INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 11, Issue 1 - V11I1-1518)
Available online at: https://www.ijariit.com

Implementation of reflective programming in .NET

Hakik Paci
hpaci@fti.edu.al
Polytechnic University of Tirana, Tirana, Albania

Dorian Minarolli
dminarolli@fti.edu.al
Polytechnic University of Tirana, Tirana, Albania

Nelda Kote
nkote@fti.edu.al
Polytechnic University of Tirana, Tirana, Albania

ABSTRACT

Programming languages are evolving very fast, but their principles are almost the same. There are two main concepts about the writing process of software, Closed Code and Open-Source Software. Most developers prefer to have the source code of the software developed by other developers so they can understand, read, and change the product very easily. Even when a product is an open source, the number of developers who modify the product is very low compared to the number of users who use the product.

In this paper we will present an implementation of reflective programming in .NET to allow other developers to extend the capabilities of a product with their contributions on writing source code even if the project is not open source.

Keywords: *Reflective programming, plugin, open project, .NET technology*

1. INTRODUCTION

Open-source software can offer an open gate to the developers to use the software written by software developers and to extend with their needs. They can contribute to the development of new features, bug fixing, optimizing the code, etc. of a product. So, from the developer's point of view, the open-source projects are very good. Also, open-source projects have their weak point when you compare them with closed-source projects, and some of most weak points the security, the copyright of the source code, and the difficulty of modifying the old versions to a new one when the developers of open-source projects can change the programming techniques.

In this paper we will talk about the possibility of extending a software product which is not open source but is developed using reflective programming to allow the integration of plugins developed by third parties.

Generally, these types of products are developed when the number of types of hardware components to realize something is too much or the number of algorithms to analyze, optimize, etc. are not limited to the algorithms integrated into the product.

There are different ways to extend the capabilities of a product, for example, using another product that receives requests from the main product a return the result and the communication between the products is via a known way like Web Services, file share, network connection, etc. Another way to allow the developer to use some functionalities developed by third parties is to integrate their libraries in the development process, but this is not the case for extending the existing products.

In this paper, we will present an implementation of reflective programming in .NET to allow other developers to extend the capabilities of a product with their contributions to writing source code even if the project is not open source.

2. REFLECTIVE PROGRAMMING IN .NET

Reflection helps programmers make generic software libraries to display data, process different formats of data, perform serialization or deserialization of data for communication, or do bundling and unbundling of data for containers or bursts of communication. [1] Reflection can be used for observing and modifying program execution at runtime.

A reflection-oriented program component can monitor the execution of an enclosure of code and can modify itself according to a desired goal of that enclosure. This is typically accomplished by dynamically assigning program code at runtime. [1]

The reflection technique can be implemented in different programming languages like Java, .Net (C#, VB), Delphi, Python, etc.

To develop a program that can change its own structure on runtime the developer must take into consideration adding the capability on the product to inspect classes, interfaces, and methods on runtime. This means the product must do the instantiation of new objects and call of methods without knowing their names during the compilation of the source code.

Reflection is an important way to extend the functionality of a product but also it has a lot of potential. That means that, by using it not correctly can allow attackers to modify the product to steal, modify the data. Historical vulnerabilities in Java caused by unsafe reflection allowed code retrieved from potentially untrusted remote machines to break out of the Java sandbox security mechanism. [1] A large-scale study of 120 Java vulnerabilities in 2013 concluded that unsafe reflection is the most common vulnerability in Java, though not the most exploited.[1]

Microsoft in .NET programming languages offers the possibility to use reflection techniques. The following code is an example of how to use reflective programming in C#.

```
// Without reflection
var c1 = new ClassHello();
c1.PrintHello();

// With reflection
Object    c2    =    Activator.CreateInstance
("plugin.name.ClassHello");
MethodInfo     method     =     c2.GetType()
.GetMethod("PrintHello");
method.Invoke(v2, null);
```

Developing a software product which will use reflective technique we need to take the following steps:

- Exposing the functionality available to the plugins.
- Defining the structure of the interface which will be implemented by every plugin.
- Adding the possibility to load the plugins in runtime when it is necessary.
- Performing reflection to find all these classes of a plugin and instantiate/use them when their functionality is required.

The plugin must be public and well documented so the third parties can develop some functionality based on this plugin so it can be integrated on runtime using reflection.

3. IMPLEMENTATION OF REFLECTION ON REAL .NET APPLICATION

To implement the reflection on a .NET application we decided to develop an application that personalizes data on ID-1 cards with different laser engraving machines. The laser engraving machine does not have a standardization for how to communicate with third-party software like printers which use drivers provided by the vendor of printer. The reason is that these machines are not so common on a daily usage and the parametrization depends on the card structure and the security features of the card. Also, the system cannot be developed just for one type of machine because it must be used for a long period of time, and it must be integrated even with laser engraving machines which we do not know how they will communicate in feature.

To solve this problem, we prepared an interface in .NET where we defined the structure classes, fields for data, and the methods which are necessary to do the personalization and to get the result of the personalization process from the machine. The structure of data changes on every card so we had to use XML to have the possibility to have a dynamic data structure for each card.

Based on the interface the developers build plugins to communicate with laser engraving machines using the protocols of each type of machine. The machines used in this project use three communication protocols as follows:

- The Muehlbauer clp50 machine used in this project uses the file sharing protocol. This machine is very old and gets the personalization jobs using a shared folder where the machine reads files from the folder and performs the personalization.
- The Muehlbauer clp52 machine used in this project uses an internal MSSQL database and the jobs are prepared and sent to the database using some stored procedures and user credentials.
- The IXLA ID5 machine uses completely different methods to communicate with the software that will personalize the cards. The communication is done via a TCP socket where the machines wait for commands to do different tasks like moving the card, laser engraving, autoposition, etc.

To integrate the plugins on the system we developed a module that checks for ".dll" files on plugins folder and analyzes every file to verify that it is a plugin compatible with the interface we provided to the developers of third-party plugins. The plugins that are compatible with the requirements are loaded on a plugin list and based on their name/attributes we use them.

4. CONCLUSIONS

The reflective programming allows the developers to add the capability load and use third-party plugins to add the functionality of the product. This is a very potential technique and at the same time, it is very risky because it changes the structure and the control flow of an application on runtime and hackers can use it to steal and modify the data using a plugin developed by them.

To minimize the risk, and to increase the security of our software there are different techniques like loading only known plugins, sending and receiving the data between the application and the plugin in an encrypted way, etc. In our project, we verify if a plugin is a trusted one by getting the HASH when we load them and compare them to the known HASH-s. To ensure the data is not modified we use end to end encryption using AES and ECDsa algorithms.

REFERENCES

[1] Aaron Stump. " Directly Reflective Meta-Programming", https://homepage.divms.uiowa.edu/ ~astump/papers/archon.pdf.

[2] Francois-Nicola Demers and Jacques Malenfant, Reflection in logic, functional and object-oriented programming: a Short Comparative Study.

[3] Benjamin Livshits, John Whaley & Monica S. Lam, Reflection Analysis for Java, Asian Symposium on Programming Languages and Systems, 2005

[4] Koved, L., Pistoia, M., Kershenbaum, A.: Access rights analysis for Java. In: Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp. 359–372 (2002).

[5] Hirzel, M., Diwan, A., Hind, M.: Pointer analysis in the presence of dynamic class loading. In: Proceedings of the European Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp. 96–122 (2004).