Securities in Android using SELinux

S. S. Sambare
Dept of Comp Engg,
Pimpri Chinchwad
College of Engineering,
Pune

Sneha Bhambhani
Dept of Comp Engg,
Pimpri Chinchwad
College of Engineering,
Pune

Ritesh Tejwani
Dept of Comp Engg,
Pimpri Chinchwad
College of Engineering,
Pune

Prerana Rai
Dept of Comp Engg,
Pimpri Chinchwad
College of Engineering,
Pune

ABSTRACT
With the increase in technology, the current use of mobile phones is increasing with a rigorous amount and so we need to assure that the information stored in our cell phones is secure and is not being misused. The apps when installed in Android do not provide high level security to the information present in our cell phones and thus the implementation of SELinux helps in securing the information more effectively. Android being a Linux based system can support SELinux and thus provide users with a robust Mandatory Access Control (MAC) to ensure full system security. It not only provides flexible security but also helps in reduction of performance overhead only by implementing a limited chip area.

Keywords
Android, SELinux, Security, Mobile devices, MAC.

1. INTRODUCTION
Android is the most popular mobile operating systems with 80% of global market share. It was released by Google in 2008.It is an open source operating system, primarily based on the Linux Kernel. Android applications are written in Java and run on virtual machines. The open nature of Android attracts a variety of third-party application market places. These, either provide an alternative for the devices that are not allowed to ship with Google Play Store, or provide applications that cannot be offered on the Google Play Store due to policy violations, or for some other reason. Android is mainly designed for use in smart phones, tablets etc. Recently, we also have AndroidTV for televisions, AndroidAuto in cars, AndroidWear for watches and many more. Malware attacks propagating into smart phones include cellular networks, Bluetooth, the Internet, USB, and other peripherals[2]. Security mechanisms such as anti-malware and anti-spam software, host-based intrusion detection tools, and firewalls are available, but not widely used. Today, Android has the largest base among all the operating systems.

Hence, security in Android is of immense importance and is required to be very strong.

2. SECURITY WITH SELinux
The core of the security at application-level in Android is its permission mechanism. Contradicting a typical Linux-based Personal system, different applications in Android are executed as different users. This preventive measure causes the bar to rise on successful exploitation because one application can't affect others in a normally. However, more processes could run as the same user, and, particularly, several system daemons run as system, radio and root users. The security mechanisms of both Android-specific and Linux inherited are insufficient and too coarse-grained to deal with the security issue. [1]. While installing an application, the application displays a dialog indicating the permissions requested and asks if it should continue the installation. The user can not accept or refuse individual permissions – he must accept or refuse all the requested permissions only as a block. The security model of Android is based on the concept of sandbox. Every application runs in the separate individual sandbox. Beginning with the 4.3 Android version, SELinux further describes the boundary limits of the Android application sandbox[4]. Android uses SELinux as a part of its security model, which enforces MAC(mandatory access control) over other processes, including the processes running with superuser/root privileges. Android security is enhanced by SELinux that confines privileged processes and automating the creation of security policy. Many contributions have been made to it by various organizations. Using SELinux, Android can do better confinement of system services, access control to application and system data and logs, reducing the ill-effects of malicious code, protecting users from flaws in code on mobile and other handheld devices. It is basically operated in 3 modes :

1. Disable.
2. Permissive.
3. Enforcing.

2.1 System Architecture

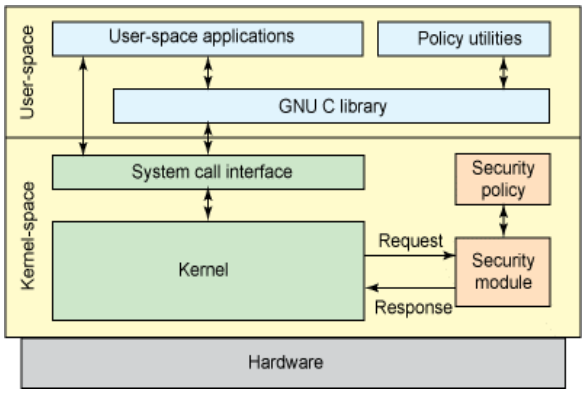

Fig. 1.Integration of Security module [1].

2.2 The SE Android Policy
SE Android policy is one of the cores of the entire SE Android security mechanism. In addition, the security architecture must also have a strict security policy to ensure that the access subject has only minimal access permissions to the object, so that the program can execute the basic functions but will be prevented from executing malicious operations[3].

The SE Android's implementation is in enforcing mode, instead of the non-functional disabled mode or the notification-only permissive mode, to act as a reference and facilitate testing and development. SELinux does not change any existing security in the Linux environment; instead, LSM

extends the security model to include Mandatory Access Control (e.g., both MAC and DAC are enforced in the SELinux environment).

2.3 The Extended Policy

Android has added some new features to the 4.4 version of the Android OS ("Kitkat"). The most important change among the new features is the ability to integrate SE Android in enforcing mode, which means access permissions for all Android components is under the control of SE Android[5].

SELinux embedded Android refers to Security Enhancements for Android, a security solution for Android that identifies and addresses critical gaps. Initially, the project's scope was to enable the use of SELinux in Android to limit the damage that can be done by flawed or malicious apps and to enforce separation guarantees between apps. SELinux LSM in Android is now the overall framework for implementing SELinux mandatory access control (MAC) and middleware mandatory access control on Android.

2.4 SELinux Components

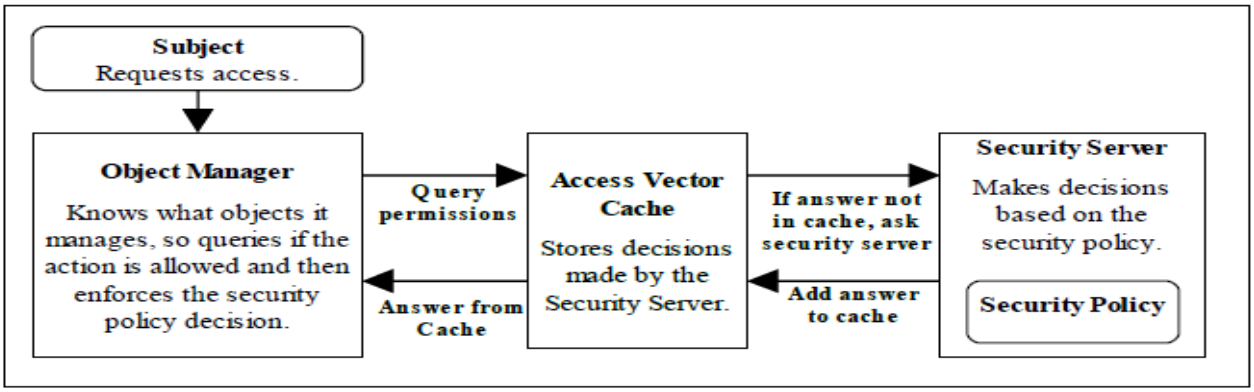

Fig. 2. SELinux Components[3].

A high level diagram of the SELinux core components that manage enforcement of the policy and comprise of the following:

1. A subject that must be present to cause an action to be taken by an object (such as read a file as information only flows when a subject is involved).
2. An Object Manager that knows the actions required of the particular resource (such as a file) and can enforce those actions i.e. allow it to write to a file if permitted by the policy server.
3. A Security Server that makes decisions regarding the subjects rights to perform the requested action on the object, based on the security policy rules.
4. A Security Policy that describes the rules using the SELinux policy language.
5. An Access Vector Cache (AVC) that improves system performance by caching security server decisions[6].

2.5 Hooking with LSM

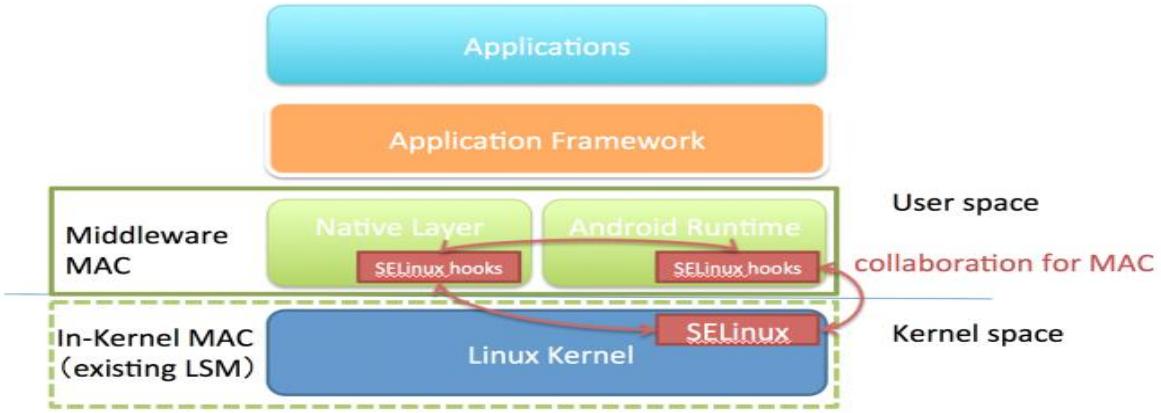

Fig. 3.Hooking with SELinux LSM[4]

The concept hooking covers a wide range of techniques used to modify or augment the behavior of an operating system, of applications, or of other software components by intercepting function calls or messages or events passed between software components. Fuctional Code that handles such intercepted function calls, events or messages is called a "hook".

It is sometimes possible that there is simply not enough space for our hooking method. This happens when there is either an

unconditional jump instruction, or a return instruction[7]. Such situation occurs when we've simply hit the end of a function.It is perfectly possible to have multiple layers of hooks, i.e. Multiple hooks on one function. The only one condition which must apply; when installing a new hook over an existing hook, then you will have to make sure that the required length of the new hook is equal to, or lower than, the length of the existing hook in the code.

2.6 SE Policy

```
# camera interface
type camerad, domain;
permissive camerad;
type camerad_exec, exec_type, file_type;
init_daemon_domain(camerad)
unconfined_domain(camerad)
relabelto_domain(camerad)
allow camerad tombstone_data_file:dir relabelto;
# force automatic transition when init spawns us
# Leaves camerad unconfined, at present
# Allow domain transition to this domain
# Allow debugged to manipulate tombstone files[8].
```

3. IMPLEMENTATION STEPS IN DETAIL

1. Upgrade phone to latest Android version – 4.3+ (Jellybean) .(Contains SELinux module in complete Enforcing mode.)
2. Root Phone (Jailbreak).
3. Get root access.
4. Develop Base policy.
5. Set the system to enforcing mode by using command 'setenforce 1'.
6. Edit the config file and set
   Selinux= enforcing, then reboot the system.
7. Test that the policy meets the security requirements or not.
8. Ensure you are logged in as root and Selinux is in enforcing mode to
perform build process.
9. Produce a policy.conf file with the text editor (vi or gedit).
10. Compile the policy with 'checkpolicy' to produce the binary policy file.
11. Make the following directories to store the file
 - mkdir /etc/selinux/monolithic_test/policy
 - mkdir -p /etc/selinux/monolithic_test/context/files
 12. Reboot system
13. Login as root.
14. setenforce 1 (enforcing).

The new policies would be enforced.

4. CONCLUSION

Security is correctly perceived as a crucial property of mobile operating systems. The integration of SELinux into Android is a significant step toward the realization of more robust and more flexible security services. Our approach is a natural application of this design. The potential of an SELinux based solution like SEIntentFirewall is extensive and leads to a significant improvement in access control enforcement and app isolation. The project would facilitate the usage of any application on the phone without the risk or threat to any confidential or personal data to be accessed by any other application. It therefore protects the personal information of the user from being misused thus ensuring full time security.

5. REFERENCES

[1] Simone Mutti, Enrico Bacis, "An SELinux based intent manager for Android", IEEE CNS 2015.

[2] Prof Dr. Frank Bellosa, Stefan Brahler, "Analysis of Android Architecture", IEEE 2015.

[3] Lukas Aron and Petr Hanacok at BRNO university of technology, "Introduction to Android 5 Security ", CEUR-ws.org, 2014.

[4] Chetan C Kotkar, Pravin Game, "Exploring Security mechanisms to Android Device", International Journal Of Advanced Computer Research, 2013.

[5] Han Bing at North China University of Technology,"Analysis and research of system security based on Android ", 5th International Conference of Intelligent computation technology and automation, 2012

[6] Asaf Shabtai, Yuval Feldel and Yuval Elovici at Ben-Guiron University, "Securing Android Powered Mobile devices using SELinux ", IEEE 2010.

[7] SELinux/Tutorials/Creating your own policy module file https://wiki.gentoo.org/wiki/SELinux/Tutorials/Creating_your_own_policy_module_file

[8] Building a Local Policy Module https://www.centos.org/docs/5/html/Deployment_Guide-en-US/sec-sel-building-policy-module.html