JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Self-protection of Mobile Applications Using Reflection Design

Nadir K.Salih, Saria Mohieldeen

Computer Engineering, College of Engineering, University of Buraimi, Oman

Abstract : Mobile apps often handle sensitive information such as personal details, financial data, and communication. Badly designed applications may contain security vulnerabilities that can be a source of danger to users and their privacy. Reflection is the ability of a program to examine and modify the structure and behavior of an object at runtime. We use reflection design in java programming language with android to solve the mobile application problems. We started to apply the concept of the autonomic system that let the system work without intervention of the user. It has given by implemented and designed method that using the reflection concept. In the reflection we design the methods to do the prevention process and determine the data which we need to save it by putting constraint in all steps to sure from it.

Keywords :Autonomic computing, Self-protection, Reflection

I. Introduction

Compared to traditional mobile phones, which mainly provide mobile telephony functions, Smart phones are general-purpose handheld com-putting and communications devices that support multimedia communications and applications for entertainment and work [1]. One of the most attractive features of smart phones is the availability of many apps for users to download and install. Due to this quantum jump leap in functionality, threat of upgrading traditional mobile phones to smart phones is tremendous [2]. However, it also means hackers can easily distribute malware to smart phones, launching various attacks. This issue should be addressed by both preventive approaches and effective detection techniques. [3]. Android Market has fewer retractions when it comes to registering as a developer. This is Android's strategy to encourage future app developers, but this also makes it is easier for cybercriminals to register as developers to upload their malicious apps or their Trojanized counterparts. And so, Android is a modern and popular software platform for smart- phones [3]. Among its predominant features is an advanced security model which is based on application-oriented mandatory access control and sandboxing [2]. This allows developers and users to restrict the execution of an application to the privileges it has (mandatorily) assigned at installation time. The exploitation of vulnerabilities in program code is hence believed to be conned within the privilege boundaries of an application's sandbox [4]. So the presence of bugs and security holes is statistically numerous and variety so we needs to combine protection mechanisms inside the mobile system to become applications self-protection. Android is an open, security focused, mobile platform that is programmed with Java [5]. Mobile application's is very important for the production companies, so they take many Techniques to protect it. The one of ways to protect the mobile applications is self-protection using reflection. The term Self-protection is one of autonomic system that aims at enabling computing infrastructures to perform administration tasks without human intervention that means ability of an autonomic system to secure itself against attacks to detect illegal activities and to trigger counter measures in order to stop them [6]. As a result of using reflection in the E-Vote application we contributed on

1. Increase self-security in mobile applications.
2. As a result of using reflection in the application has achieved results to provide data security
3. Combine protection mechanisms within the mobile system which facilitate self-protection for applications.
4. Protect sever from overload which happen form attacker more over request (Reply message) replay after send.
5. Create method use to prevent grant robot permission.

This paper consists of five sections: the first section describes the introduction of the research. The second section describes the related work and motivation of the article. The third section describes the methodology for the project. The fourth section describes the self-protection for the E-vote application. Finally in section five conclusion and recommendation our research.

II. Related work

Adrienne Porter Felt et al. [7] they built Stowaway, a tool that detects over privilege in compiled Android applications they used automated testing tools on the Android API to build the permission map that is necessary for detecting over privilege. Long Lu et al. [8] they study a general category of vulnerabilities found in Android apps, namely the component hijacking vulnerabilities. They used CHEX that is a static analysis method to automatically vet Android apps and detect possible hijack-enabling flows by

conducting low-overhead reach ability tests on customized system dependence graphs. To tackle analysis challenges imposed by Android's special programming paradigm. D. Bucerzan, C. et al [9]. The Android SmartSteg app can quickly hide and encrypt files using digital images of MB dimensions as a cover. LSB data masking information is integrated with random function and symmetric key encryption to transfer digital information, securely between Android smart phones. Lucas Davy et al. [10] it turned out that an escalation attack is possible. They show that a real application exploited at run time, or a malicious application can escalate the granted permissions. Our results immediately indicate that the Android security model cannot handle the pass-through attack and that the Android protection box model fails as a last resort against malware and advanced uptime. Golam, et al [11] they used the popular TaintDroid system is used to track sensitive information on Android mobile phones. The data is tracked during its publication through variables, exchanged messages and files, by marking them with contaminated tags, such as device identifiers or user contact details, then warnings are issued when this information is misused, we provide a set of attacks to track robot-based damage. The technologies presented are applied in the Android app, ScrubDroid. They have successfully tested our app with TaintDroid apps for Android versions 2.3 to 4.1.1, although dye tracking may be a valuable tool for developers, but it will not effectively protect sensitive data from the black box of the zealous attacker that applies any of the anti-tracking methods the pollutants provided. Olivier Mehani et al. in [12] they using taint analysis for tracking the privacy information on Android-based mobile conclude that, although taint tracking may be a valuable tool for software developers, it will not effectively protect sensitive data from the black-box code of a Motivated attacker applying any of the presented anti-taint tracking methods. Hao Peng et al. [13] they used the idea of risk classification and improved communication process for android applications. Thomas Bl¨asing et al. [14] they suggest an Android Sandbox (AASandbox) app that can perform static and dynamic analysis on Android programs to automatically detect suspicious apps. Static analysis scans the program for harmful patterns without installing it. Dynamic analysis performs the application in a completely isolated environment, that is, the protection box, which intervenes and records a low-level Interactions with the system for further analysis. Both the security box and detection algorithms can be deployed in the cloud, providing quick and distributed detection of suspicious software in a mobile app store like Google's Android Market. In addition, AASandbox may be used to improve the efficiency of classic antivirus applications available for Android. Fumihiro Kanei et al. [15] they proposed a method of strong self-protection against evasion attacks. The suggested method automatically creates the ability to rediscover discovery in applications. The detection code randomly divides into several blocks, which are inserted directly into the zip of applications. The results of the evaluation indicate that the degree of durability, which is calculated based on false positives from the attackers' point of view, is 3.5 times higher than in the current method. The suggested method can also easily protect apps because they only require their secondary code. j Ametller et al.[16] they present a new solution for the implementation of flexible protection mechanisms in the context of mobile agent systems, where security problems are currently a major issue. In their scheme, agents protect their code and data by carrying their own protection mechanisms. This approach improves traditional solutions, where protection was managed by the platform. The implementation is far from trivial. They have implemented this scheme in the JADE framework, using Java. Any application using mobile agents can incorporate these mechanisms to implement agent protection with a minimum impact in the existing code base. The conclusion Mobile agent protection can be achieved using cryptographic methods. The requisite verification and decryption tasks can oversee either the agent's code or the host platform.

III. Methodology

Autonomic Computing presents a new paradigm where computing systems manage themselves, guided by high-level objectives to enabling computing infrastructures to perform administration tasks without human intervention [17]. The term self-protection is one of the autonomic systems that aim to enable computer infrastructures to perform administrative tasks without human intervention. This means that an independent system can secure itself against attacks to detect illegal activities and launch counter measures to stop them [18]. Mobile Application Security Test is a process to review application properties and vulnerability code. It is a combination of static analysis, code review and penetration testing in the reflection model, one of the components can think about and modify itself, in terms of different situations, so the concept of reflection will be used to achieve self-protection.

a. Reflection Concept

Compositional adaptation enables software to modify its structure and behavior dynamically in response to changes execution environment [19].

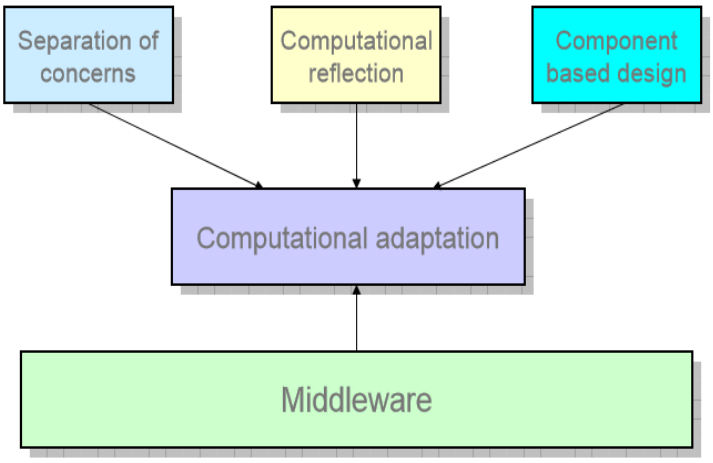


Figure 1 Main Technologies Supporting Composition adaptation

The reflection mechanisms of a programming language provide a running program with the ability to examine itself and its environment to perform self-examination, a program needs an accessible representation of itself; this level of indirection is facilitated through metadata and is fundamental to a reflective system [20]. The two main aspects of self- manipulation are introspection and intercession, which are the abilities of a program to observe and modify (respectively) its own state and behavior. Both aspects require a mechanism for encoding execution state as data. In Java this is realized with so-called Meta objects, which

provide access to the representation of Java classes and are available in the java.lang.reflect package [19]. The java reflection allows the program system to examine or modify the behavior of classes, interfaces, methods, and fields at run time, allowing the program to adapt to dynamically changing runtime environments [21]. Computational reflection refers to a program's ability to reason about, and possibly alter its own Behavior. Reflection enables a system to reveal selected details of its implementation without compromising portability

b. Levels of Reflection

a. Introspection. : No alteration, read only to let an application observe its own behavior
b. Behavioral Reflection. Limited Alteration possible, such as method invocation. In general: behavior of operations is changeable.
c. Structural Reflection. The ability to change data structures used by the runtime system, to let a system or application acts on these observations and modify its own behavior. Such as Class for example.

c. Reflection Types

a. Structural reflection Addresses issues related to class hierarchy, object interconnection, and data types. As an example, a met level object can examine a base-level object to determine what methods are available for invocation.

b. Behavioral reflection Focuses on the application's computational semantics. For instance, a distributed application can use behavioral reflection to select and load a communication protocol well suited to current network conditions.

d. Reflection Process:

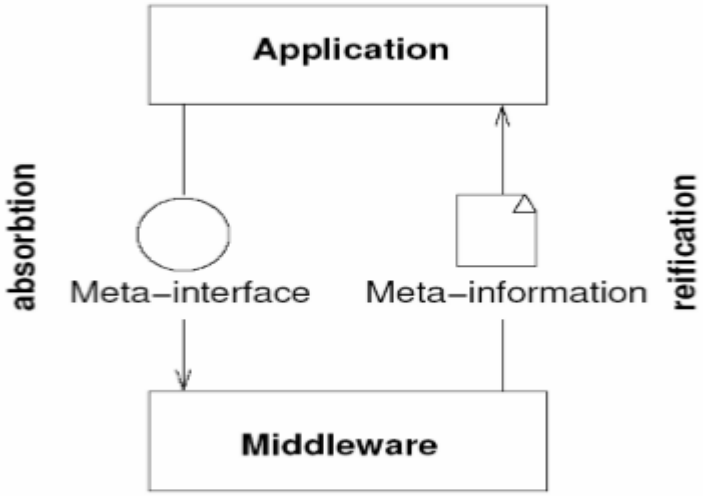


Figure 2 Reflection process

As shown in figure 2 Absorption is the process where some aspects of the system are altered or overridden. **Reification** The key to the approaches to make some aspects of the internal representation of the middleware explicit, and hence accessible from the application, through a process. The behavior of the middleware with respect to a specific application is described as a set of associations between:

1. the services that the middleware customizes,
2. the policies that can be applied to deliver the services, and
3. the context configurations that must hold for a policy to be applied

Reflection design: Our model assumes that the behaviors of the middleware with respect to a particular service is determined at any time by one and only one policy as depicted in figure 3

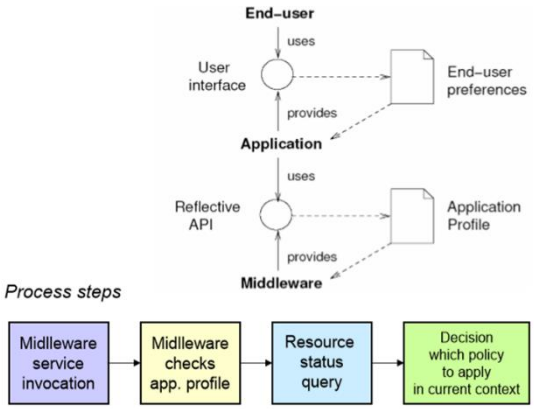


Figure 3 Reflection design

Enhance object-oriented design and programming by techniques for runtime system adaptation. There are two key notions:

i. *Design elements*: We use the term design element to denote representations of design decisions in programs. In fact, we require that design elements are amenable to reflection such that design decisions can be observed and modified at runtime
ii. *Design operators*: When compared to basic techniques such as the use of a Meta object protocol, the use of design elements makes runtime system adaptations more disciplined and more manageable. To this end, we provide abstractions that capture common

design elements in a reusable manner. Applications of such abstractions perform system adaptations at a design level; hence, we call them design operators [19].

IV. Self-Protection E-Vote Application and Results

The implementation stage has been done in developing the online E-voting application and discussing the results. The development involves the Graphical User Interface (GUI) design, data base, and the coding development for entire application. The method used in developing the application and database also is discussed here. E-voting application has been developed using Android Studio. The source code of the application using the Java programming language and the database was created using MySQL (using PHP language to create server pages). Below figure 4 shows the implemented framework of the E-voting application using reflection:

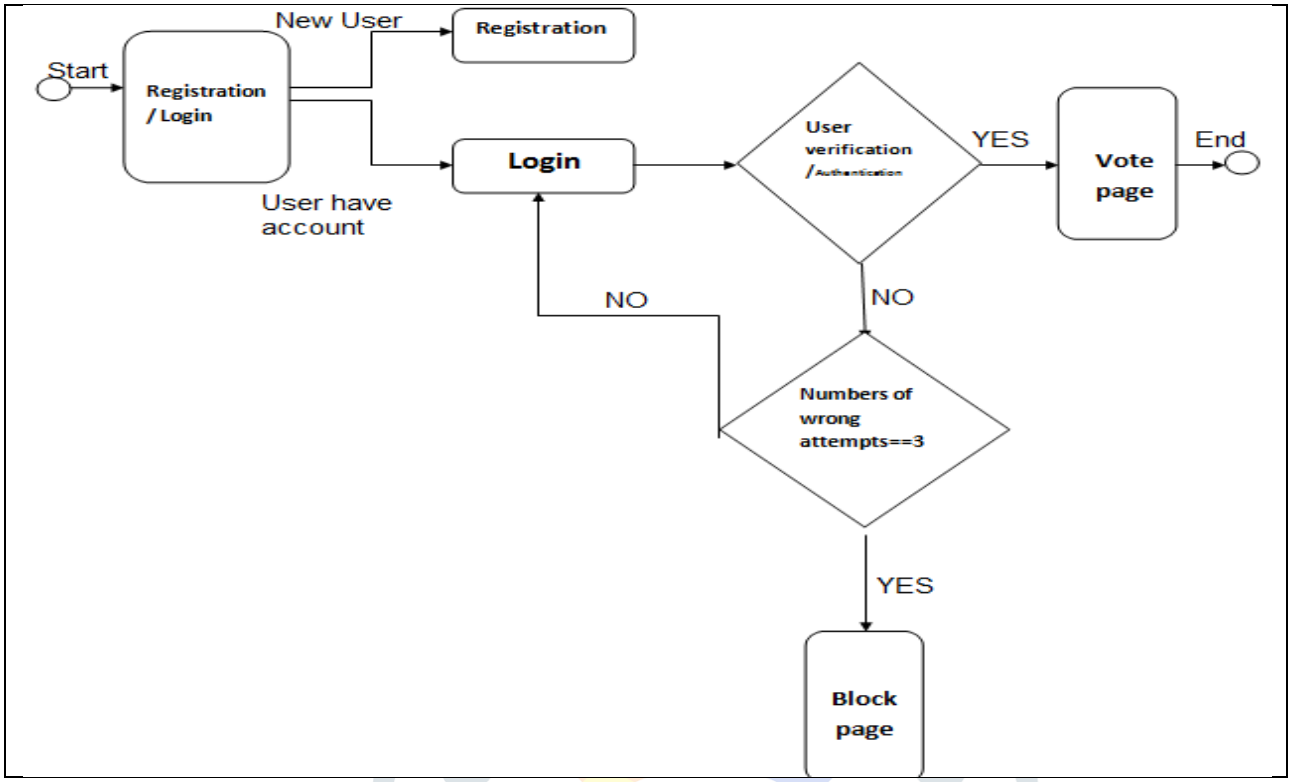


Figure 4 E-Vote Application framework using Reflection

For this application, it was developed in Android Studio using the Java programming language. Window 8 is used as a operating system with an Intel (R) processor and 4 GB of RAM to develop the application environment. This application relies on the civil registry databases in the registration process where the national number from the user is validated. The table (4.1) as the below shows the environmental needs for the application development.

Table 1 Environmental needs for the application development

Type	Tool	Platform
Programming Platform	Android Studio	Windows
Programming Language	Java	Windows
Operating System	-	Windows 10
Hardware	-	Asus Laptop
Processor	Core 3	Intel (Side) Core i3,4030U with 1.9GHz
RAM	-	4GB
Data base	XAMP	MySQL

a. Interface design

The interface allows the user to interact with the application. The application of electronic voting here consists of four interfaces, the first interface is registration, the second interface is the login, the third is the vote, and the last interface appears in the case of the user who was blocked, the figure 5 in the below show the E-voting model:

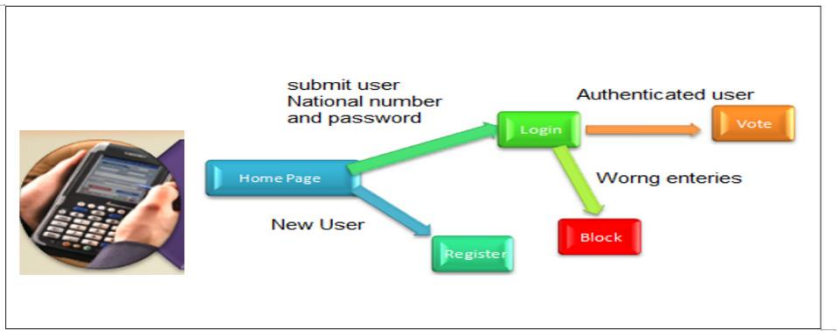


Figure 5 E-Vote interfaces model

It is the first screen in the application to appear to the user when it is opened, and the new users are registered to the application. The screen contains three fields: the national number, which consists of 11 digits. The entrance is a number and must be found in the civil registry database, the verification code is a number and consists of 4 digits where it is sent to the user and the password numbers are 8 digits. The interactive systems between the application files and server files are depicted in figure 6.

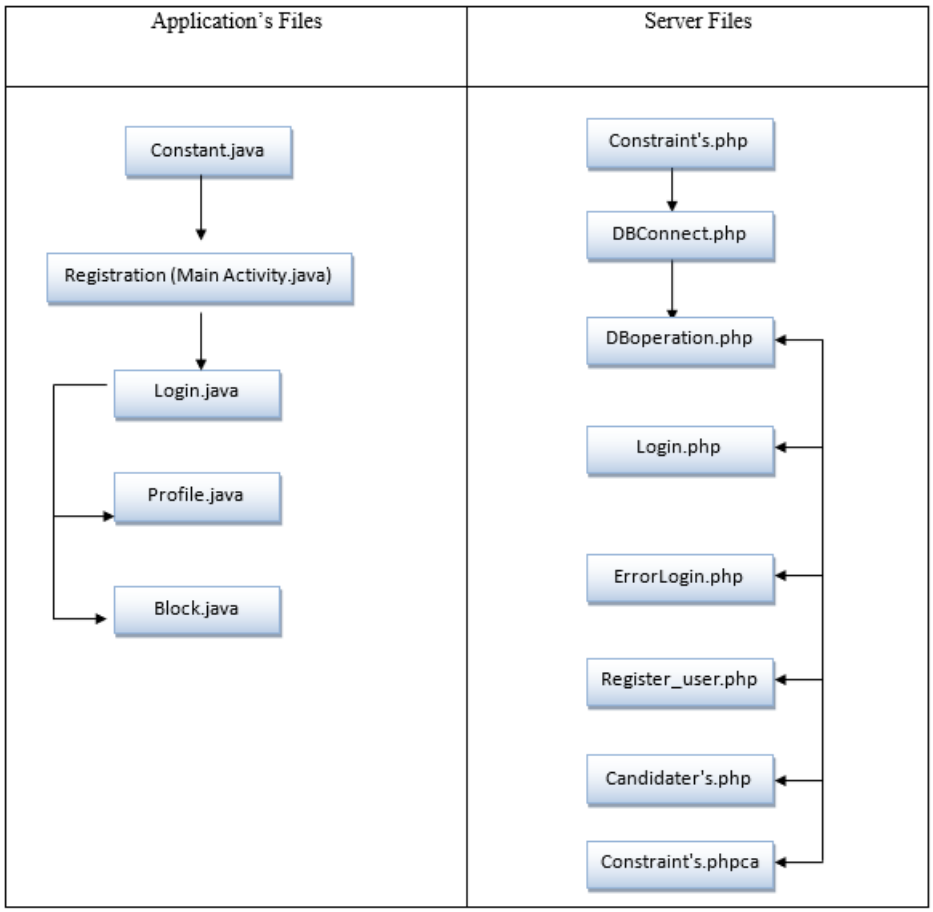


Figure 6 Interactive systems files

IsBlock Function: This function returns a Boolean value (true / false). It's required in a Login method (later on). This function is invoked with the beginning of the program for the first time. Which means it will be called onCreate method in mainActivity. It tests the user's status if it is prohibited from login. The following table (4.3) shows the function code:

Table 2 isBlock Function

```
public boolean isBlock()
{
    SharedPreferences sharedPreferences  = mCtx.getSharedPreferences(SHARED_PREF_BLOCK,Context.MODE_PRIVATE);
    if (sharedPreferences.getString(SHARED_PREF_BLOCK,null) != null){
        return true;
    }else {
        return false;
    }
}
```

UserBlock Function :function that prevents the IP address in case of wrong login more than 3 times using the (stop) function in the login file and keeping the Internet address in (context) to prevent it from entering again, the following table (4.4) shows the function code:

Table 3 UserBlock Function

```
public boolean userBlock()
{
    SharedPreferences sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_BLOCK,Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString(SHARED_PREF_BLOCK,Constants.IP);
    editor.apply();
    return true;
}
```

IsLogin Function: Is a function that tests the status of the user currently logged in or not, returns the value (true / false) and checks the sharing preferences files with private mode, the following table (4.5) displays the function code:

Table 4 isLogin Function

```
public boolean isLogin()
{
    SharedPreferences sharedPreferences  = mCtx.getSharedPreferences(SHARED_PREF_NAME,Context.MODE_PRIVATE);
    if (sharedPreferences.getString(SHARED_PREF_NID,null) != null){
        return true;
    }else {
        return false;
    }
}
```

UserLogin Function:After the actual registration process, (enter the national number and password to activate login) we use this function to save user information in a shared preferences. This information we will use to know the user have a login or not. The following table (5) displays the function code:

Table 5 UserLogin Function

```
public boolean userLogin(String n_id)
{
    SharedPreferences sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_NAME,Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString(SHARED_PREF_NID,n_id);
    editor.apply();
    return true;
}
```

GetMacAddress Function:This function returns the address of the device that was logged in, and saved in the shared preference file to validate if is this MAC Address is blocked or not in Block Mac Function. The following table (6) that display the function code:

Table 6 GetMacAddress Function

```
public String getMacAddress()//function to return MAC Address for the device
{
    WifiManager wimanager = (WifiManager) mCtx.getSystemService(Context.WIFI_SERVICE);
    String macAddress = wimanager.getConnectionInfo().getMacAddress();
    if (macAddress == null) {
        //macAddress = "Device don't have mac address or wi-fi is disabled";
        SharedPreferences sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_NAME,Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString(SHARED_PREF_BLOCK_MAC, macAddress);
        editor.apply();
    }
    return macAddress;
}
```

BlockMacAddress Function:The function prevents MAC address in case of wrong login more than 3 times using the stop function in the log file and keeping the MAC address in (context) to prevent it from entering again, the following table (7) shows the function code:

Table 7 Block Mac Address Function

```
public boolean blockMac()
{
    SharedPreferences sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_BLOCK_MAC,Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString(SHARED_PREF_BLOCK_MAC, getMacAddress());
    editor.apply();
    return true;
}
```

Unblock Function:Clear all data related to the IP Address and MAC Address from sheared preferences file, the following table (8) shows the function code:

Table 8 Unblock Function

```
public boolean unBlock()
{
    SharedPreferences sharedPreferences  = mCtx.getSharedPreferences(SHARED_PREF_BLOCK,Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.clear();
    editor.apply();
    return true;
}
```

LOGIN CLASS FUNCTIONS:LOGIN FUNCTION:USING THE JSON OBJECT CLASS ALREADY IN A GROUP TO SEND THE LOGON DATA TO THE SERVER AND BYPASSING THE PROTECTED HASH MAP TO ALL EXCEPTIONS WHEN WE SEND PARAMS BY HTTP REQUEST (POST), ON THE SERVER WE ACTUALLY EXPECT THAT PARAMS BY HTTP REQUEST IS SPECIFIED. PHP SENDS A RESPONSE IN THE ARRAY (JSON_ENCODE).THIS ARRAY HAVE TWO KEYS ONE IS FOR ERRORS (TRUE/FALSE) AND THE OTHER FOR DATA (MESSAGE) THAT WE NEED TO USE IT IN OUR APP.

b. Block Class function: In this class we check if the user is block or not by using isBlock function from share preferences class. The flowing figure 7 shows the flowchart for Block process:

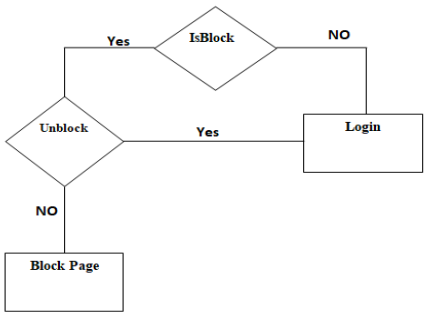


Figure 7 Block Process

c. **Discussion:** Reflection design is optimal with a mobile application to implement security at the application level, especially in electronic voting, which requires a high degree of integrity and transparency during the voting process due to the sensitivity of the data used to ensure the correct results are obtained. This application uses the concept of reflection started from the exploitation of operations on the application so that the procedures take place in the application at the end user and the procedures are validated and approved or not in the server where the data is synchronized with the civil registry data to ensure equal voting rights. In this way, we have separated the operations into two parts, the first part is at the application level and does not contain any sensitive information other than keeping the national number of the vote, which is the result of a valid log-in process synchronized with the civil registry data, which is after the registration process, which means that it must be registered in the voting program Email first by application, this ensures that the server is not overtaxed by unauthorized data, and thus we ensure that there is no re-transmission of data again to the server (replay after send) and that there is no concern about the SQL injection. As for the issue of sending and receiving data from the server to the server, the Super Global Post guarantees the confidentiality of data in the web or HTTPs request, an application that is usually in a secure format, especially after addition character "S", which means that it is secure from beside re-send data request. Finally, reflection helps us protect against data capture as it passes to the server due to the inability to change data except by log out of the application and login again, which requires another correct data again.

V. Conclusion and Recommendations

The purpose of using reflection is to make the application protect its data without interference from any external factor. We conclude that we protect our application's data from detection and theft by the attackers with using reflection design because of using reflection in the application has achieved flaw the results to provide data security. From the results of this study, it's recommended that: Farther study and research must be done in self-protect error injection in mobile application using reflection. Using reflection to make certain self-protect to decrease bugs.

References

1. Asim A Al-Abri, Nadir K Salih. Mobile Application Malware Self-Diagnosis. IEEE International Conference on Electrical Engineering, Computer Science and Informatics (EECSI). Pp 338-342, 2023.
2. Nadir K.Salih1,Hanan Ahmed. Survey on QoS for Applications. International Journal of Research Publication and Reviews. Vol 3, no 6, pp 2157-2164, June 2022.
3. Amal S. Al Maamari , Nadir K.Salih. Comparison between Black hole and Flooding Attack in Mobile Ad hoc Network and their Simulation Study. International Journal of Research Publication and Reviews. Vol 3, no 11, pp 3048-3053, November 2022.
4. Asim A. Al Abri, Nadir K. Salih. Study of Self-Protection in Mobile Applications. International Journal of Research Publication and Reviews. Vol 3, no 11, pp 3048-3053, November 2022.
5. Amal S. Al Maamari ,Nadir K. Salih. Study of attacks on wireless Network. International Journal of Computer Engineering and Technology. Volume 13, Issue 2, May – August 2022, pp. 21-26.
6. Nadir Kamal Salih, Self-Diagnosis of Cancer Using Case Base Reasoning Algorithm. International Journal of Research in Engineering, Science and Management. Volume4, Issue6,P: 24-28,2021.
7. FELT, Adrienne Porter, et al. Android permissions demystified. In: Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011. p. 627-638.

8. Lu, L., Li, Z., Wu, Z., Lee, W. and Jiang, G., 2012, October. Chex: statically vetting android apps for component hijacking vulnerabilities. In Proceedings of the 2012 ACM conference on Computer and communications security (pp. 229-240). ACM.
9. BUCERZAN, Dominic; RATIU, C.; MANOLESCU, Misu-Jan. SmartSteg: a new android based steganography application. International Journal of Computers Communications & Control, 2013, 8.5: 681-688.
10. Davi, L., Dmitrienko, A., Sadeghi, A.R. and Winandy, M., 2010, October. Privilege escalation attacks on android. In international conference on Information security (pp. 346-360). Springer, Berlin, Heidelberg.
11. BABIL, Golam Sarwar, et al. On the effectiveness of dynamic taint analysis for protecting against private information leaks on Android-based devices. In: 2013 International Conference on Security and Cryptography (SECRYPT). IEEE, 2013. p. 1-8.
12. SARWAR, Golam, et al. On the Effectiveness of Dynamic Taint Analysis for Protecting against Private Information Leaks on Android-based Devices. In: SECRYPT. 2013.
13. Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Potharaju, R., Nita-Rotaru, C. and Molloy, I., 2012, October. Using probabilistic generative models for ranking risks of android apps. In Proceedings of the 2012
14. BLÄSING, Thomas, et al. An android application sandbox system for suspicious software detection. In: 2010 5th International Conference on Malicious and Unwanted Software. IEEE, 2010. p. 55-62.
15. KANEI, Fumihiro, et al. Poster: Protecting Android Apps from Repackaging by Self-Protection Code. 2017.
16. Ametller, J., Robles, S. and Ortega-Ruiz, J.A., 2004, July. Self-protected mobile agents. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1 (pp. 362-367). IEEE Computer Society
17. DAWSON, Dylan. Facilitating autonomic computing using reflection. 2009. PhD Thesis.
18. MALEK, Sam, et al. Security and Software Engineering. In: Handbook of Software Engineering. Springer, Cham, 2019. p. 445-489.
19. MCKINLEY, Philip K., et al. Composing adaptive software. Computer, 2004, 7: 56-64.
20. PONTES, Felipe, et al. Java reflection API: revealing the dark side of the mirror. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, 2019. p. 636-646.
21. LI, Yue; TAN, Tian; XUE, Jingling. Understanding and Analyzing Java Reflection. ACM Transactions on Software Engineering and Methodology (TOSEM), 2019, 28.2: 7.