ENHANCING E-HEALTH CYBERSECURITY AND RESILIENCE: SHIFTING FROM MONOLITHIC TO MICROSERVICES ARCHITECTURE

Cristian CONTAȘEL[1], Răzvan RUGHINIȘ [2], Dumitru Cristian TRANCĂ [3], Dinu ȚURCANU [4]

This study provides an innovative architectural model for e-Health systems that aims to improve cyber resilience while maintaining high availability under fluctuating traffic loads. We examined typical cybersecurity incidents in the field of e-Health, their correlations with architectural defects, and frequent design patterns in currently operational systems. A testing approach based on our research finds those weaknesses and proposes viable fixes. In this paper, a comprehensive support strategy for transitioning from conventional monolithic architectures to microservices is presented. This change makes use of cloud computing's vertical and horizontal scalability to maximize resource utilization while ensuring system reliability. We also discuss deployment ideas for the new microservices, focusing on operational resilience and cybersecurity in e-Health environments.

Keywords: e-Health, Cloud Computing, Microservices Architecture, Monolithic Architecture, Microservices Deployment, Scaling

1. Introduction

Nowadays, e-Health software systems are becoming more frequent in our lives. We use these systems whether we are simple patients, doctors, or collaborators with a medical institution. By automating various flows and procedures, these systems improve medical safety, reduce human error, and save money.

Given their rapid adoption, e-health systems require robust cybersecurity to protect against DDoS and ransomware attacks. These incidents can disrupt healthcare, compromise patient care, and compromise data privacy. To address these issues, we propose that e-Health systems evolve from monolithic to

[1] As., Dept. of Computer Science and Engineering, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: cristian@hanzu.ro
[2] Prof., Dept. of Computer Science and Engineering, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: razvan.rughinis@upb.ro
[3] Lecturer, Dept. of Computer Science and Engineering, National University of Science and Technology POLITEHNICA Bucharest, e-mail: dumitru.tranca@upb.ro
[4] Assoc. Prof., Dep. of Software Engineering and Automatics, Technical University of Moldova, Chișinău, Moldova, e-mail: dinu.turcanu@adm.utm.md

microservices architectures. This architectural transformation isolates breaches to particular services and improves scalability to handle sudden demand spikes, which occur frequently during health crises like the COVID-19 pandemic. Our article will explain in detail how microservices can improve healthcare IT infrastructure cybersecurity and operational resilience.

In the past, e-Health software had a slow adoption rate because of strict requirements and legal regulations that required extensive research and testing before it could be made available. In 2018, the proportion of patients who used medical software services was nearly zero [1]. According to Business Research Company's 2024 forecast [2], the medical software market is expected to grow at a rate of 14,6% per year until 2028.

The COVID-19 pandemic accelerated the adoption of these systems by limiting contact between people and forcing both patients and medical units to adopt new e-Health systems for patient care. Because of COVID-19, the number of patients who began using medical software services increased to 13% of all patients [3]. Unfortunately, this increase in adoption highlighted the limitations of e-Health systems, as many of them became overwhelmed by the volume of traffic they had to handle or were unable to accommodate new features.

To determine the traffic variation for e-Health systems, a traffic analysis was conducted using CO APCD public data from April 2018 to March 2024. The results are shown in Fig. 1 and Fig. 2.

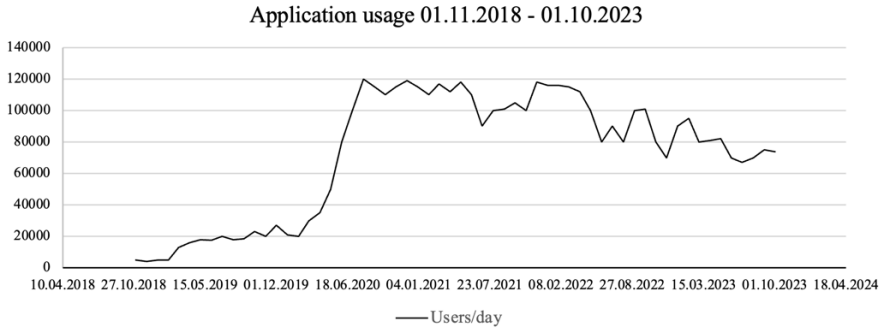

Fig. 1. Application usage between 01.11.2018 and 01.10.2023, based on the number of users/day

As described in Fig. 1, before COVID-19, the e-Health system had a small number of users per day (less than 20,000), but after the pandemic, the number increased to 121,000 users per day.

After all of the restrictions were lifted, the number of users decreased slightly, by approx. 9% per month. This decrease was caused by urban people in general, while in rural areas, usage did not change significantly between 2021 and 2023, as can be distinguished in Fig. 2.

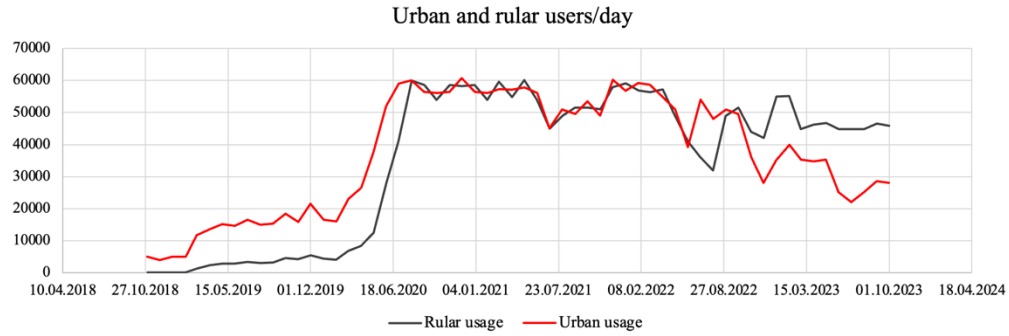

Fig. 2. Application usage between 01.11.2018 and 01.10.2023, divided by area type.

The same trend is confirmed by KFF and Epic Research's [4] analysis of the share of outpatient visits by telehealth by area type, which is presented in Fig. 3.

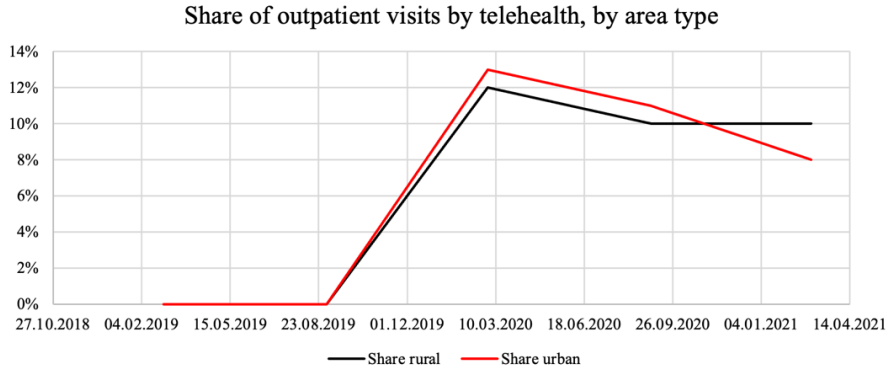

Fig. 3. KFF and Epic Research analysis of share of outpatient visits by telehealth by area type

The current challenge in medical software systems is how to mitigate cybersecurity threats and risks, and also how to handle the performance issues caused by the significantly increased workload as a consequence of their rapid adoption.

This article propose a new microservice e-Health system to effectively manage fluctuations in traffic and reduce costs by incorporating both vertical and horizontal scaling methods.

2. Related Work

One of the global impacts of the coronavirus was in the healthcare industry. COVID-19 caused overcrowding in the hospitals, which makes it impossible for patients and doctors to meet in person for a consultation. e-Health systems, in addition to the roles for which they were developed, also took on a new role as a mediator in order to respond to this situation. Additionally, new

features were either developed or used more frequently in order to meet this demand: contact tracing, telehealth (online consultation with a doctor), automated diagnosis, forecasting of material resource requirements, and individual medical record about the COVID-19 illness.

The NIST cybersecurity framework is one of the lightweight models for addressing new threats and risks present in e-Health systems. The NIST cybersecurity framework (CSF) consists of the following stages: identify, protect, detect, respond, and recover the system.

The most common COVID-19-related attacks on e-Health systems are ZOOM bombing, COVID-19 phishing attacks, malware, and network availability [5].

In this paper, we discuss network availability issues in e-Health systems. In order to do so, we extract a common architectural model based on a study that was carried out on medical units. This study identifies the most commonly used e-Health software systems, as well as the key features required. The study included 45 hospitals and medical clinics in Bucharest, in the public and private sectors.

The main features requested by industry from an e-Health system are: electronic health record, medical diagnosis, e-prescribing, telemedicine, medical database, medical imaging; medical laboratory, and clinic management.

All of these features enable effective collaboration among patients, doctors, and clinics, making any disease easier to control or cure.

Based on the architecture of e-Health software systems, 17 out of 45 entities use only local software systems, while 28 use web applications. The most common architecture followed the model-view-controller monolith pattern (MVC).

In software engineering, a monolithic application refers to an application that is designed as a single service [6]. This approach generates some advantages in terms of cost reduction [7]: easy to deploy, easy to debug, faster end-to-end testing, increased performance, and one code-base.

Because of all those advantages that generate a rapid development and testing, the monolithic architecture is present in a lot of e-Health software systems.

To create a common overview of the analyzed e-Health software systems, we identify the core and optional modules that can be combined to create a system that includes all of the previously described features.

The monolithic e-Health architecture pattern that contains all the modules in order to support the core and optional features is described in Fig. 6.

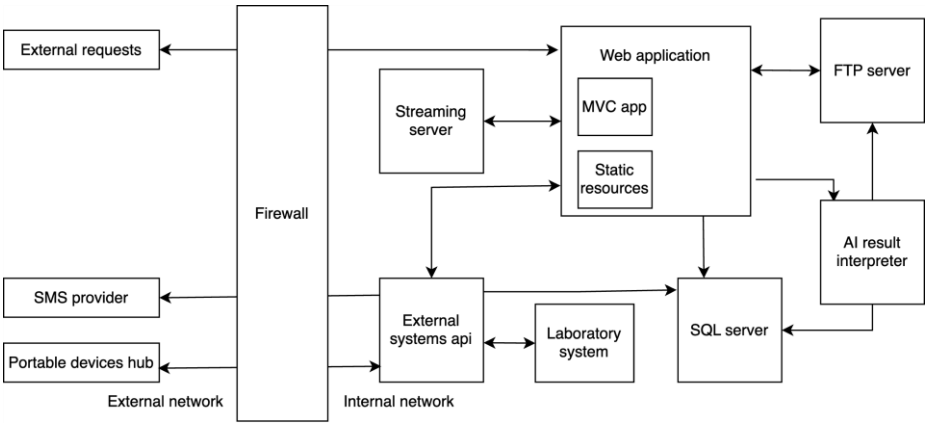

Fig. 6. Monolithic e-Health architecture software pattern

3. Monolithic e-Health architecture software performance

To identify the main issues in the system, the following types of performance tests were carried out: stress testing, endurance testing, and spike testing.

Stress testing is a performance test that provides an overview of system capacity limits and determines the architecture's robustness [8].

The endurance test is used to determine how long the system can operate under continuous load [9]. In general, memory usage is monitored in order to detect memory leaks.

Spike testing is performed to identify the system's behavior when the number of users or their actions unexpectedly increases, in order to determine what actions are required to handle dramatic load changes.

In e-Health software, the main concerns about the number of users come from the patient role. Because of this issue, the designed scenarios should be based on the main action that patients can perform. The testing scenarios are described in table 1.

Table 1

Defined test scenarios

Test nr.	1	2	3
Goal	Retrieve appointment results	Schedule telemedicine appointment	Retrieve laboratory result
Involved components	Firewall, Web application, FTP server, SQL server	Firewall, Web application, SQL server, Streaming server	Firewall, Web application, SQL server, FTP server, Laboratory system

Apache JMeter was used to perform automated testing. Apache JMeter is a widely used open-source framework designed for conducting performance testing [1]. The instances of JMeter was hosted on Azure by using Azure Cloud Service.

To determine the maximum number of threads supported by JMeter in Azure Cloud Service, a progressive increase in the number of threads was performed from 100 to 2000, with 10 threads added each step.

The upper limit for JMeter determined during the test was 1000 threads per instance; after this limit, JMeter's performance was degraded. The results of the stress testing are presented in Table 2.

Table 2

Stress testing results

Number of threads (users)	Test 1		Test 2		Test 3	
	Error (%)	Success (%)	Error (%)	Success (%)	Error (%)	Success (%)
150.000	0.00	100	0.00	100	0.00	100
160.000	0.00	100	1.3	98.7	0.2	99.8
170.000	0.4	99.6	4.7	95.3	1.2	98.8
200.000	14.57	85.43	23.5	76.5	45.2	54.8
250.000	35.67	64.33	48.16	51.84	72.8	27.2

The error % in table 2 denotes the proportion of requests with errors among the total number of requests made by JMeter during the test. We consider a request to have an error status if it failed or returned an error code (HTTP Status Codes class 400 or 500).

The success% represents the percentage of successful requests among all requests made by JMeter during the test execution. We consider a request to be successful if its HTTP Status Codes class is 200 or 300.

The number of users was limited during the tests to 250.000 due to the cost of resources required to perform the stress test, but, as shown in table 2, the system performance was damaged very strongly after the number of 170.000.

4. Ensuring monolithic e-Health system stability

To guarantee the e-Health system's availability and prevent system failure, a series of actions can be implemented at a reduced cost in accordance with the results of the testing scenario.

How the main concerns about the number of users is derived from the patient role, it is possible to restrict the access for that category of users based on

a queue system, in order to limit the active user to a maximum number of 150.000 or lower, depending on how many doctors should be accommodated within the system. The average waiting time in the queue was 19 minutes. The time was determined by averaging the queue's waiting times over a seven-day period.

To make that distinction, the system can use different endpoints to enable doctors to use the system. That endpoint can be restricted by role or IP address to ensure that it is only accessible from hospitals and medical clinics. The Spring Security module was used to enforce the restrictions.

For monolithic architectures, the only available scaling is vertical, which is limited by hardware constraints.

5. Proposed microservices cloud architecture for e-Health systems

In order to effectively manage load differences and maintain a lower infrastructure cost, the primary characteristic of the new architecture is its vertical and horizontal scalability [10].

The proposed system architecture is based on microservices, which allow the original monolithic application to be split into multiple independent services capable of performing work independently. This independence enables the booth scaling system to be implemented. We will be able to initiate new workers for each service in accordance with the system load. Additionally, we will be able to enhance the computational power of current workers.

The splitting of the microservices was done based on the functionality of the system to be able to provide the features to users independently of each other, so that if a set of microservices no longer works properly, the system can manage the rest of the features independently.

In order to accomplish this, we divide the services into Level 1 and Level 2 services. Level 1 services are mapped to various system features in order to provide system functionality. They have a caching and optimization of the request mechanism in place, and they are also capable of storing data in the SQL module.

The level 2 services are the ones that provide support for the level 1 services and are capable of integrating with various subsystems that are not scalable, such as external providers or outdated applications (e.g., laboratory systems).

Messaging and queues are used to facilitate communication between services. All of this logic is abstracted in the Message broker.

This separation of services between Layer 1 and Layer 2 is also necessary to facilitate the seamless transition of the application from a monolithic architecture to a decoupled architecture. The main goal of this progressive migration is to use the power of microservices even if the original architecture has not been fully migrated.

The transition from a monolithic to a microservice architecture is presented in Fig. 7.

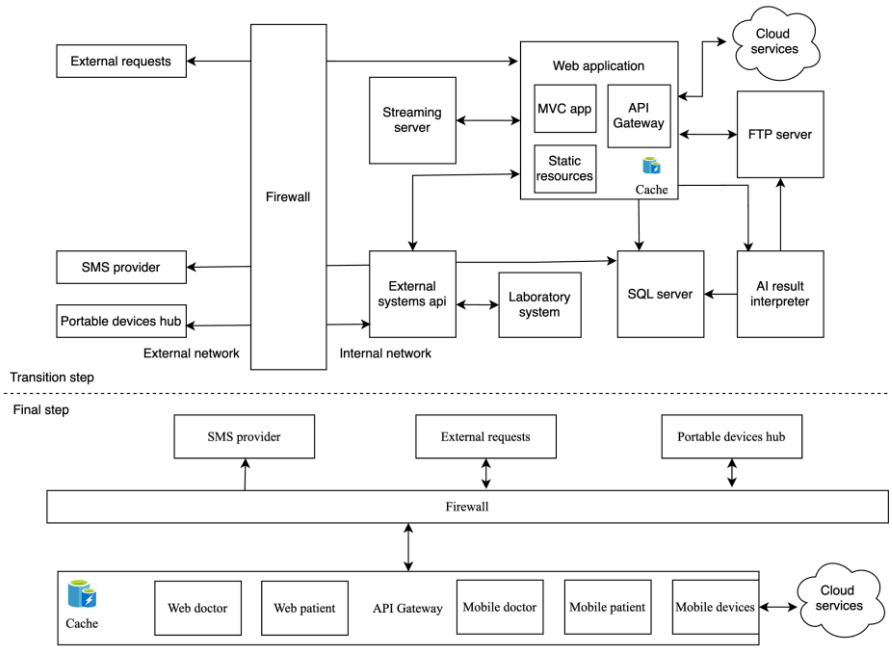

Fig. 7. e-Health system architecture transition.

The transition step involves the addition of cloud services one by one, and the new API gateway module ensures seamless connectivity with these services. In order to optimize performance, a caching system was implemented.

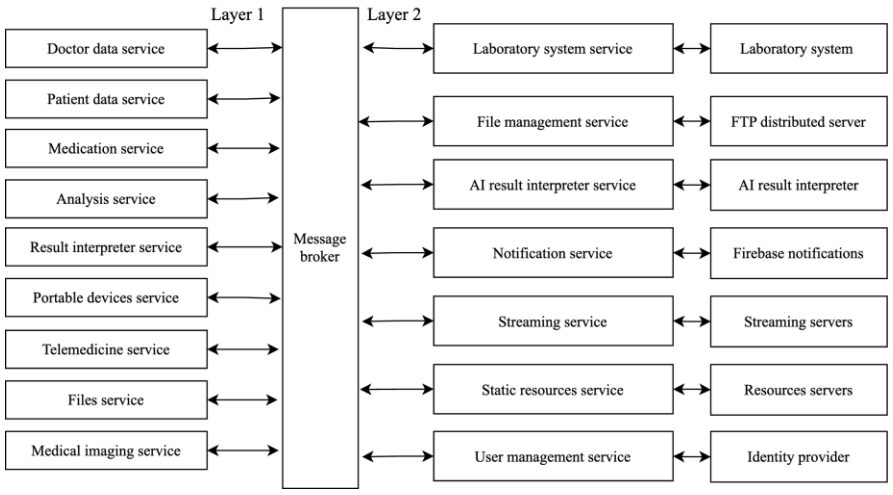

Fig. 8. Microservices of the resulting e-Health system

Fig. 8 presents the entire set of microservices integrated into the new architecture. The microservices architecture that results is composed of 16 services, which are categorized into Layer 1 and Layer 2 levels.

6. Microservices e-Health systems deployment

Based on the analysis made by Tabish Mufti, Pooja Mittal and Bulbul Gupta in the paper "A Review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud Platform (GCP) Services" [11], Microsoft Azure was chosen as the cloud provider due to lower service costs when compared to AWS and GCP, as well as the availability of additional options such as machine learning, analytics services, and AI support.

The main options for deploying microservices in Microsoft Azure are Cloud Services and Azure Web Apps. To determine which solution was used, the effective cost for 24 hours was used. The cost of booth solutions is shown in table 3 and table 4, and it was calculated using West Europe region datacenters with an Azure Hybrid Benefit Windows license and a one-year savings plan.

Table 3

Deployment Scenario cost for Azure App Service

Microservice	Instance nr.	Tier	Tier 24h cost ($)	Total 24h Cost ($)
Doctor data service	2	S1: 1 Cores, 1.75 GB RAM, 50 GB	2.40	4.8
Patient data service	2	S1: 1 Cores, 1.75 GB RAM, 50 GB	2.40	4.8
Medication service	2	S1: 1 Cores, 1.75 GB RAM, 50 GB	2.40	4.8
Analysis service	2	S2: 2 Cores, 3.5 GB RAM, 50 GB	4.80	9.6
Result interpreter service	2	S2: 2 Cores, 3.5 GB RAM, 50 GB	4.80	9.6
Portable devices service	2	S3: 4 Cores, 7 GB RAM, 50 GB	9.60	19.2
Telemedicine service	3	S2: 2 Cores, 3.5 GB RAM, 50 GB	4.80	14.4
Files service	2	S3: 4 Cores, 7 GB RAM, 50 GB	9.60	19.2
Medical imaging service	2	S3: 4 Cores, 7 GB RAM, 50 GB	9.60	19.2
Laboratory system service	1	S3: 4 Cores, 7 GB RAM, 50 GB	9.60	9.60
File management service	1	S2: 2 Cores, 3.5 GB RAM, 50 GB	4.80	4.80

AI result interpreter service	1	S2: 2 Cores, 3.5 GB RAM, 50 GB	4.80	4.80
Notification service	1	S1: 1 Cores, 1.75 GB RAM, 50 GB	2.40	2.40
Streaming service	2	S2: 2 Cores, 3.5 GB RAM, 50 GB	4.80	9.6
Static resources service	1	S2: 2 Cores, 3.5 GB RAM, 50 GB	4.80	4.80
User management service	1	S3: 4 Cores, 7 GB RAM, 50 GB	9.60	9.60

Table 4

Deployment Scenario cost for Azure Cloud Service

Microservice	Instance nr.	Tier	Tier 24h cost ($)	Total 24h Cost ($)
Doctor data service Patient data service	2	D3: 4 vCPUs, 14 GB RAM, 200 GB	14.23	28.46
Medication service Analysis service Medication service Result interpreter service AI result interpreter service Medical imaging service	2	D4: 8 vCPUs, 28 GB RAM, 400 GB	28.49	56.98
Portable devices service Notification service	2	D3: 4 vCPUs, 14 GB RAM, 200 GB	14.23	28.46
Telemedicine service Streaming service	3	D13: 8 vCPUs, 56 GB RAM, 400 GB	29.81	89.43
User management service Static resources service File management service Laboratory system service Files service	1	D4: 8 vCPUs, 28 GB RAM, 400 GB	28.49	28.49

According to tables 3 and 4, the estimated cost of Azure Cloud Service deployment is $231.82, while Azure App Service deployment is estimated at $151.2. However, the decision to deploy the microservices into one of these solutions should also consider the computational power required to handle the same number of requests.

The number of instances and their capabilities were selected based on the temporal and spatial complexity of microservices, with the goal of ensuring an average time of less than 0.1 ms for messages in the message broker.

7. Microservices e-Health systems performance

In order to measure the performance differences between the original architecture and the microservices architecture, the exact same set of tests (described in Table 1) was executed. However, the tests were executed on both the Azure Cloud Service and Azure App Service deployment scenarios.

How the original architecture was able to handle the tests for 170,000 users, now the test scenarios begin at 200.000 and are run in Azure Cloud to provide adequate power.

Table 5

The result of stress testing using Azure App Service with automatic horizontal scaling.

Number of threads (users)	Test 1		Test 2		Test 3	
	Error (%)	Success (%)	Error (%)	Success (%)	Error (%)	Success (%)
200.000	0.00	100	0.00	100	0.00	100
250.000	0.2	99.8	0.1	99.9	0.00	100
300.000	0.7	99.3	0.4	99.6	0.3	99.7
350.000	1.25	98.75	1.45	98.55	1.2	98.8
400.000	1.3	98.7	1.7	98.3	2	28.0

Table 6

The result of stress testing using Azure Cloud Service with horizontal scale mechanism.

Number of threads (users)	Test 1		Test 2		Test 3	
	Error (%)	Success (%)	Error (%)	Success (%)	Error (%)	Success (%)
200.000	0.00	100	0.00	100	0.00	100
250.000	0.00	100	0.00	100	0.00	100
300.000	0.9	99.1	0.5	99.5	0.6	99.4
350.000	1.1	98.9	1.3	98.7	1.5	98.5
400.000	1.0	99.0	1.25	98.75	2.1	97.9

According to the stress tests, presented in table 5 and table 6, the system's performance has substantially improved. However, the addition of new machines to the system, which is caused by horizontal scaling, results in some failed requests.

8. Conclusions

The e-Health software system market is a dynamic market that is constantly evolving in response to technological advancements and the introduction of new devices. .

The current software generally uses a monolithic architecture design, which is enforced by the presence of legacy technologies and standards, due to the fact that e-Health software systems have been used in a relatively narrow and closed market for long time.

Unfortunately, the monolithic architecture can no longer cope with market demands and fluctuations in order to meet actual cyber resilience requirements and medical projects exceed the entry-level constraints.

In order to mitigate the risk that impacts network availability and guarantee the high availability of e-Health software systems, this paper suggests the implementation of cloud computing and microservices as a solution. The main goal of the transition from monolithic architecture to microservices architecture in e-Health software systems is to establish a new layer between legacy software and the new expectations and behaviors of users. Additionally, it intends to provide support and scaling to accommodate a high volume of concurrent requests.

This requirement was highlighted and enforced during the COVID-19 cyber resilience in e-Health system. During the pandemic period, e-Health systems had to deal with major changes in user behavior, which highlighted the need for decoupling performance between different software modules.

To make that performance decoupling possible, this paper proposes a transition to microservices that allow vertical and horizontal scaling. In order to minimize hardware overhead, the proposal is to utilize cloud solutions to host all components of the e-Health system.

The performance of a microservices e-Health system hosted in the cloud remains the same regardless of the type of PASS used, as demonstrated in this paper. The solution of Azure Cloud Services offers greater control, while the solution of Azure VMs or Azure App Service offers greater abstraction.

The transition from a monolithic to a microservices architecture enhances e-health cybersecurity. Single points of failure can make a monolithic system vulnerable to DDoS attacks, which overwhelm it with traffic. The microservices architecture distributes load across smaller, scalable services. A monolithic system cannot absorb and mitigate increased traffic like this distribution.

Additionally, zero-day exploits target unknown software vulnerabilities. A single vulnerability in a monolithic system has the potential to compromise the entire system. The exploit does not spread across microservices because it only affects the compromised service. Each microservice can be isolated, patched, and redeployed independently, improving system security.

Monolithic architectures make security updates difficult and risky, requiring downtime. Service-by-service updates are easier with microservices. This speeds patch implementation and reduces downtime, improving system security.

Furthermore, the microservices architecture isolates databases and other resources by separating functions into services. This isolation helps to limit data breaches to a single service rather than the whole system. Unlike monolithic systems, microservices can use security protocols that are appropriate for their needs. This flexibility maximizes security based on service sensitivity and needs.

Moving from monolithic to microservices architecture in e-health systems improves resilience in several ways. Microservices reduce system downtime because minor failures do not impact the entire system. Only the broken microservice needs to be patched; the rest is operational. This minimizes downtime, which is critical for 24/7 healthcare services. Moreover, microservices scale up or down independently based on demand, improving traffic spike response. Microservices can handle high user loads, such as during a health crisis, by adding resources to busy services without affecting less busy ones.

Because microservices are separate, updates and bug fixes can be applied to individual services without crashing the system. This lets security patches and new features be released quickly and securely, keeping the system secure. In addition, microservices architecture separates services, so a security breach in one doesn't affect others. Containment reduces data breaches.

Each microservice can have customized security measures. This enables more precise security tailored to each service's data or transactions.

Thus, the proposed microservices architecture improves e-health systems' cyber-resilience by addressing these vulnerabilities, making them better prepared for current and emerging cybersecurity threats. Microservices architecture also makes e-health systems more flexible, reliable, and secure, which is crucial for meeting healthcare technology's growing demands. In healthcare, system availability and data integrity are crucial.

REFERENCES

[1] - B. Kalis, M. Collier, and R. Fu, “10 promising AI applications in health care,” *Harvard Business Review*, 2018.

[2] - Business Research Company, “Medical Software Global Market Report 2024 – By Software Type,” 2024.
[3] - M. J. Dickstein, K. Ho, and N. Mark, “Market segmentation and competition in health insurance,” *Journal of Political Economy*, vol. 132, no. 1, pp. 1-33, Jan. 2024.
[4] - N. Mulvaney-Day, D. Dean, K. Miller, and J. Camacho-Cook, “Trends in use of telehealth for behavioral health care during the COVID-19 pandemic: Considerations for payers and employers,” *American Journal of Health Promotion*, 2022.
[5] - T. Weil and S. Murugesan, “IT risk and resilience—Cybersecurity response to COVID-19,” *IT Professional*, vol. 22, no. 3, pp. 6-11, 2020.
[6] - G. Blinowski, A. Ojdowska, and A. Przybyłek, “Monolithic vs. microservice architecture: A performance and scalability evaluation,” *IEEE Access*, vol. 10, pp. 1-13, 2022.
[7] - U. Chouhan, V. Tiwari, and H. Kumar, “Comparing microservices and monolithic applications in a DevOps context,” in *Proc. 2023 3rd Asian Conf. on Innovation in Technology (ASIANCON)*, India, 2023, pp. 45-53.
[8] - J. E. Breneman, C. Sahay, and E. E. Lewis, *Introduction to Reliability Engineering: A Complete Revision of the Classic Text on Reliability Engineering*, USA, 2022.
[9] - S. Pargaonkar, “A comprehensive review of performance testing methodologies and best practices: Software quality engineering,” *International Journal of Science and Research (IJSR)*, 2023.
[10] - V. Millnert and J. Eker, “HoloScale: Horizontal and vertical scaling of cloud resources,” in *Proc. 2020 IEEE/ACM 13th Int. Conf. on Utility and Cloud Computing (UCC)*, UK, 2020, pp. 89-98.
[11] - M. Czuper, *Applying Automated Performance Testing with Apache JMeter*, Espoo, 2022.
[12] - B. Gupta, P. Mittal, and T. Mufti, “A review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud Platform (GCP) services,” in *Proc. 2nd Int. Conf. on ICT for Digital, Smart, and Sustainable Development*, 2021, pp. 123-130.