Sofia University "St. Kliment Ohridski"
Faculty of Mathematics and Informatics
Department "Computing systems"

DEVELOPMENT OF HIERARCHICAL TAXONOMY THAT INCORPORATES PATTERNS FOR IMPROVING SECURITY IN INFORMATION SYSTEMS BASED ON MICROSERVICE ARCHITECUTRE

Synopsis

of the PhD Thesis
conducted for the purpose
of receiving the academic title
"Doctor of Philosophy" (PhD)
in the field of
4.6. Informatics and Computer Science

Submitted by: Mag. eng. Tihomir Dimitrov Tenev
Advisors: Assoc. prof. Simeon Emilov Tsvetanov, PhD

Assoc. prof. Dimitar Y. Birov, PhD

Sofia, 2020

CONTENT

List of figures

INTRODUCTION

The main goal of the thesis is to develop a hierarchical taxonomy of patterns for improving security in information systems.

The proposed taxonomy is entirely dedicated to a next-generation software architecture, Microservice Architecture, which enables developers to create flexible applications that cover in a short period of time heavy workloads, without influencing of the normal operation.

Threats were analyzed for each of the areas defined for architecture of this kind, enabling more accurate positioning of the selected security patterns.

The description of the hierarchical taxonomy is accomplished using an object-oriented modeling language that comprises a language to define interfaces in term of "Managed Object Format".

A graphical interface was used to present ideas on paper so that they could be graphically depicted. It illustrates the links between defined areas and selected patterns.

Modern microservice management tools are reviewed and solutions are proposed following the best practices of selected security patterns from the hierarchical taxonomy of security patterns.

The dissertation is divided into eight chapters, each of them has conclusion.

Chapter one discusses microservice architecture with a conceptual application. The need of security enhancements is justified and the benefits of using security patterns are considered.

Chapter two reveals the threats in the Account and Identity area, gives a list of security patterns, and provides recommendations for each pattern.

Chapter three reveals the threats in the Communication area, gives a list of security patterns, and provides recommendations for each pattern.

Chapter four reveals the threats in the Storage area, gives a list of security patterns, and provides recommendations for each pattern.

Chapter Five reveals the threats in the Microservices environment, gives a list of security patterns, and provides recommendations for each pattern.

Chapter Six reveals the threats in the Microservices application distributed across platforms from different vendors, gives a list of security patterns, and provides recommendations for each pattern

The seventh chapter is devoted to the construction of a hierarchical taxonomy of security patterns for information systems based on microservice architecture, a methodology for describing a hierarchical taxonomy, the relationship definitions between different areas, creation of a graphical interface and analyzing the achieved results.

Chapter Eight describes the implementation of modern microservice management tools that can be configured following the best practices of selected security patterns from the Hierarchical Taxonomy of Security Patterns. Solutions are offered according to the requirements set in each of the patterns.

1 I CHAPTER. IMPROVING THE SECURITY OF SOFTWARE SYSTEMS BASED ON MICROSERVICE ARCHITECTURE

1.1 *Microservice Architecture.*

A modern solution to the problem of managing large software applications is to use software architecture for developing products called „Microservice Architecture“ [1]. It has many advantages over the rest software architectures. An example is the reduction of complexity in large applications by breaking down the business logic toward smaller functional units. These units are called microservices. Each unit performs a specific activity and does not override someone else's functionality.

Figure 1 shows a conceptual architectural model of an online store using Microservice Architecture. It shows only some of the basic functionality required in case of reviewing and ordering an item. The individual modules are designed to be built with the help of different programming languages, in order to represent the flexibility of this type of architecture.

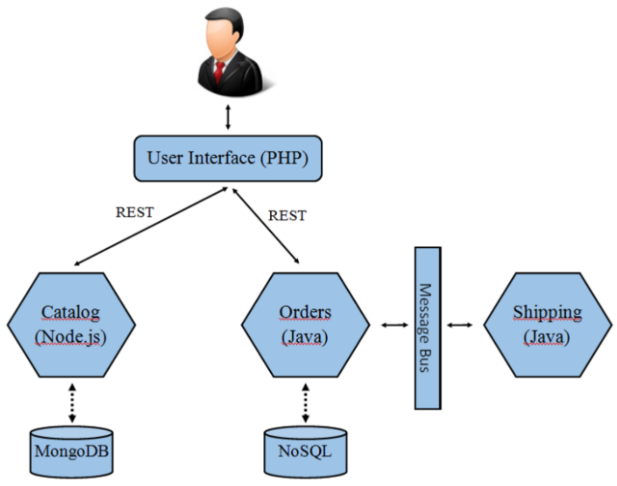


Figure 1 Conceptual architecture model

1.2 *Security patterns for enhancing the security.*

Nowadays, a major challenge for industries is to increase security across the hosts and their software. A good approach is to ensure the security from the very beginning of designing and building a specific system and its associated software [2]. Otherwise, security threats can be expected at any time during operation. In this context, finding a way to catch the threats and implementing it in the right way and place will greatly facilitate the job of programmers and engineers. An example of such a way of counteraction is the approach of applying Security Patterns.

The term Security Patterns comes from the so-called Design Patterns [3]. As of Design Patterns, each security pattern has its own structure. It serves for better reading and applying. Most of the models selected and shown in the following chapters contain the following points - *Intent, Context, Problem, Solution, Structure, Implementation, Consequences* and *Known uses*. For a more intuitive selection of security patterns, it is decided to consider two main points - *Context* and *Solution*. The *Context* describes the nature of a situation, which includes domain assumption and expectation of a system environment. The *Solution* guides in solving a problem by providing some information for correcting the issue.

The idea of using Security Patterns lies on the research done towards Design Patterns [4] [5]. The result of this study shows that the implementation of Design Patterns significantly contributes to the improvement of software functionality.

A major difficulty encountered when working with security patterns is that they provide solutions for common problems. After a thorough investigation of each patter a satisfactory transformation was achieved. The transformation is presented in the form of recommendations and is described in chapters 2, 3, 4, 5 and 6.

1.3 Vulnerability analysis and recommendations when selecting security patterns.

After extensive research, it was decided to use a STRIDE based vulnerability analysis approach [6]. This approach advises readers to divide an application into several components for more accurately identifying the types of attacks that might be encountered. The STRIDE acronym stands for: **S**poofing, **T**ampering, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, and **E**levation of Privilege.

The next few chapters shows some of the eventual threats in different areas of microservice architecture. Then all the selected security patterns are categorized so that they can be classified into the different categories of STRIDE. Some patterns cover more than one of the STRIDE categories.

Chapters 2, 3, 4, 5 and 6 present several lists of security patterns for each of the defined areas. The advantages and disadvantages of using a microservice architecture to develop a software application, the need of security enhancements and the result of threat analysis are taken into account. Each of the patterns is accompanied by a recommendation for a place of use that would assist developers in the process of building a sustainable application based on microservice architecture. The recommendations for the place of use of each security pattern are derived from its structural points - *Context* and *Solution* and then adapted to a specific situation.

1.4 Constructing a hierarchical taxonomy of security patterns

The hierarchical taxonomy of security patterns presented in Chapter 7 is achieved through the analysis made in the next few chapters - 2, 3, 4, 5 and 6. All microservice architecture categories are covered in the above chapters, as well as a list with Security Patterns. The categories are: Account and Identity, Communication, Data Persistence, Environment, and Third-Party Suppliers. For greater precision, the "Communication" area is divided into two sub-areas - "Inter-connection" and "Messages". The description of the taxonomy itself was made using object-oriented CIM (Common Information Model) modelling [7]. CIM is a conceptual information model for describing different entities that contributes to the structuring of security patterns collected so far. The entire ontology has been transformed graphically using the free online UI Wireframing [8].

1.5 Choosing modern technologies for microservice management

Each of the five areas of microservice application can be managed through different technologies. The choice of advanced microservice management technologies is described in Chapter 8. The Account and Identity area can be managed using Kong [9]. The Communication area is divided into two sections, Inter-connection and Messaging. The tool that can be used for communication between microservices is RabbitMQ[10]. The tools that are considered in the Storage area are: MongoDB database [11] as well as Filesystem type ext4 [12] represented in Operating System CentOS [13]. Tools offering functionality that can be implemented in the Microservices environment are Docker container [14], CentOS [13] and IaaS [15]. The Microservices platform deployed on multiple vendor platforms can be managed using clustering tools like Kubernetes (K8s) [16].

MAIN PURPOSE AND TASKS OF THE DISSERTATION WORK

The purpose of the dissertation is to develop a hierarchical taxonomy of patterns for improving security in software systems based on microservice architecture. The taxonomy lies on the individual categories defined for the microservice architecture, as well as the vulnerability analysis made for each of these categories. This enables more accurate selection of Security Patterns.

The following tasks are formulated in relation to the main purpose:

1. To categorize and granulate new generation software architecture, also called Microservice Architecture, for its presentation in different forms.

2. To analyze the threats for each of the presented categories of microservice architecture.

3. Identifying appropriate security patterns that fall into categories, relying on vulnerability analysis.

4. To transform the justifications and solutions provided by the various security patterns into the context of a microservice architecture.

5. To make a hierarchical model of all categories of microservice architecture, to serve as a skeleton in the construction of a detailed hierarchical taxonomy.

6. Identifying and using appropriate object-oriented modeling language to bring selected security patterns.

7. Transform the modeling language so that it can be represented graphically.

8. Finding new products that can create a sustainable environment for microservices.

9. Finding ways of applying the chosen products for building Microservice environment based on the solutions from Security Pattern.

2 II CHAPTER. PROPOSALS FOR MITIGATION OF THREATS IN ACCOUNT AND IDENTITY

2.1 Conceptual model

One of the goals in this section is to highlight the vulnerabilities that could be experienced by a microservice-based software application with external users - Figure 2. A case is also considered where for some reason the microservice is placed with a vulnerability within (Orders), which may interfere toward the application.

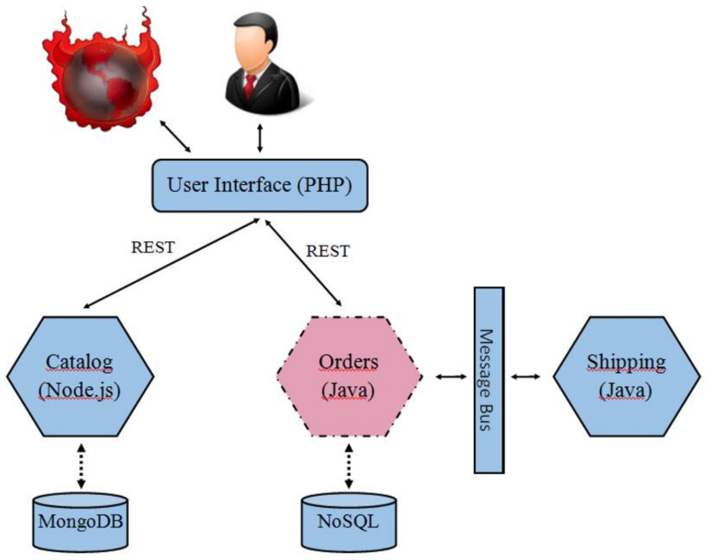


Figure 2 Attempt for breaking the functionality in Microservice application

2.2 Vulnerability analysis

The current point is analyzing threats on a system based on microservice architecture. The threats could come from both external users and microservices located in the same domain. Taking into account the conditions described for each of the points in the STRIDE threat detection model and interpreting them in the context of the selected area “Account and Identity environment”, the following short analysis can be made:

Spoofing is a type of abuse in which an intruder tries to gain access to a system or information of a user by pretending to be the user himself.

Tampering means making unlawful changes to adjust a key in order to access a microservice or manipulate information for work sessions.

Repudiation is interpreted as a denial of acceptance of a claim.

Information Disclosure can occur when unauthorized users access information that is prohibited from disclosure.

Denial of Service refers primarily to the depletion of resources on one or more server units that host microservices.

Elevation of Privilege means allowing a user to execute a command without having the necessary privileges.

2.3 Recommendations for choosing security patterns

Security patterns related to the “Account and Identity” category are offered. Recommendations for places where they could be used have been added to each pattern. The list of all patterns, as well as the STRIDE category which they belong to, is shown in Figure 3.

A paperwork in this field has been published [17].

Patterns	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
A Pattern for WS-Trust	▯		▯			▯
Access Control List	▯				▯	
Account Lockout	▯					
Actor and Role Lifecycle Pattern			▯	▯		
Administrator Objects	▯					▯
Authenticated Session	▯					
Authenticator	▯					
Authorization						▯
Biometrics Design Alternatives	▯					
Capability	▯				▯	
Client Input Filters	▯					▯
Credential Delegation				▯		
Directed Session		▯				
Grant-Based Access Control Pattern				▯		
Password Design and Use						▯
Privilege-Limited Role				▯		
Role-Based Access Control (RBAC)	▯					
Session-Based Attribute-Based Authorization	▯					▯

Figure 3 Security patterns for Account and Identity

Conclusion

The features of the microservice application in the section "Account and Identity" are considered. A case where for some reason a microservice is placed with a vulnerability in it is also considered.

A conceptual model of a microservice application on a cloud platform is proposed. Service models that are suitable for this type of application are also specified.

A link has been made between the microservice application and external users. The first example is with a direct User-Microservice connection and the second is by implementing of an additional tool - API Gateway.

Vulnerability analysis was performed on a system based on microservice architecture. The threats could come from both external users and microservices located in the same domain. The threat analysis approach used is STRIDE.

Security patterns that relate to the Account and Identity section are gathered.

A categorization of each of the security patterns using STRIDE based threat analysis has been made.

Recommendations have been added to each of the security models as of the place they could be used.

3 III CHAPTER. PROPOSALS FOR MITIGATION OF THREATS IN COMMUNICATION

3.1 *Conceptual model*

The goal is to highlight the vulnerabilities that may occur when communicating among microservices located in a single domain - Figure 4. For greater precision, it was decided to divide the communication into two points - " Inter-connection " and "Messages".

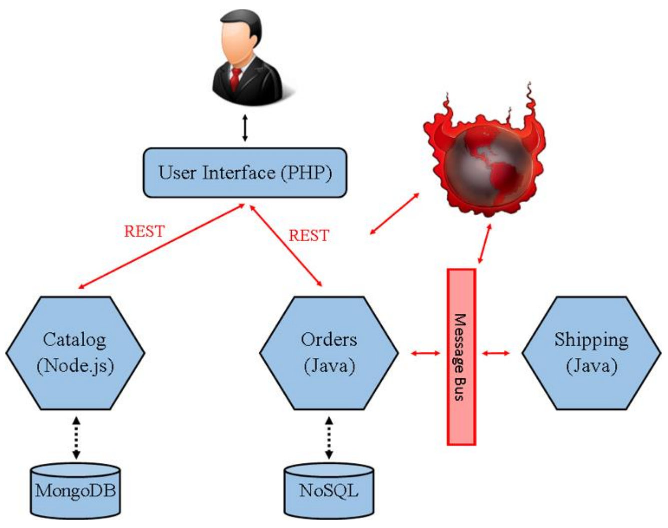


Figure 4 Attempt of eavesdropping of communication among microservices

3.2 *Vulnerability analysis*

The current point is analyzing the threats on a system based on microservice architecture. The threats could come from both external users and communication tools located in the same domain. Taking into account the conditions described for each of the points of the STRIDE threat detection method and interpreting them in the context of the selected Communication environment, and in particular Inter-connection and Messages, the following short analysis can be made:

Spoofing is a type of abuse in which an intruder tries to gain access to a bus (misleading that he is authorized like everyone else), where microservices are listening.

Tampering means making unlawful changes to the data stream when such actions are prohibited.

Repudiation is interpreted as a refusal to accept a claim.

Information Disclosure can occur when unauthorized users access information that is prohibited from disclosure.

Denial of Service refers primarily to the exhausting of resources of one or more server components used by microservices.

Elevation of Privilege means allowing a user to connect to a bus, where authorization is required.

3.3 Recommendations for choosing security patterns

Security patterns that are relevant to the Communication section and in particular "Inter-connection " and "Messaging" are proposed. Recommendations for places where they could be used have been added to each model. The list of all models, as well as the STRIDE category where they belong to, is shown Figure 5 and Figure 6.

A paperwork in this field has been published [18].

Patterns	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Authoritative Source of Data		▯			▯	
Content–independent Processing		▯		▯		
Input Guard		▯				
Multiple Secure Observers				▯		
Output Guard		▯				
Secure Channels		▯		▯		
Secure Communication				▯		
Security Association	▯	▯				
XML Firewall	▯	▯				

Figure 5 Security Patterns for Inter-connection

Patterns	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Anonymity Set				▯		
Cryptographic Protocol				▯		
DATA INTEGRITY IN P2P-SYSTEMS		▯		▯		
Hidden Metadata				▯	▯	
Layered Encryption				▯		
Morphed Representation				▯		
XML Encryption Syntax and Processing		▯		▯		

Figure 6 Security patterns for Messages

Conclusion

A microservice application in the Communication section is considered. For a deeper precision, the Communication part is divided into two sections "Inter-connection" and "Messages".

A conceptual model of a microservice application using REST and Message Bus communication tools is proposed. The considered communication approaches are - Synchronous and Asynchronous.

The different types of packages that can pass through the connectors of a microservice application - Binary, XML and JSON - are reported.

Threat analysis was performed on a system based on microservice architecture. The threats could come from both external users and microservices located in the same domain. The vulnerability analysis approach used is STRIDE.

Security patterns for Microservice communication area are proposed.

A categorization of each of the security patterns by STRIDE based threat analysis has been made.

Recommendations have been added to each security pattern as the place of use.

4 IV CHAPTER. PROPOSALS FOR MITIGATION OF THREATS IN DATA PERSISTANCE

4.1 Conceptual model

The main purpose of this section is to highlight the vulnerabilities that could be encountered when managing and storing data from microservices - Figure 7.

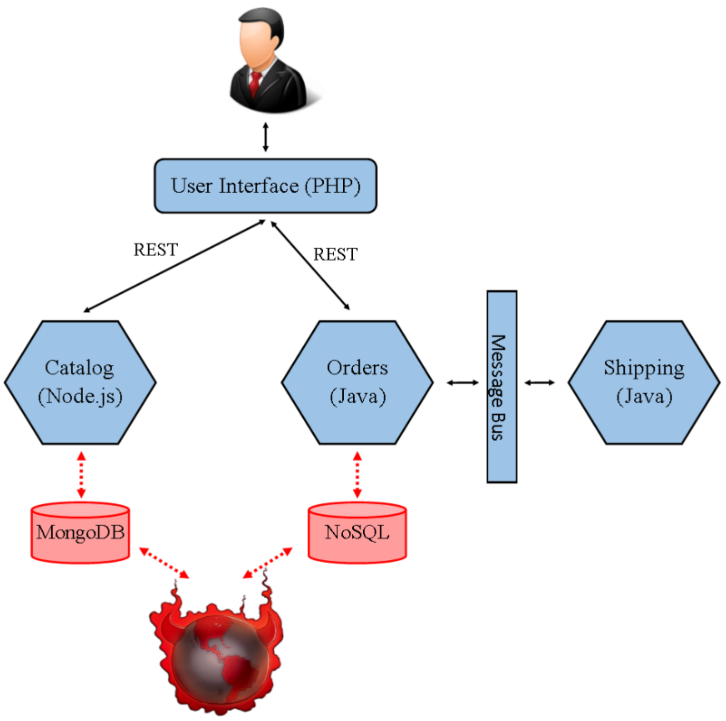


Figure 7 Attempt of manipulating data stored by microservices

4.2 Vulnerability analysis

The current point is analyzing threats to a system based on microservice architecture. Threats could come from both external users and tools that help manage and store data. Taking into account the conditions described for each of the points in the STRIDE threat detection model and interpreting them in the context of the selected data storage environment, the following vulnerability analysis is performed:

Spoofing is a type of abuse in which an intruder attempts to gain access to data owned by another microservice.

Tampering means making unlawful changes to a data stored by a microservice when such actions are prohibited.

Repudiation is interpreted as a refusal to accept a claim.

Information Disclosure may occur when an unauthorized user accesses unauthorized file.

Denial of Service refers primarily to the exhausting of resources on one or more server units running microservices.

Elevation of Privilege means allowing a user to access a non-privileged data warehouse.

4.3 Recommendations for choosing security patterns

Security patterns are suggested that relate to the Data Persistence section. Recommendations for places where they could be used have been added to each pattern. The list of all patterns as well as the STRIDE category where they belong to is shown in Figure 8.

A paperwork in this field has been published [19].

Patterns	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Clear Sensitive Information[8]				□		
Encrypted Storage[11]		□		□		
File Authorization[1]	□	□		□		
Limited View[10]				□		
Multilevel Security[1]		□		□		
Secure Directory[8]		□				

Figure 8 Security Patterns for Data Persistence

Conclusion

A conceptual model of microservice application in the Data Persistence section is discussed.

The requirement when designing a microservice application was considered - all data owned by a microservice should not be allowed for use by other microservices.

A variant for data exchange between microservices was considered. The tool that enables such exchange is Message Bus.

The advantage of microservice applications is easily transformed and adapt as a cloud solution.

Vulnerability analysis was perfomred on a system based on microservice architecture. The threats could come from both external users and microservices located in the same domain. The threat analysis approach used is STRIDE.

Security patterns that relate to the Data Persistence section are provided.

A categorization of each of the security patterns by STRIDE based threat analysis has been made.

Recommendations have been added to each security pattern as where they could be used.

5 V CHAPTER. PROPOSALS FOR MITIGATION OF THREATS IN MICROSERVICE ENVIRONMENT

5.1 *Conceptual model*

The problem explored here is how to create a secure environment that optimally accommodates microservices. In most cases, microservices can be distributed either on a standalone operating system or by using containers [1] [20] – Figure 9.

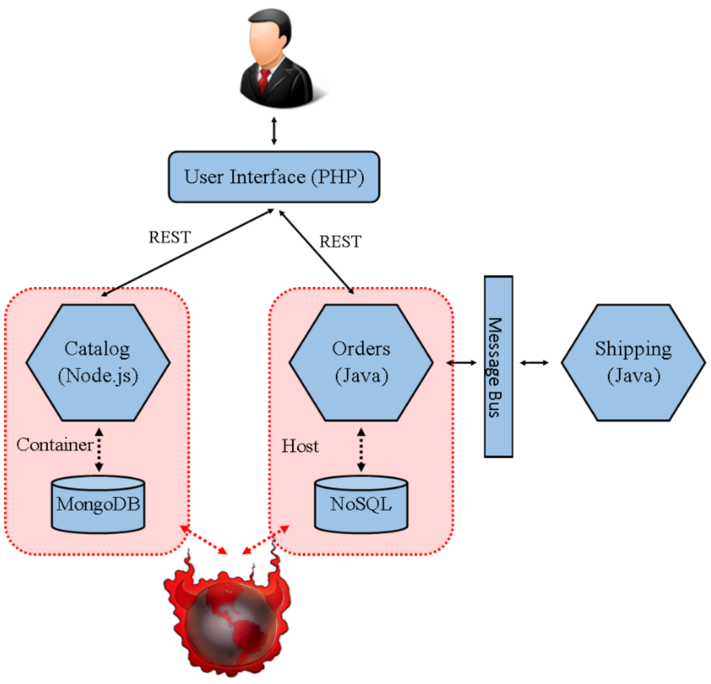


Figure 9 Attempt to influence the environment hosted by microservices

5.2 *Vulnerability analysis*

The current point is analyzing threats to a system based on microservice architecture. Threats could come from both external users and tools that help manage and store data. Taking into account the conditions described for each of the points in the STRIDE threat detection model and interpreting them in the context of the selected part of the Microsystem environment, the following short threat analysis is performed:

There is a type of fraud in which an intruder tries to gain access to both the system hosted by the microservices and to the information that the microservices possess, pretending to have the necessary authorization. These cases usually lead to unwanted consequences and fall into the **Spoofing** category.

Maintaining the integrity of an information is strictly imperative. The STRIDE data integrity protection category is **Tampering**.

If a user takes an action but claims it failed to do so, this would reduce liability. The STRIDE category to which this type of denial belongs is **Repudiation**.

Another situation is where an unauthorized user/microservice operates with information that is prohibited from disclosure. The closest STRIDE category is ***Information Disclosure***.

Another threat is when there is an attempt to overflow a hardware resource - memory, processor, data storage, or more. The category that relates to microservice ability is ***Denial of Service***.

Another example is when a user has the opportunity to do something without the necessary specific access rights. The category that looks at situations of this type is ***Elevation of Privilege***.

5.3 Recommendations for choosing security patterns

Security patterns proposed relate to the Microservice environment area. Recommendations are also added to each model for places where they can be used. The list of all patterns as well as the STRIDE category to which they belong is shown in Figure 10 and Figure 11.

A paperwork in this field has been published [21].

Patterns	Threat modelling approach					
	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
Administrator Hierarchy	▯		▯			▯
Building the Server from the Ground Up		▯			▯	
Checkpointed Systems					▯	
Controlled Execution Environment		▯				
Documenting the Server Configuration		▯		▯	▯	
Patching Proactively		▯		▯	▯	
Pathname Canonicalization		▯				
Protection Rings		▯				
Testing on a Staging Server					▯	
Secure IaaS/open IaaS/OpenStack	▯	▯	▯	▯	▯	▯

Figure 10 Security patterns for Microservice Environment

Patterns	Deployment approach	
	Standalone OS	Container
Administrator Hierarchy	▯	▯
Building the Server from the Ground Up	▯	
Checkpointed Systems		
Controlled Execution Environment	▯	
Documenting the Server Configuration	▯	▯
Patching Proactively	▯	▯
Pathname Canonicalization	▯	
Protection Rings	▯	
Testing on a Staging Server	▯	▯
Secure IaaS/open IaaS/OpenStack		

Figure 11 Types of deployment approaches for each of the security patterns

Conclusion

A microservice application in the Microservice environment area is considered. For the sake of clarity, several environments that can be deployed on a microservice are considered: Operating System, Container, and IaaS.

A conceptual model of a microservice application is offered. It represents three of the selected storage media: Operating System, Container, and IaaS. Considered are the advantages and disadvantages of the three types of environments in which microservices can be deployed

Vulnerability analysis was performed on a system based on microservice architecture. The threats could come from both external users and microservices located in the same domain. The threat analysis approach used is STRIDE.

Security patterns are proposed that relate to the Microservice environment area. A categorization of each of the security patterns by STRIDE based threat analysis has been made.

Recommendations have been added to each security pattern as to where they could be used.

6 VI CHAPTER. PROPOSALS FOR MITIGATION OF THREATS IN MICROSERVICES DISTRIBUTED ON PLATFORMS OWNED BY DIFFERENT VENDORS

6.1 *Conceptual model*

One of the tasks of this section is to highlight the vulnerabilities that could occur in software applications that have been decided to be split and positioned in two or more vendors – Figure 12

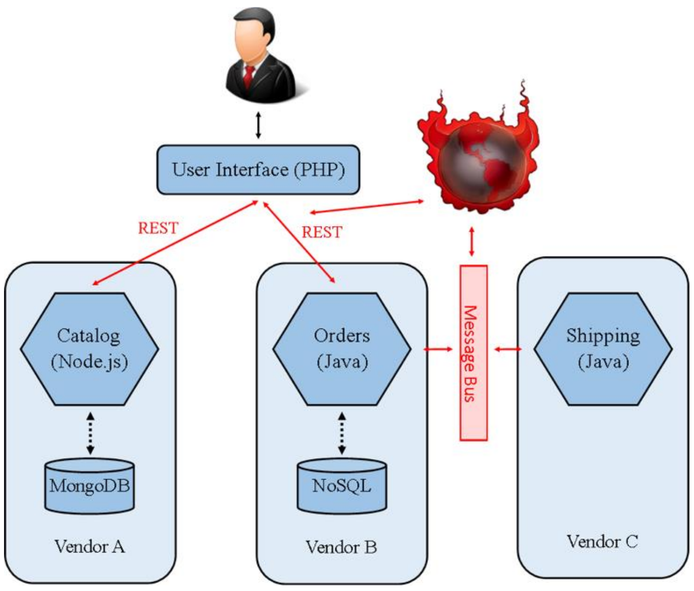


Figure 12 Attempt to influence the communication between microservices distributed in different providers

6.2 *Vulnerability analysis*

The current point is analyzing threats to a system based on microservice architecture. The threats could come from both external users and tools that enable communication between providers. Taking into account the conditions described for each of the points in the STRIDE Threat Detection Model and interpreting them in the context of the selected part "Microservices distributed on different vendor platforms", the following short threat analysis is made:

An example of ***Spoofing*** is when a microservice tries to request data from a microservice hosted by another provider that is already under the control of a malicious user.

Tampering means performing illegal activities to firewall settings.

Repudiation, or in particular a denial, to accept a claim.

Information Disclosure can occur when an unauthorized user manipulates configuration files of microservice environment without the necessary privileges.

Denial of Service addresses cases where resources are exhausted.

Elevation of Privilege addresses cases where a user has acquired rights that allow him or her to manage a software application hosted on multiple vendors.

6.3 Recommendations for choosing security patterns

Security patterns are proposed that relate to the part "Microservices distributed across multiple vendor platforms". Recommendations for places where they could be used have been added to each model. The list of all patterns, as well as the STRIDE category to which they belong, is shown in Figure 13.

A paperwork in this field has been published [22].

Patterns	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
3rd Party Communication		▯		▯		
AGENCY GUARD		▯		▯		
AGENT AUTHENTICATOR	▯					
Application Firewall	▯				▯	
Cloud Access Security Broker	▯					▯
Integration Reverse Proxy					▯	
Known Partners	▯			▯		▯
Log for Audit		▯	▯			
Secure Assertion		▯	▯		▯	
Secure Logger			▯			

Figure 13 Security Patterns for Microservices distributed across multiple vendor platforms

Conclusion

A microservice application in the section "Microservices hosted on platforms of different vendors" is discussed. The service model that has been considered is PaaS (Platform as a Service).

A conceptual model of a microservice application distributed over two suppliers is proposed.

Vulnerability analysis was performed on a system based on microservice architecture. The threats could come from both external users and microservices located in the same provider. The threat analysis approach used is STRIDE.

Security models are proposed that relate to the part "Microservices hosted on platforms from different vendors".

A categorization of each of the security patterns by STRIDE based threat analysis has been made.

Recommendations have been added to each security pattern as to where they could be used.

7 VII CHAPTER. DEVELOPMENT OF HIERARCHIC TAXONOMY OF PATTERNS TO IMPROVE SECURITY IN SOFTWARE APPLICATIONS BASED ON MICROSERVICE ARCHITECTURE

7.1 *Choosing a methodology for describing a hierarchical taxonomy*

The methodology chosen to describe the hierarchical taxonomy of security patterns relies on the rules declared in CIM (Common Information Model) [7]. CIM is a conceptual information model for describing different management entities, application sites, and service providers. Information models of this kind use different types of elements such as Classes, Properties, Methods, and Association. The following points provide examples of each of the used elements.

For the purposes of current research, the Interface Definition Language (IDL) or the so-called Managed Object Format (MOF) is used. MOF syntax is described in the form of notations defined in the Augmented BNF style for Syntax Specifications [23]. The main way of describing objects and derived instances is in text form. Comments on each element are only allowed in Unicode or UTF-8 format. In the current case, security patterns are used in the context of a software application based on a microservice architecture.

The common name "Security" is used as the schema name. The Scheme-Class connection is indicated by the "_" sign:

class Security_MicroserviceSecurity {}

The hierarchy classes begin with an abstract class that defines the basic elements that all subclasses will inherit. The name MicroserviceSecurity was selected as the abstract class. The classes that inherit the MicroserviceSecurity abstract class represent the individual areas of microservice architecture defined in Chapters 2, 3, 4, 5, and 6. All selected security patterns are also presented in the form of classes. Relations between classes representing security patterns and their associated areas are described in Section 7.2. The relationship between classes and abstract classes is defined using „:“ :

[Description (" ... „]
class Security_APatternforWSTrust : Security_MicroserviceSecurity {   }

Properties represent a value used to characterize a class. Properties are unique and are defined within the scope of the class. They consist of name, data type and value. Properties are defined in the abstract class „*MicroserviceSecurity*“ as:

[Override ("Context"), MaxLen (64), Description ("... "]
string Context;

Their value is given in the security pattern subclasses:

```
Context = " ... ";
```

Reference is a special type of property used as a reference pointer to a class. Reported with the keyword REF:

```
[Override ( "PatternAuthenticator" ),
Max ( 1 )]
Security_Authenticator REF PatternAuthenticator;
```

Association is a type of class that has two or more references. Associations are relations between two or more classes. It helps to group the different security patterns into their respective category:

```
[Abstract, Description (" .. ") ]
class Security_DataPersistance : Security_Environment {

        [Override ( "PatternClearSensitiveInformation" ),
        Max ( 1 )]
        Security_ClearSensitiveInformation REF PatternClearSensitiveInformation;
}
```

Qualifiers are values that provide additional information about classes, associations, properties, or references. All qualifiers have a name, type, value and scope. The qualifiers mentioned below are used in order to distinguish how many times (Max (1)) a security pattern can be applied in the context of the category to which it belongs:

```
[Override ( "PatternAdministratorHierarchy" ),
Max ( 1 )]
Security_AdministratorHierarchy REF PatternAdministratorHierarchy;
```

7.2 Defining relationships between different areas

Chapters 2, 3, 4, 5, 6 cover all five areas - *Account and Identity, Communication, Data Persistence, Environmen*t, and *Third-Party Suppliers*. For greater precision, the “Communication” area is divided into two sub-areas – “Inter-connection” and "Messages". Figure 14 shows the arrangement of all areas:

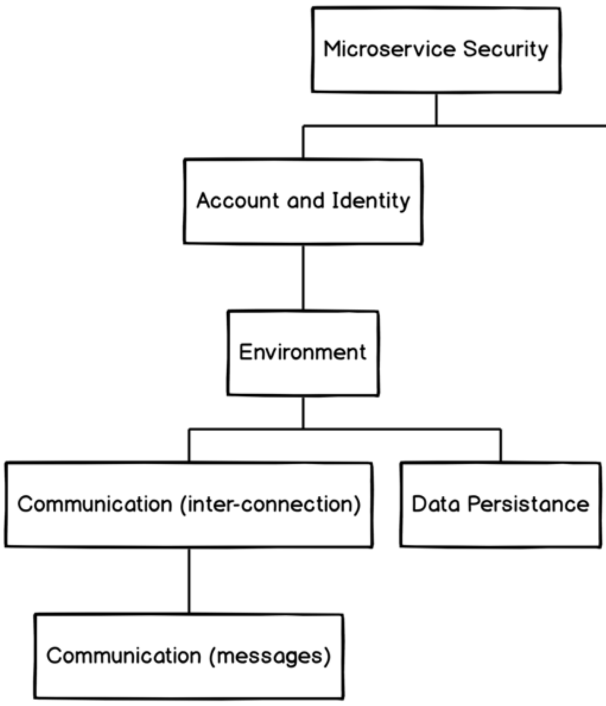


Figure 14 Hierarchy of Microservice Architecture areas

Microservice Security stands at the highest level in the hierarchy. It branches in two directions. The first is in the context of enhancing security only within the scope of one service provider [24] regardless of whether the software application is hosted on more than one vendor. If there are two or more suppliers, this branch applies to each supplier separately. The second branch covers only the cases where there is more than one service provider. It seeks to increase security between providers.

The *Account and Identity* area stands first on the top of Microservice Security branch. The reason of high importance is due to the fact that end users are the ones who create preconditions for information leakage. If users have unrestricted rights to the whole system, it would facilitate malicious external users to obtain these rights and do irreparable things. The next area is *Environment*. Making a proper configuration and maintaining the environment is essential to avoid unwanted behavior. Updating the main libraries used by the application will mitigate some vulnerabilities. The *Environment* breaks into two sub-areas. The first one is related to *Communication*, which takes place between individual microservices. The second is related to the information stored by each microservice - *Data Persistence*. *Communication* is divided into two sub-areas - *Inter-connection* and *Messages*. The first in that order is placed *Inter-connection,* because it is fundamental in communication between the individual microservices. The Messages appear when the connection between the microservices is established.

Security patterns are grouped and presented in Chapters 2, 3, 4, 5, and 6, and then depicted in Figure 3, Figure 5, Figure 6, Figure 8, Figure 10, and Figure 13. They are presented in tabular form with an explanation of which STRIDE category they belong to. Each of the areas shown in Figure 14 has a list of security patterns. All CIM cross-domain relationships are implemented through one of the functionalities called inheritance. All categories are presented in the form of classes.

7.3 Graphic representation of the hierarchical taxonomy

The data provided in the form of CIM object modeling can be displayed using various tools. The example being considered is by using the Wireframing UI [8]. Wireframing is a manner of presenting ideas on paper so that they can be graphically depicted. This helps to better illustrate the hierarchical taxonomy presented above.

Figure 15 shows a diagram of a software application using microservice architecture. Hyperlinks are added in order to give access to lists of each of the categories described in Chapters 2, 3, 4, 5 and 6.

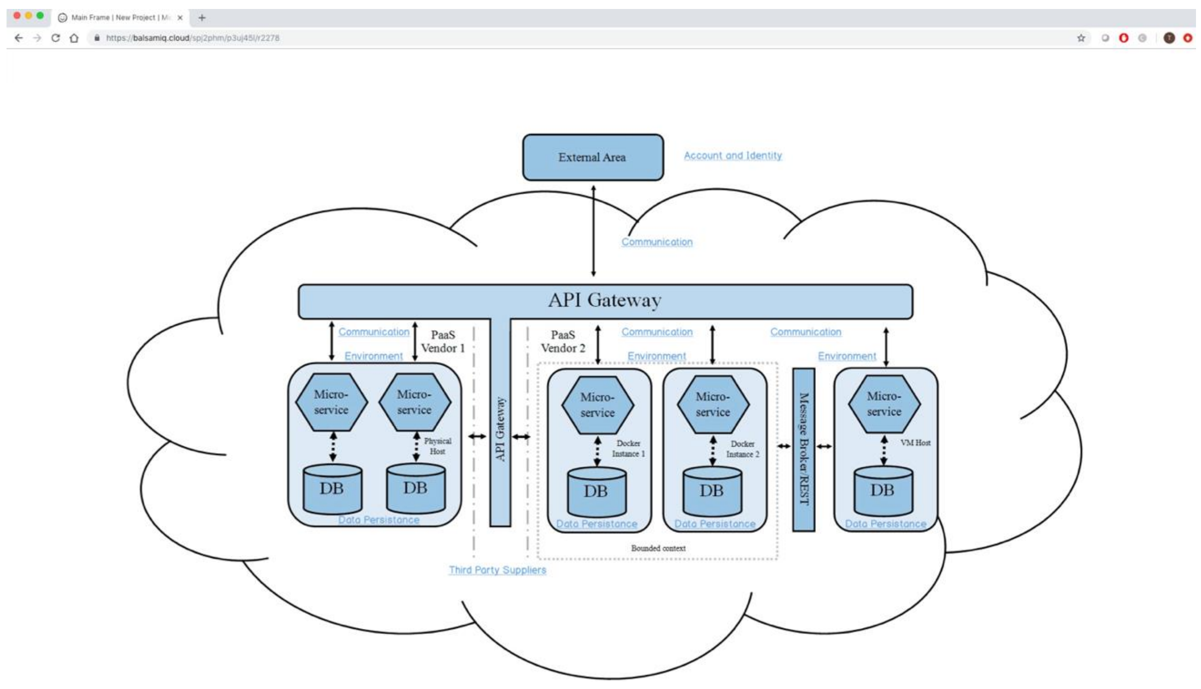
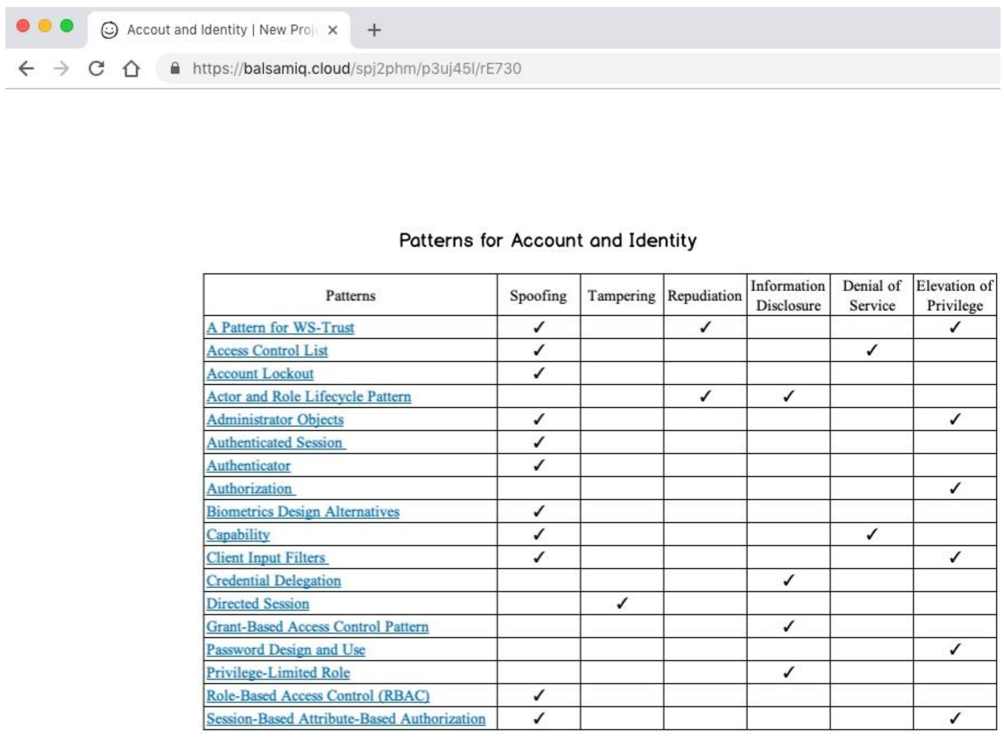


Figure 15 Scheme of a software application applying microservice architecture

Clicking on each of the categories will navigate to a second page that shows all the security patterns associated with their respective category. Figure 16 shows an example after clicking on the hyperlink *Account and Identity*.

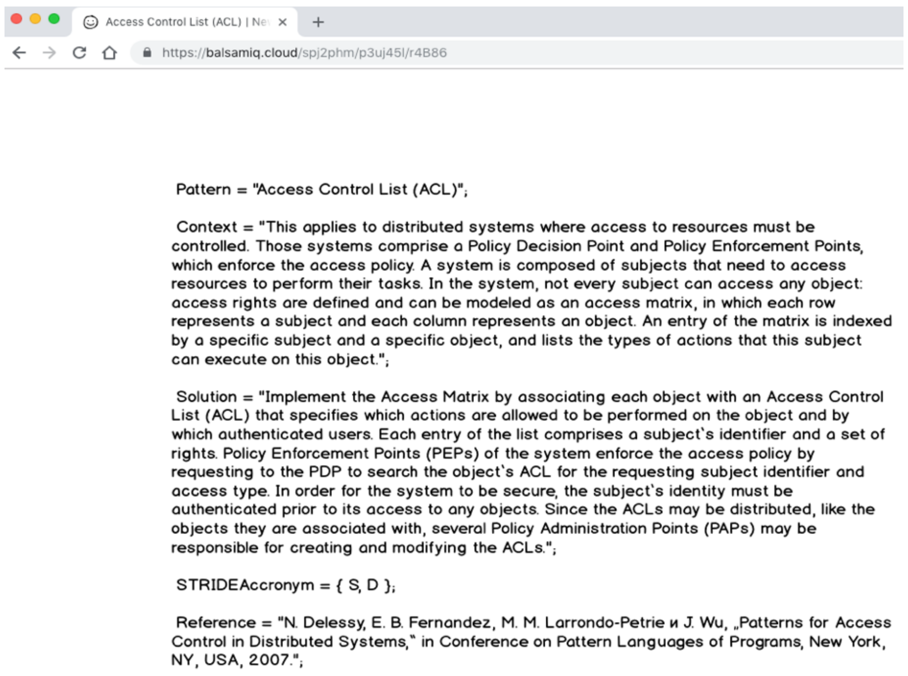

Account and Identity | New Proj
https://balsamiq.cloud/spj2phm/p3uj45l/rE730

Patterns for Account and Identity

Patterns	Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevation of Privilege
A Pattern for WS-Trust	✓		✓			✓
Access Control List	✓				✓	
Account Lockout	✓					
Actor and Role Lifecycle Pattern			✓	✓		
Administrator Objects	✓					✓
Authenticated Session	✓					
Authenticator	✓					
Authorization						✓
Biometrics Design Alternatives	✓					
Capability	✓				✓	
Client Input Filters	✓					✓
Credential Delegation				✓		
Directed Session		✓				
Grant-Based Access Control Pattern				✓		
Password Design and Use						✓
Privilege-Limited Role				✓		
Role-Based Access Control (RBAC)	✓					
Session-Based Attribute-Based Authorization	✓					✓

Figure 16 List of security patterns presented graphically

Each security pattern has a hyperlink to information that is stored in the CIM classes. Figure 17 shows an example of Access Control List (ACL) security pattern.


Access Control List (ACL) | Ne
https://balsamiq.cloud/spj2phm/p3uj45l/r4B86

Pattern = "Access Control List (ACL)";

Context = "This applies to distributed systems where access to resources must be controlled. Those systems comprise a Policy Decision Point and Policy Enforcement Points, which enforce the access policy. A system is composed of subjects that need to access resources to perform their tasks. In the system, not every subject can access any object: access rights are defined and can be modeled as an access matrix, in which each row represents a subject and each column represents an object. An entry of the matrix is indexed by a specific subject and a specific object, and lists the types of actions that this subject can execute on this object.";

Solution = "Implement the Access Matrix by associating each object with an Access Control List (ACL) that specifies which actions are allowed to be performed on the object and by which authenticated users. Each entry of the list comprises a subject`s identifier and a set of rights. Policy Enforcement Points (PEPs) of the system enforce the access policy by requesting to the PDP to search the object`s ACL for the requesting subject identifier and access type. In order for the system to be secure, the subject`s identity must be authenticated prior to its access to any objects. Since the ACLs may be distributed, like the objects they are associated with, several Policy Administration Points (PAPs) may be responsible for creating and modifying the ACLs.";

STRIDEAccronym = { S, D };

Reference = "N. Delessy, E. B. Fernandez, M. M. Larrondo-Petrie и J. Wu, „Patterns for Access Control in Distributed Systems," in Conference on Pattern Languages of Programs, New York, NY, USA, 2007.";

Figure 17 Graphic representation of the Access Control List (ACL) security pattern

7.4 Analysis of the results achieved

The benefits of using CIM object-oriented modeling are the ability to structure the data in order to describes each of the security patterns, to make associations among security patterns, and to build cross-domain relationships. These three aspects form a hierarchical taxonomy of security patterns for enhancing security in software applications based on a microservice architecture.

Conclusion

The methodology chosen to describe the hierarchical taxonomy of security patterns relies on the rules declared in CIM (Common Information Model).

The chosen language for describing all elements of the CIM methodology is determined to comply with the conditions set in Managed Object Format (MOF).

The elements used to form the entire hierarchical taxonomy of security patterns are: Scheme, Class, Properties, Links and Associations, Qualifier.

The hierarchy of microservice architecture areas is graphically provided. All the links between the different areas are described.

Shown are some examples in the form of "Managed Object Format (MOF)", representing the various elements: Scheme, Class, Properties, Links and Associations, Qualifier.

The result of creating a hierarchy of microservice architecture areas is graphically represented using UI Wireframing.

An analysis was made of the results achieved.

8 VIII CHAPTER. APPLICATION APPIES SECURITY PATTERNS USING NEW TECHNOLOGIES FOR MICROSERVES MANAGEMENT

The following points discuss advanced microservice management tools that can be configured following the best practices of selected security patterns from the Hierarchical Taxonomy of Security Patterns described in Chapter 7.

8.1 Development of a platform for microservice application using modern technologies

The platform is made up of open source components, most of them modified according to certain requirements, while others have been designed for the purpose of the microservice platform.

Chapter 2 looks at the "Account and Identity" area. It can be controlled by various tools that provide API Gateway functionality. The product used in the current conceptual model is Kong [9]. Chapter 3 covers the area of Communication. It is divided into two parts: " Inter-connection" and "Messages". The tool that can be used for communication between microservices is RabbitMQ [10]. Chapter 4 looks at the Data Storage area. It is basically addressed to one of the microservice architecture limitations - any microservice can operate with information to which only it has access. When gathering information stored by another microservice, it must contact the microservice that owns this information in order to avoid direct access. Chapter 5 looks at the Microservice Environment area. It refers to the environment in which microservices can resides. Products that offer such functionality are: Docker container [14], CentOS [13], and IaaS [15]. Each environment has a set of tools that can contribute to security enhancing. Chapter 6 looks at the area "Microservices deployed on multiple vendor platforms". It discusses the possibility of separating the business logic of a microservice software application and placing the individual elements on two or more service providers. This type of clustering can be managed using Kubernetes (K8s) [16].

Figure 18 presents an architecture for a microservice application management platform that presents all five areas discussed in the previous chapters. All the tools that provide the successful implementation of most of the security patterns described in the hierarchical taxonomy of security patterns are shown.

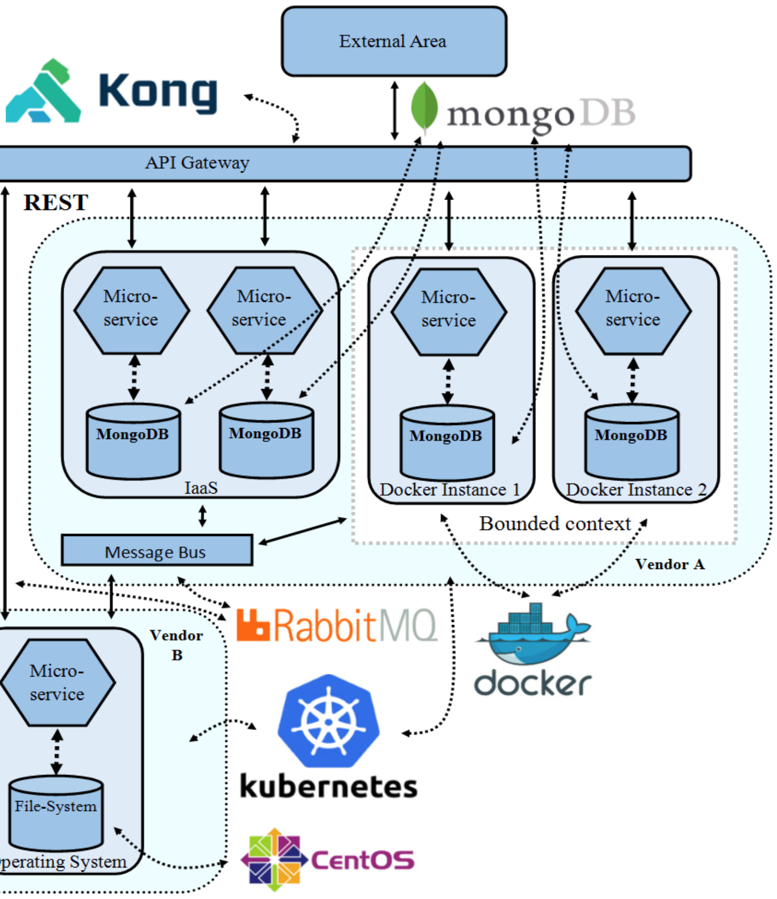


Figure 18 Architecture of a Microservice Application that includes the covered threaten areas

8.2 Implementation of patterns for a reliable platform built by modern business solutions

The following points discuss examples of applying security patterns presented in the hierarchical taxonomy of security models in Chapter 7 with the help of modern business technologies. Each of the technologies is discussed in the previous point 8.1.

8.2.1 Account and Identity

The business application considered here is Kong [9]. It should be noted that if you consider applying Kong, you must have it installed on all hosts that microservices reside. A list of all Account and Identity security patterns is provided on Figure 3.

8.2.2 Communication

The point discusses examples of applying security patterns represented in the hierarchical taxonomy of Security Patterns (Chapter 7) using modern business technologies. The tool that can be used for communication between microservices is RabbitMQ [10]. The lists of all Security patterns are presented at Figure 5 and Figure 6.

8.2.3 Data Persistence

The point discusses examples of applying security patterns represented in the hierarchical taxonomy of security patterns (Chapter 7) using modern business technologies. The tools that could be used when storing data are: MongoDB database [11] as well as Filesystem ext4 [12], which is part of Operation System CentOS [13]. The list of all Security Patterns that take part of "Data Persistence" is shown in Figure 8.

8.2.4 Microservice Environment

The point discusses examples of applying security patterns represented in the hierarchical taxonomy of security patterns (Chapter 7) using modern business technologies. For current purposes, tools such as: Docker container [14] and CentOS [13] Operation System have been used. A list of all Security Patterns for Microsystem Environment is presented on Figure 10.

8.2.5 Microservices distributed across platforms from different vendors

The points discusses examples of applying security patterns represented in the hierarchical taxonomy of security patterns (Chapter 7) using modern business technologies. The Kubernetes [25] tool has been explored. It is mainly used for managing clusters consisting of Docker instances. A list of all security patterns from Microservices Distributed to Multiple Vendors Platforms is presented on Figure 13.

Conclusion

A study has been made about implementation of modern technologies. It helps in constructing of architecture of a platform for managing microservice applications. It aims to cover all areas defined in Chapters 2, 3, 4, 5 and 6.

Sample test scenarios and solutions are presented for almost any of the selected security patterns.

Solutions to selected security patterns in the Account and Identity area are presented using the Kong product and its core API Gateway functionality.

Solutions to selected security patterns in the Communication area using RabbitMQ are presented. It gives autonomy for individual microservices by providing a channel to which microservices send and receive messages.

Solutions for selected security patterns in the Data Persistence area are provided using MongoDB database and the ext4 file system provided by the CentOS operating system.

Solutions are provided for selected security patterns in the Microservices environment using a range of products: Docker container, CentOS and IaaS.

Solutions have been provided regarding selected security patterns for area: Microservices deployed on multiple vendor platforms. The product that consist of tools for clustering a microservice application is Kubernetes (K8s).

9 CONCLUSION

The present thesis proposes a solution for enhancing security in information systems based on Microservice Architecture.

The developed hierarchical taxonomy of security patterns provides an easy and convenient way to find the right security pattern or set of security patterns for certain cases.

The taxonomy, represented by an object-oriented modeling language, gives the opportunity for including new security patterns.

The developed language allows a graphical representation of the various relationships as well as security patterns using modern graphical interfaces.

Modern technologies that can be used in the construction and management of a complete Microservice software application are discussed.

A platform architecture for managing microservice applications is offered, as well as solutions for implementing almost all security patterns described in the hierarchical taxonomy of security patterns using modern business tools.

10 CONTRIBUTIONS

- Research and analysis of architectures based on microservices is made for security purposes.
- A conceptual model using microservice architecture has been proposed, which helps in defining the vulnerable areas.
- Threat analysis was performed toward the defined Microservice areas. Lists of appropriate Security Patterns as well as adapted solutions are proposed.
- A hierarchical model is presented, and a hierarchical taxonomy of security patterns is developed using object-oriented modeling.
- A graphical interface has been made to illustrate the links between vulnerable areas in microservice architecture and selected security patterns.
- The architecture of a platform implementing the proposed patterns is presented using modern microservice management technologies.

11 REFERENCES

[1] C. Richardson and F. Smith, MICROSERVICES From Design to Deployment, NGINX, 2016.

[2] M. Weiss and H. Mouratidis, “Selecting Security Patterns that Fulfill Security Requirements,” *Proceedings of the 16th IEEE International Conference on Requirements Engineering,* pp. 169-172, 2008.

[3] E. Gamma, J. Vlissides, R. Johnson and R. Helm, Design Patterns: Elements of Reusable Object-Oriented Software, 1994.

[4] “Understanding software design patterns,” [Online]. Available: https://opensource.com/article/19/7/understanding-software-design-patterns.

[5] “Design Pattern - Overview,” [Online]. Available: https://www.tutorialspoint.com/design_pattern/design_pattern_overview.htm.

[6] A. Shostack, THREAT MODELING: Designing for Security 1st Edition, John Wiley & Sons, Inc., 2014.

[7] “DMTF Releases CIM 2.52,” [Online]. Available: https://www.dmtf.org/content/dmtf-releases-cim-252.

[8] “Balsamiq Cloud,” [Online]. Available: https://balsamiq.cloud/.

[9] “Kong Gateway,” [Online]. Available: https://konghq.com/kong.

[10] “RabbitMQ,” [Online]. Available: https://www.rabbitmq.com/.

[11] “MongoDB,” [Online]. Available: https://www.mongodb.com/.

[12] “An introduction to Linux's EXT4 filesystem,” [Online]. Available: https://opensource.com/article/17/5/introduction-ext4-filesystem.

[13] “CentOS Project,” [Online]. Available: https://www.centos.org/.

[14] “Docker Homepage,” [Online]. Available: https://www.docker.com/.

[15] E. B. Fernandez, H. Washizaki and N. Yoshioka, “Patterns for Secure Cloud IaaS (Infrastructure as a Service),” in *Asian Pattern Languages of Programs (PLoP) Conference,* 2016.

[16] “Kubernetes (K8s),” [Online]. Available: https://kubernetes.io/.

[17] T. Tenev and D. Birov, “Security Patterns for Microservice Account and Identity,” in *15th International Conference on Informatics and Information Technologies,* Mavrovo, 2018.

[18] T. Tenev and D. Birov, “Security Patterns for Microservice Communication,” in *Четиридесет и седма пролетна конференция на Съюза на математиците в България,* Borovets, 2018.

[19] T. Tenev, “SECURITY PATTERNS FOR MICROSERVICE DATA MANAGEMENT,” in *Doctoral Conference: Young Scientists,* Sofia, 2018.

[20] C. Posta, Microservices for Java Developers, 2016.

[21] T. Tenev and S. Tsvetanov, “Enhancing security in Microservice environments,” in *9th Balkan Conference in Informatics,* Sofia, 2019.

[22] T. Tenev and D. Birov, “SECURITY PATTERNS FOR MICROSERVICES LOCATED ON DIFFERENT VENDORS,” in *VII International Conference on Engineering, Technologies and Systems TECHSYS*, Plovdiv, 2018.

[23] [Online]. Available: https://www.dmtf.org/sites/default/files/standards/documents/DSP0221_3.0.0.pdf.

[24] P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” 2011. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-145/final.

[25] “Kubernetes,” [Online]. Available: https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/.

PUBLICATIONS RELATED TO THESIS

1. Tihomir Tenev, Dimitar Birov, Security Patterns for Microservice Account and Identity, In proceedings of 15th International Conference on Informatics and Information Technologies, 2018, pages:124-128, ISBN:978-608-4699-08-8,

2. Tihomir Tenev, Dimitar Birov, Security Patterns for Microservice Communication, Доклади на Четиридесет и седма пролетна конференция на Съюза на математиците в България, 2018, ISSN (online):1313-3330

3. Tihomir Tenev, SECURITY PATTERNS FOR MICROSERVICE DATA MANAGEMENT, In proceedings of Doctoral Conference: Young Scientists, 2018, pages:575-581, ISBN:978-954-07-4611-1

4. Tihomir Tenev, Dimitar Birov, SECURITY PATTERNS FOR MICROSERVICES LOCATED ON DIFFERENT VENDORS, VII International Conference on Engineering, Technologies and Systems TECHSYS 2018, Technical University – Sofia, Plovdiv, 2018, pages:130-133, ISSN (online):2535-0048

5. Tihomir Tenev, Simeon Tsvetanov, Enhancing security in Microservice environments, 9th Balkan Conference in Informatics, ISec2019 Workshop, 2019