed by many mobile devices manufactures for a range of platforms. On the downside, Android is also fast becoming a platform for security attacks owing to its increasing user base and its philosophy of being open source. Android applications run over permission based framework in which each application’s access is controlled according to the permission that it acquires during installation. However, recent research shows that android permission framework is inherently by design are weak and privileged permissions can be obtained by malicious application by launching privilege escalation attacks. Through these attacks, an application may gain permission to perform a privileged task which it is not authorized. In this document we have surveyed the most prominent approaches by which privilege escalation attacks can be conducted. We also discuss ways to detect and put in place defense mechanisms to prevent occurrence of these attacks.

General Term: Security

Keywords: Android security, Privilege Escalation, Pileup Flaws

1. MOTIVATION
As android is the most dominant mobile operating system, the reason behind it, is open source philosophy and not bounded to one manufacturer. Due to its constantly emerging fame and common usage, Android is also an objective of many attacks. Recently the android permissions framework has been targeted as exploit opportunity, as it fail in responding transitive permission usage which allows bypassing the application with less permission to use the resources which need the higher permission. As a result an application which has less permission called non-privileged application is not restricted to access components of an application which more privileged called privileged application [1]. This survey paper focuses on Android security. How different approaches can lead to privilege escalation attacks, what are the detection and defense techniques available by which these attacks should be avoided.

2. METHODOLOGY
Many researchers have given different approaches to execute privilege escalation attack. This section describes the approaches used, how all vulnerable application can be detected and what all mitigation steps have been put forward.

2.1 Approaches
The following approaches have been studied, component –based privilege escalation attack, application-level privilege escalation attack, exploiting pileup flaws.

2.1.1 Component-based permission escalation
One of the most dominant features of android is sandboxing. All application run on their own sandboxes and utilizes the resources provided by them. Sandboxes separate each application with one another and provide all the resources, control which they need to execute. Android’s security architecture does not deal with transitive permission usage

Figure 1 illustrates the site where privilege escalation attack becomes possible. Three android application A,B,C are isolated and running in their own sandboxes. A consist of component CA1 and CA2 without any granted permissions. Similarly B consists of two components CB1 and CB2 with granted permission P1, but neither CB1 nor CB2 are protected with any permissions this means B’s component can access the component of application protect with P1. C has two components CC1 and CC2 which do not have any granted permissions but both are protected with permission P1 and permission P2 respectively. As in figure 1 application A’s component will not be able to access the components of application C as they are protected by permission P1. But components CA1 and CA2 can access component of CC1 and CC2 indirectly via CB1 or CB2,CB1 can be accessed by component of application A as CB1 & CB2 are not protected by any permissions. In turn CB1 can access the components CC1 as application B posse’s permission P1.
2.1.2 Application-level privilege escalation attack

Application level privilege escalation focus on two types of attacks [2][3]: (i) confused deputy, where a malicious application uses the weak edge of another privileged application. (ii) Collusion attack, where applications are combined together with their permissions so that they can perform actions which is not possible from individual’s permissions. This model focuses on security framework which prevents and detects the confused and collude attacks, policies which define permission usage. Figure 2 shows the architecture of security framework based on application level privilege escalation attacks.

![Figure 2: Framework Architecture](image)

According to Bugiel et al., it performs runtime monitoring of the application and explore the communication link of the application in order to ensure that all application is using the system centric policies [2]. They have also conducted heuristic analysis of Android’s system behavior while using some popular application to identify attack patterns, categorize unusual opponent models, and what are the challenges they face while making the architecture which detect the privilege escalation attack [3].

2.1.3 Exploiting the Pileup flaws

This approach focuses on android updating mechanism and mainly put spotlight on its Package Management Service (PMS). Pileup flaw (lead to privilege escalation attacks), is identified, through which malicious application can tactically announce various set of privileges and characteristic on a low-version operating system (OS) and stopped till it is raised to escalate its privileges on the higher version systems[4]. Some Pileup vulnerabilities discovered from Android PMS such as Permission Harvesting and Preempting, Shared UID Grabbing, Data Contamination to name some. All these flaws are exploited at the upgrading time when lower version android devices are upgrading to higher version android device.

2.2 Detection/Defense Approach

All approaches mentioned above have some finding and protection techniques, so that by at prior stage this malicious application can be removed without underlining its utility.

2.2.1 Component based privilege escalation

To avoid or stop this attack the application B must impose extra verification on permission to make sure that application using component of application B must contain granted permission P1.

Normally, it can be done by using reference monitor which identify the weakness in the code of the component [1].

2.2.2 Application-level privilege escalation attack

This model defines some attack pattern based on the system centric policy. Some policy must enforce as critical permissions in the system. If those critical permissions used together may lead to attack. For example if any application (apart from benign) expand it privileges to use Internet and Location information can be defined as an attack pattern and will lead to privilege escalation attack[2][3].

2.2.3 Exploiting the Pileup flaws

This model gives the detection as well as defense techniques by designing new tool SecUp (figure 3) which detect the PileUp flaws by scanning the application before updating its version. This tool takes a proactive action for malicious application. It consists of vulnerability detector, illustrated in figure 4, an exploit opportunity analyzer, a risk database and a scanner app [4].

![Figure 3: SecUp architecture](image)

![Figure 4: vulnerability Detector](image)

3. OBSERVATIONS

Comparison of these approaches is shown in table 1. Component based privilege escalation attack exploits the memory reserved by an application. Extra permission check must be enforced to guarantee that vulnerable application not exploit the privileged application interface. In application-level privilege escalation attacks the policies must be chosen in a way that it would not stuck the genuine application work. Policies must be enforced for each application working under this architecture. In exploiting Pileup flaws, PMS has grabbed all focus in application code; apart from PMS there are many factors which can lead to privilege escalation attack like installation of non-system applications. Other services must be checked for pileup flaws Device policy manager, service manager etc.

| Table 1: comparisons of approaches |
|-----------------|-----------------|-----------------|-----------------|
| **Approach**    | **work on**     | **Detection**   | **Defense**     |
| Component based | Transitive      | NA              | Prevention      |
|                 | permission usage|                 | techniques is   |
|                 |                 |                 | mentioned       |
| Application     | System centric  | By Attack       | Define system   |
| level           | policies,       | patterns        | policies        |
|                 | communications  |                 |                 |
|                 | links           |                 |                 |


<table>
<thead>
<tr>
<th>Exploiting Pileup flaws</th>
<th>Pileup Flaws</th>
<th>Use SecUp tool</th>
<th>Use SecUp tool</th>
</tr>
</thead>
</table>

4. IMPLICATIONS OF OBSERVATION

The observation done in previous section can be further implement by using component based privilege escalation attack approach where advance attack techniques like return oriented programming can be used to pursue this attack. PileUp flaws are mentioned only for system application, these flaws can be identified for non-system application using different service than PMS.

5. REFERENCES


