PROCEEDINGS OF SCIENCE

Smooth migration of a feature-rich vulnerability analysis engine within a security portal for users with divergent skill levels

Tadashi Murakami*
High Energy Accelerator Research Organization (KEK), Japan
E-mail: tadashi.murakami@kek.jp

We have been operating a vulnerability management portal site for the DMZ network, namely the network which allows connections from the Internet. In the DMZ network, security management is crucially important, and vulnerability management is useful for maintaining security. The portal site is named DMZ User's Portal and has been successfully operating for 13 years. For DMZ User's Portal, we have adopted the same series of a vulnerability analysis engine, which has many advantages, but a more powerful inspection performance of the engine has gradually become required in preparation for today's hard security circumstances. Now, we decided to replace the engine with a more powerful and complex one. With the replacement, it is desirable to continue the successful experiences and contributions of the portal site. However, it is quite a difficult task without the careful design and development of the modules in advance.
This paper presents the design and methods for the smooth migration of the feature-rich vulnerability analysis engine within the security portal site. The key point is the careful consideration of the module dependency. To achieve a lower degree of module dependency, the techniques of Object-Relational (O/R) mapping, code generation, wrapper architecture, template engine consolidation, and test case were leveraged. We can continue to operate the portal site while inheriting the successful experiences as well as gaining the benefits of a new and powerful vulnerability analysis engine.

International Symposium on Grids & Clouds 2019, ISGC2019
31st March - 5th April, 2019
Academia Sinica, Taipei, Taiwan

*Speaker.

PoS(ISGC2019)013

1. Introduction

Vulnerability management [1] is useful for maintaining security with keeping the flexibility of the network environment, especially for the DMZ network, which allows connections from the Internet. As a part of computing and network services in KEK, we have been operating a vulnerability management portal site named DMZ User's Portal for 13 years [2, 3].

For vulnerability scanning [4, 5] in DMZ User's Portal, we have offered a vulnerability analysis engine which has advantages in presenting countermeasures. The countermeasures have been useful references for vulnerability management in the DMZ network in KEK and has been adopted for long years with hardware replacement for several times. But a more powerful inspection performance of the engine has gradually become required in preparation for today's more difficult security circumstances security circumstances. Now, we decided to replace the engine with a more powerful and complex one. For the replacement, it is desirable to continue the successful experiences and contributions of the portal site, such as complexity-hiding, the knowledge-base repository, and covering divergent skill levels among the host administrators in the DMZ network (DMZ admins). However, this is quite a difficult task without the careful design and development of the modules in advance.

This paper presents the design and methods for the smooth migration of a feature-rich vulnerability analysis engine within DMZ User's Portal. The key point is the careful consideration of the module dependency as regards the essential functions of the vulnerability analysis engine, web interface, email notifier, and database. To achieve a lower degree of module dependency, the techniques of Object-Relational (O/R) mapping, code generation [6], wrapper architecture, template engine consolidation, and test case were leveraged. The combined use of these techniques has brought not only modularity but also maintainability to the modules for the essential functions. Consequently, it has enabled us to replace the vulnerability analysis engine with minor modifications of the user interfaces within the web and email. In this way, we can continue to operate the portal site while inheriting these successful experiences, as well as gaining the benefits, of a new and powerful vulnerability analysis engine.

The organization of this paper is as follows. Section 2 describes our 13-year-long experience of vulnerability management with DMZ User's Portal. Section 3 discusses the replacement of the vulnerability analysis engine. Section 4 presents the design and techniques for the smooth migration of the vulnerability analysis engine. Finally, Section 5 summarizes the paper.

In the CHEP2018 conference and the subsequent paper [2], the experience with DMZ User's Portal has already been described. Since the subject of this paper is the migration of the vulnerability analysis engine, Section 2 briefly reviews the experience. In addition, this paper presents the flow of annual self-inspection (with Figure 1) for the first time.

2. DMZ User's Portal – the experience of vulnerability management using a portal site

In some research institutes, such as KEK, DMZ servers are operated with various kinds of demands, such as various usage (experiment, login shells, public relations, etc), various type of research and users, and various kind of groups (e.g., number of members varies from 10 to over

1000). In some of the cases, a login shell service may be used by hundreds of accounts in various countries. Naturally, the skill levels among the host administrators vary widely.

In such institutes, the command-hierarchical manner is not suitable because it is difficult to cover various circumstances. In other words, variety and flexibility are crucially important in these research institutes.

DMZ User's Portal promotes self-security management for DMZ admins by providing easy-to-operate user interfaces so that the DMZ admins can manage and handle the vulnerabilities by themselves. Using DMZ User's Portal, DMZ admins can conduct vulnerability scans on their own. Also, the owners of the hosts that are having serious vulnerabilities can receive notifications according to the results of the weekly automatic scans conducted by the system of DMZ User's Portal. In this way, DMZ User's Portal supports variety and flexibility in security viewpoints.

Another important point is that DMZ User's Portal is a modular system. DMZ User's Portal uses the middlewares of a relational database, a vulnerability scanner, a web server, and a mail server. Each of the middlewares is wrapped by its own wrapper module. This architecture improves module dependency between the modules and enables us to realize the smooth migration of the feature-rich vulnerability analysis engine within DMZ User's Portal.

In KEK, all of the DMZ admins have their own accounts for the portal site and can manage the vulnerabilities by themselves. Moreover, the portal site was adopted for two other sites with different operational policies and is now in operation there.

Figure 1 shows the flow of an annual self-inspection conducted in KEK. In (1), a DMZ admin can check the scan results at any time. In (2), if there are no severe vulnerabilities in the results, the flow proceeds to (5) and the self-inspection is finished. If there are any serious vulnerabilities, the flow proceeds to (3) and the DMZ admin handles the vulnerabilities. Also, the DMZ admin can handle the vulnerabilities by reporting to prepare a supplemental report that includes mitigation measures, proofs of false positives, etc. When the measures are completed, the flow proceeds to (4) and the DMZ admin executes an on-demand scan again. Next, the flow returns to (1) and the DMZ admin checks the scan results again. In this way, the DMZ admin can execute on-demand scans for any number of times to cope with the vulnerabilities.

3. Replacement of the vulnerability analysis engine

In DMZ User's Portal, we have offered a vulnerability analysis engine which has advantages in presenting countermeasures for 13 years. Using DMZ User's Portal, all DMZ admins came to handle the vulnerabilities in the order of the severities detected by the vulnerability analysis engine promptly. Especially, they came to cope with the serious vulnerabilities promptly. On the other hand, a more powerful inspection performance of the engine has gradually become required in preparation for today's more difficult security circumstances.

It is inevitable for a powerful vulnerability analysis engine to detect more numbers of false-positive vulnerabilities than less powerful ones. Without reliability for the vulnerability management technologies, it is difficult to manage false positives properly. Through the long-year experience, DMZ admins came to get accustomed to and have the reliability for vulnerability management technologies. Now, we decided to replace the engine with a more powerful and complex one.

For the replacement, it is desirable to continue the successful experiences and contributions of the portal site. One of the important contributions covers the divergent skill levels among the DMZ admins in KEK, by simplification of the intricate use of the vulnerability analysis engine. Note that the analysis engine itself is designed for network security experts and too complex for most of DMZ admins to use. It is important to design and implement complexity-hiding. Another important contribution is the portal site's accumulation of much know-how gained from the portal site's operation, which includes the user-responses of the DMZ admins. The portal site acts as a knowledge-base repository.

Without the careful design and development of the modules in advance, it is almost impossible to continue the contributions above, especially for a large replacement such as a vulnerability analysis engine.

4. Design and techniques for smooth migration of vulnerability analysis engine

4.1 Overview

This paper presents the design and methods for the smooth migration of a feature-rich vulnerability analysis engine within the security portal site, DMZ User's Portal. The key point is the careful consideration of the module dependency as regards the essential functions of a vulnera-

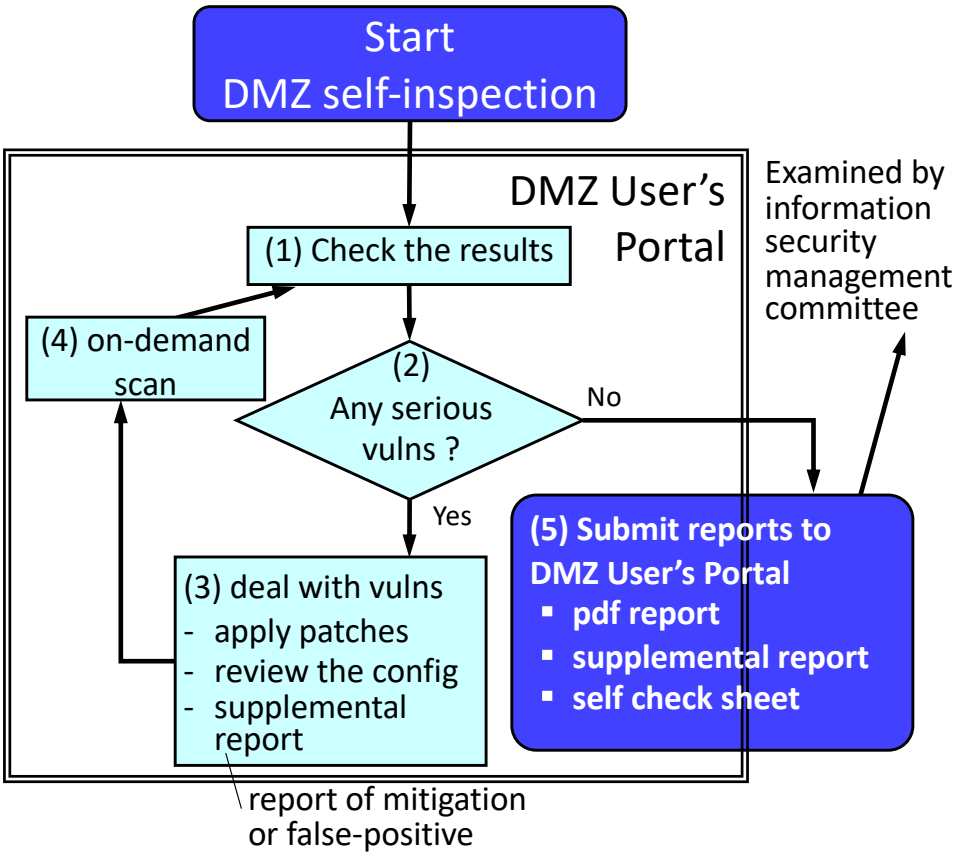

Figure 1: Self inspection for DMZ hosts.

bility analysis engine, web interface, email notifier, and database. To achieve a lower degree of module dependency, the techniques of O/R mapping, code generation [6], wrapper architecture, template engine consolidation, and test case were leveraged. The combined use of these techniques has brought not only modularity but also maintainability to the modules for the essential functions mentioned above. Module independence and maintainability brings the advantage of requiring only minor modifications to replace the vulnerability analysis engine, especially the user interfaces within web and email. In this way, we can continue to operate the portal site while inheriting the successful experiences, as well as gaining the benefits, of a new powerful vulnerability analysis engine.

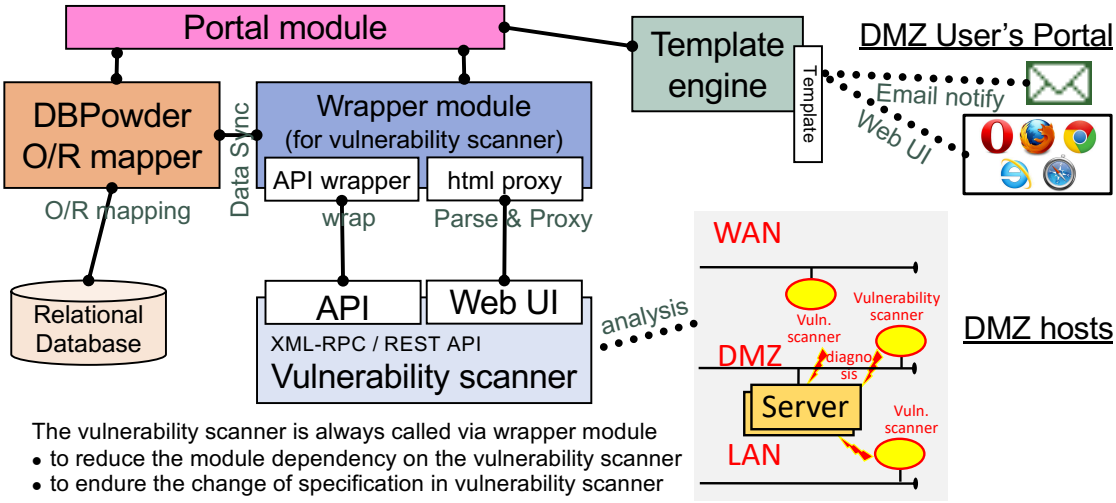

Figure 2: System architecture of DMZ User's Portal (minor modifications of [2]).

Figure 2 shows the system architecture of DMZ User's Portal. The portal module uses the other three components and organizes the functions. The wrapper module for the vulnerability scanner receives and handles all requests related to the vulnerability scanner. DBPowder receives and handles all requests related to the relational database. DBPowder also helps the development around the relational database. The template engine generates web UI and email texts. Each module acts as an interface by wrapping the middlewares of the relational database, vulnerability scanner, web server, and mail server. The wrapper architecture improves the usability and maintainability of the target module by providing an API in a programming language in a stable way that absorbs the change of the usage when upgrading the target module. Therefore, the wrapper module improves module dependency by being controlled by the management module.

This section continues as follows. Section 4.2 describes the techniques for code generation, and Section 4.3 describes the techniques for O/R mapping. Section 4.4 describes the techniques, which includes object-oriented design, separation of the API module, code generation, and test case, for the APIs of the vulnerability scanner. Section 4.5 describes the wrapper technique and Section 4.6 describes template engine consolidation.

4.2 Code generation

Instead of code being written by a developer, a code-generation technique generates code by

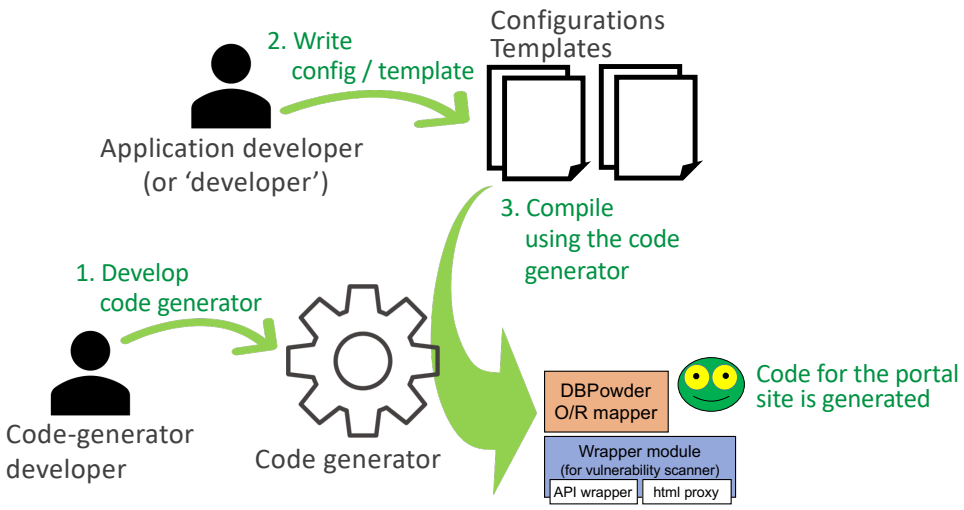

Figure 3: Code generator.

using a configuration file or a template file. Figure 3 shows the procedures. A code generator is developed by a code-generator developer in the first step. In the second step, an application developer (hereinafter simply called 'developer') writes configurations and/or templates. In the third step, the developer compiles the config or template using the code generator developed in the first step. The output of the third step is the required code (e.g., for the portal site).

Using a code generator, similar descriptions and their iterations can be parameterized and split into configurations and templates. For example, the values of an important parameter, such as the behavior of an API, can be converted into the behaviors themselves as function or method calls, with gathering into one template code with parameterized configurations including the behavior. Section 4.4.2 shows a useful usage.

4.3 O/R mapping: data handling program and database

A relational database is suitable for the management of complex data. In this scenario, the same data may be used in multiple contexts.

Figure 4 shows a simple example of a user-register-host data schema. Despite the data representation's being simple in the user, host, and register, there may exist two contexts, such as "A user has multiple registered hosts" and "A host is managed by multiple admins." If the data have to be managed entirely by objects in one representation, classes have to be separated into a normalized form, and each of the classes has to be managed with bi-directional links. Such a universal expression in an object (e.g., in Object code (3) in Figure 4) is too complex to use in each application context.

To address the complex situation above, we developed DBPowder [7, 8] O/R mapping framework as another research project. "O/R mapping" means the mapping between Object and Relation (relational table). DBPowder realizes multiple object data representations for one relational data representation, to offer suitable outputs for both database schema and object code.

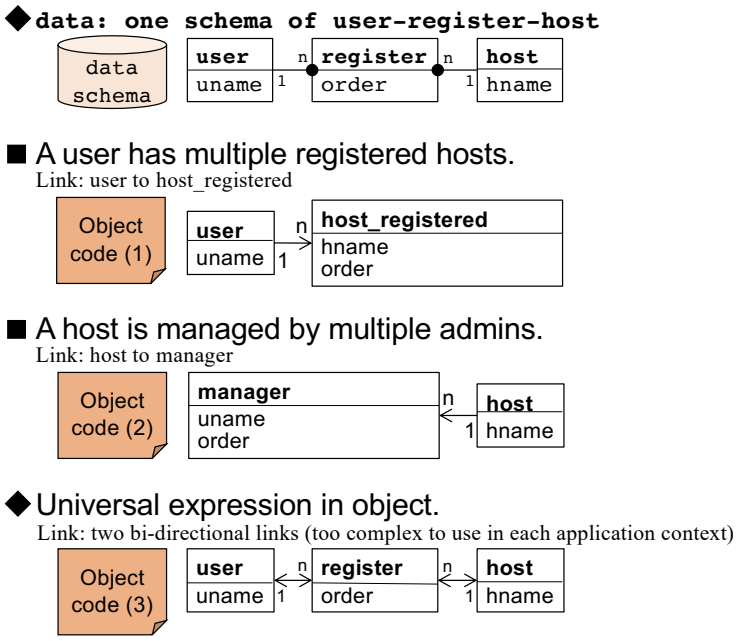

Figure 4: An example of user-register-host data schema in multiple contexts.

4.4 APIs for vulnerability scanner

For the APIs of the vulnerability scanner, module dependency is more carefully considered. Figure 5 shows the challenges of using APIs for vulnerability scanner. The one is that simple, patterned, and cumbersome implementation is required. The other is that the behavior of the vulnerability scanner may change when its version is upgraded or the scanner is replaced. To address the challenges, we adopted the techniques shown in the following subsections. Section 4.4.1 describes an object-oriented design used with the APIs. Section 4.4.2 describes the separation of the API module. To achieve this, a code-generation technique is used. Section 4.4.3 describes a test case technique adopted in the APIs.

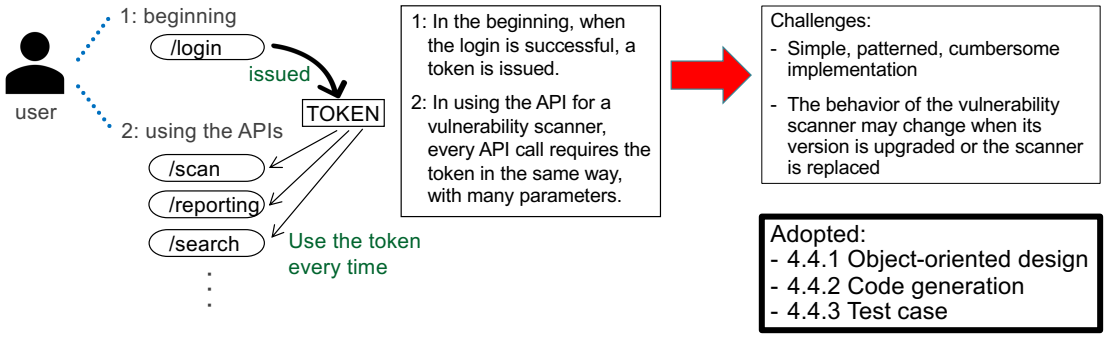

Figure 5: Challenges of using APIs for vulnerability scanner.

4.4.1 Object-oriented design

The API for a vulnerability scanner is usually supplied with some simple messaging architec-

ture, such as XML-RPC [9] or REST-API [10]. In many cases in the beginning of using the API, login is mandatory using the messaging architecture adopted in the vulnerability scanner (e.g., REST-API in our new scanner.) When the login is successful, the API issues a session token. After the login, every call of the API has to set the session token in the same way. For example, the XML-RPC-based APIs of previous scanner requires the session token as a top parameter value in the XML-RPC method call, and REST-API-based APIs of the new scanner requires the session token as a field value in the 'X-SecurityCenter:' http header.

For a programming language, it is desirable for the series of API to be provided by libraries of object-oriented design. For example, a session object is created when a login is successful. The session object, which knows the username, calls the APIs,

4.4.2 Code generation to separate API module

Some of the vulnerability scanners, including our previous and new scanners, have their own APIs. In the web age, a native binding of the APIs for a programming language is not usually supported because there are many candidates of programming languages. Therefore, to use the APIs, it is mandatory to write many lines of cumbersome interface codes between the APIs and the programming languages selected for the API implementation. To make matters worse, the behavior of the vulnerability scanner may change when its version is upgraded or when the scanner is replaced. Therefore, it is important to separate the interfaces between the vulnerability scanner and other modules.

The APIs often have many parameters for handling. The many parameters bring a simple, patterned, and cumbersome implementation of the APIs. To avoid this, it is more suitable to use a code generator and implement an API library.

For example, the API for a vulnerability scanner (e.g., XML-RPC or REST-API) requires the same remote-procedure-call method in different operations such as scanning a host, getting a result of a scan, registering a new host, and so on. In such a case, it is desirable for an application code to be converted from the parameters about behavior (e.g., scan, get a result, register a host) into the behaviors themselves as function or method calls. Using a code generator, all of their conversion codes can be gathered into one template code with parameterized configurations including the behavior.

4.4.3 Test case

To check the behavior of the APIs, it is useful to prepare a test case in which the parameters for the method (or function) calls and expected return values are written in advance. By simply running the test case, developers can check whether the API works as well as it had before the test case.

A test case is especially useful when a version upgrade or replacement of the vulnerability scanner is executed. If the result of the test case is successful, the API can operate as before.

4.5 Wrapper: wrap-generated code

The code generator (Section 4.2) overwrites the existing generated code when a developer reruns the code generator. Even if a developer modifies the generated code, any of the modification parts will be overwritten when in regeneration.

In the portal environment, the code generators of the O/R mapper and API for the vulnerability scanner add wrapper codes with the generated codes. A wrapper code simply wraps the original code by using inheritance. The wrapper has no logic and is never rewritten by the code generators. Therefore, developers can customize the behavior of the generated code by modifying the wrapper code. Here, a wrapper acts as a buffer zone for the generated code and developers' code.

4.6 Template engine consolidation

Messaging, mainly in web and email, is an important part for a portal site. In the development of DMZ User's Portal, the messaging part is separated from the program code into resources, of which each has its variable text (e.g., hostname, vulnerability name list, the person's name of the DMZ admin), and the values can be assigned by the program code.

The FQDN hostname itself can act as a variable. In DMZ User's Portal, the FQDN variable switches the entire set of the resources in the messaging part. This mechanism enables us to supply the portal at three different sites.

5. Summary

We have been successfully operating a portal site named DMZ User's Portal for 13 years. This paper describes the design and development issues of the modules in preparation to replace the vulnerability analysis engine. Without careful preparation, it is almost impossible to inherit the successful experiences and contributions of the portal site.

This paper discusses the key techniques, which are code generation, O/R mapping, wrapper, APIs for a vulnerability scanner, object-oriented approach, separation of API module, code generation, test case, and template engine consolidation.

The design of the migration is already complete. The survey and trial-run of the API of the new engine are also complete. A new vulnerability scanner already operates a weekly automatic scan. However, we are still in the process of adopting the new scanner into DMZ User's Portal.

Acknowledgment

The author gives special thanks to F. Yuasa (KEK), the security group of Computing Research Center in KEK, the information security management committee (KEK), and all users of DMZ User's Portal, for fruitful discussions and advice.

References

[1] Park Foreman, *Vulnerability Management*, CRC Press, Florida (2009)

[2] T.Murakami, *Design and development of vulnerability management portal for DMZ admins powered by DBPowder*, in proceedings of *CHEP 2018*, EPJ Web of Conferences 214, 08014 (2019)

[3] T.Murakami, et al., *Long-term experiences in keeping balance between safety and usability in research activities in KEK*, in proceedings of *CHEP 2018*, EPJ Web of Conferences 214, 08001 (2019)

[4] K.Scarfone, et al., *Technical Guide to Information Security Testing and Assessment*, *NIST Special Publication 800-115*, September 2008

[5] Category:Vulnerability Scanning Tools, `https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools`

[6] D.Thomas, A.Hunt, *The Pragmatic Programmer: From Journeyman to Master*, Addison-Wesley Professional, Boston (1999)

[7] T.Murakami, *DBPowder-mdl: Mapping description language between applications and databases*, in proceedings of *ICIS 2008*, IEEE/ACIS, pp. 127 – 132, May 2008

[8] T.Murakami, T.Amagasa, H.Kitagawa, *DBPowder: A Flexible Object-Relational Mapping Framework Based on a Conceptual Model*, in proceedings of *COMPSAC 2013*, IEEE Computer Society, pp. 589 – 598, July 2013

[9] XML-RPC Specification, `http://xmlrpc.scripting.com/spec.html`

[10] What is REST – Learn to create timeless RESTful APIs, `https://restfulapi.net/`