КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (27), 2025
ISSN 2663 - 4023

DOI 10.28925/2663-4023.2025.27.724
UDC 004.49

Illia Suprunenko
Postgraduate Student at the Department of
Informational Security and Computer Engineering
Cherkasy State Technological University, Cherkasy, Ukraine
ORCID ID: 0000-0002-1188-4804
i.o.suprunenko.asp22@chdtu.edu.ua

Volodymyr Rudnytskyi
Doctor of Engineering Science,
Department of Informational Security and Computer Engineering
Cherkasy State Technological University, Cherkasy, Ukraine
Chief Researcher
State Scientific Research Institute of Armament and Military Equipment
Testing and Certification, Cherkasy, Ukraine
ORCID ID: 0000-0003-3473-7433
rvn_2008@ukr.net

CLOUD BASED ARCHITECTURE FOR ADAPTIVE LOGGING METHOD IMPLEMENTATION

Abstract. Software technology drives a considerable amount of day-to-day processes, changing and shifting the usual way of doing things and can even provide a great aid in times of crisis. From virtual private network solutions that helped battling challenges of remote work models that were necessary during initial outbreaks of COVID-19, to artificial intelligence solutions, that transform the way people learn and research information, and cloud compute models altering the way software is written and deployed — the change is everywhere. But it also brings new dilemmas, including those related to security of software and its consumers, which means that proper protection and control over computer programs is still in high demand. This paper takes a deeper look at the observability aspect of cybersecurity and presents a model of how theoretical aspects of adaptive logging method can be deployed in a real-life web-server scenario. The model is based on the infrastructure provided by one of the largest cloud computing platforms providers and shows the application and mapping of two important formal definitions to real world services. The applicability of adaptive approach is verified and it is demonstrated that a considerable number of compute platforms should be able to incorporate and execute all the necessary components, making it suitable for different applications and use cases. Also the exclusion of dedicated security mechanisms in the formal definitions of adaptive logging method is shown to be a viable method of operation, given that services provided by the cloud can enforce necessary degree of security and still be transparent to the implementation itself.

Keywords: cybersecurity; observability; logging; debugging; cloud infrastructure; architectural model.

INTRODUCTION

Software programs influence a vast number of different aspects of human lives such as studying, getting information about recent discoveries and events, connecting with other people, completing work related tasks, shopping, creating art, researching, etc. This brings new everyday possibilities and can even drastically change the ordinary way of doing things. And yet it is crucial to not get overwhelmed by new opportunities and be prepared to face challenges that immediately follow technical progress and innovations.

Problem statement. This research is mainly related to the observability aspect of cybersecurity and how technological progress calls for better and more precise solutions.

Literature review. Over the last 10–15 years the role of software technologies became more influential and helped to overcome some of the biggest challenges in recent history. During the initial lockdown periods of COVID-19 pandemic in 2020 a huge challenge for society was to avoid halting interactions and production of goods and services. But what was previously an ordinary way of things, like commuting to work and meeting with colleagues to discuss work related tasks, was almost impossible to imagine during periods when the main task was to flatten the curve and slow down the spread of pandemic. And so many businesses and workplaces moved to remote working models, but with this came new complications: communications between employees should be protected as now they no longer have the safety of eye-to-eye communication. A solution for such an issue was presented in a form of services that utilized technology of virtual private networks (VPNs) to secure communications. It was observed that the demand for those grew by 200% and remained at elevated levels even during the fall of 2020, when first lockdowns were easing off [1]. What's noticeable is that the increase in VPN traffic was mainly observed during working hours which additionally supports the idea that this was a response to a new massive challenge and software technologies played an important role in figuring out new reality.

Some other recent major technological advancements are machine learning techniques and artificial intelligence algorithms based on large language models (LLMs). The rise of natural language processing solutions, such as ChatGPT by OpenAI which uses large processed datasets to generate text responses to queries, affected different fields, for example in relation to higher education. It is now possible to have a personalized learning experience with on-demand support in the form of reviews or even virtual assistants that can aid in understanding a given subject better [2]. But the possibilities are not limited to textual and conversation-like formats: synthesizing and generating videos, as well as replacing fragments or faces in videos of someone talking are now achievable with the use of AI [3]. And yet there are certain dangers related to such technological innovations: a study shows that increasing speed and complex decision driven logic of AI-driven cyberattacks will make existing cyber defense measures outdated [4]. As different elements in the cybersecurity kill chain have different ways to utilize artificial intelligence (reconnaissance, access and penetration, delivery, exploitation or action on objectives,) it is increasingly important to invest in appropriate — possibly also AI-based — cybersecurity measures that can help battling new emerging threats.

The rise of complex and sophisticated software solutions also affected the way the final product reaches its consumers, as well as altered the requirements for scale and performance. From simple web servers that work in a stateless request-response manner to more complicated tasks like training large language models, it becomes quite common to rely on cloud computing services rather than on establishing one's own infrastructure manually. The move to cloud computing model was also observed in universities [5], as it is an important step in online education, globalization, high and constantly changing requirements. Even though the adoption percentage was fairly low, with around 71% of respondents from different IT departments in universities stating that they do not think to change their current computing model, it is still considered to be a conceivable next step. And as different programming solutions are more and more written with the help of — and for — the cloud, the concerns related to observability aspects of cybersecurity also shift. Oftentimes to properly manage, monitor and keep track of program's execution somewhere on a remote machine a special service is needed. For example, the Microsoft Azure cloud platform provides special utilities, such as Azure Monitor and Azure DevOps Services [6], to help developers collect, analyze and act on recorded data to maximize availability and track issues in their products. As more and more workloads move to cloud-based deployment models it becomes increasingly important to tackle the issues related to observability precisely and in a timely manner.

The aim of the article. Develop a practical architectural model that allows to deploy an implementation of an adaptive logging method [7] into the cloud infrastructure of one of the prominent cloud providers.

RESULTS AND DISCUSSION

An adaptive logging method is a one possible approach that can aid in handling common observability issues in software systems on different platforms, programming languages and for various software products. Its latest form has several mechanisms that make debugging easier [8]: filtering out log message calls, overriding configuration of what to include and what to skip, working with a special dynamic message variant that allows to define "on-the-fly" computation that can be used when generating log entry (and an additional mechanism to make sure that those computations are valid and don't introduce unexpected effects as logging is more of a helper functionality that should not significantly affect software system). This work is mainly focused on integrating theoretical aspects described previously in a real-world cloud-based deployment scenario leveraging infrastructure provided by Amazon Web Services (AWS) [9]. As such, the main research method used in this paper is modelling.

The main service used for the model is Elastic Cloud Compute (EC2) which is basically a form of computational resources physically distributed around the globe that can be "rented" for some time and then returned back to the provider. There are many different classes of EC2 instances for different purposes, such as "C5n"-instances optimized for computationally intensive workloads because of the high-speed communications between nodes (provided by the Nitro System and Elastic Fabric Adapter) and they can deal with different tasks, like benchmarking the performance of large-scale computational fluid dynamics applications conducted in [10]. Because the main use case for adaptive logging mechanism is to help developers write better software, it is not urgently required to have high computation power or fast connectivity, so a simpler instance can be used with a more typical task of creating a web server that handles plain HTTP requests.

All resources created in AWS cloud are placed in a Virtual Private Cloud (VPC), a special type of resource that wraps some other resources and then can be connected to the public internet. Fig. 1 describes all the expected parts:

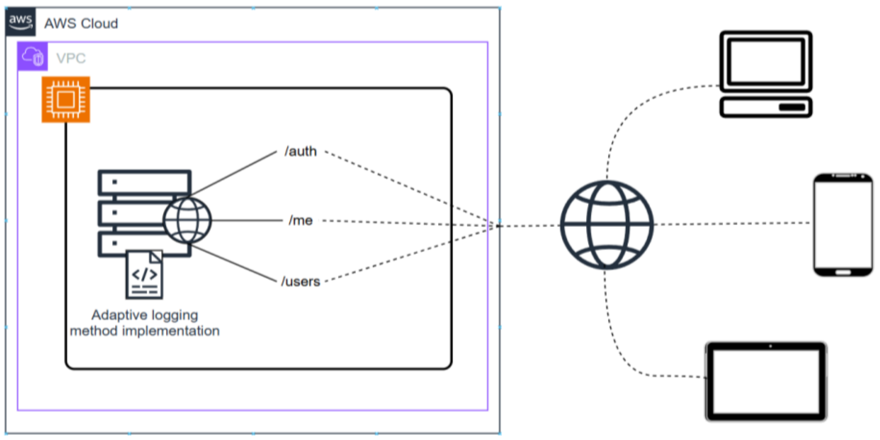

Fig. 1. Webserver (with implemented adaptive logging method) on EC2 placed in AWS cloud

Webserver is expected to run using computational resources provided by EC2, which in turn is placed in a VPC inside AWS cloud with connectivity to public internet and different clients. Three expected HTTP routes are "/auth" — which deals with authenticating users, "/me" — which returns information about currently logged in user and "/users" — route for users with administrative permissions that lists all users of the system. This example might seem trivial functionality wise, but this is just enough to showcase expected flow of adaptive configuration. Also webserver has a module that implements adaptive logging method, which in turn is used in all three routes and allows to observe internal processes.

There are two main formal functions that need to be implemented:

$$f_{log\ adp} = f(Sev, M_{DU}, T_{incl}) \tag{1}$$

which is a formal signature description of a method in an adaptive logging method implementation that is responsible for creating log messages. Parameter Sev specifies a level of "severity" chosen from an ordered set of possible values that describe how important a given invocation is, parameter M_{DU} is a related log message in either textual or script-like notation (that can be evaluated "on the fly") and parameter T_{incl} represents a set of tags — small textual identifiers, human-readable or not — that is used when deciding which invocations should be skipped or included in final output.

$$f_{init} = f(Sev, C) \tag{2}$$

which represents a method for (re-)initialization of the code abstraction that is responsible for implementing adaptive logging singleton and includes parameter Sev, which is the lowest severity level that is considered to be interesting for current debugging session and only messages with levels this or higher should be outputted, as well as configuration object C. Its internal structure consists of three main parts:

$$C = \begin{cases} T_{11}^{mod} \cap T_{12}^{mod} \cap \ldots \,||\, T_{21}^{mod} \cap T_{22}^{mod} \cap \ldots \,||\, \ldots \\ M_{dyn} = Map < string, string > \\ M_{dyn\ schm} = Map < string, O_{JSON-schema} > \end{cases} \tag{3}$$

First part is a set of tags combined on the top level with "||" operator that has properties similar to Boolean "OR" operator, and in each individual "or"-segment tags are connected using "∩" operator, which is basically a version of logical "AND". This provides an adaptive logging method implementation with the necessary set of checks to determine whether a given log invocation should be skipped or executed and functions based on the idea that only those invocations, that have at least one matching "or"-segment, where each "and"-segment matches the tag for that log call, should generate output — all the others can be skipped saving resources and time. Second is a dictionary-like structure that maps dynamic script-like log message definitions with corresponding identifiers used in code, so that those definitions can easily be changed without altering source code itself. Finally, for validation purposes a dictionary of schemas in JavaScript object notation [11] mapped to corresponding identifiers, adding the ability to run at least some verification that dynamic messages are more or less reliably doing what they are supposed to do and don't have unwanted side effects.

Taking into consideration the fact that current work is focused on creating an architecture in one of the most popular cloud provider infrastructures, it is worth mentioning that equation (1) can be omitted at this level of discussion, as it works solely in code implementation. However, (2) and (3) most certainly should be discussed here as to properly implement those some specific AWS capabilities are required.

In its entirety equation (3) was designed to be serializable to improve portability and lower the requirements to both hardware and software that a given implementation would require. As a result, there is no need to rely on runtime mechanisms of a certain platform or programming language — any adaptive logging singleton implementation should work fairly well just being able to read some textual file and interpret (or parse) it to derive configuration object (3). And so this is the first architectural decision that can be trivially implemented in the Elastic Cloud Compute service of AWS (because each virtual machine instance has at least some way of interacting with textual files) — a file-based configuration placed somewhere on the machine with a web server in our example.

Next step in making the change propagation mechanism adaptive is to use operation system process signals to start reinitialization procedure [12]. With this approach set up it is possible to adapt to changes in logging requirements "on the fly", altering the output, but persisting the state. As with file-based configuration, this mechanism is really general and is expected to be present on a large percentage of modern operating systems (for example compute instances in AWS cloud are created using Amazon Machine Images, that basically contain an operating system to run inside said virtual instance, and AMI family that AWS provides — "Amazon Linux AMI"-s — are, as name suggests, based on Linux OS, so process signals should work on those out of the box).

The final piece of the puzzle is related to the process of establishing safe and reliable connection to the virtual instance in the cloud. Once again, adaptive logging method omits this from its own formal definitions as it is expected to be handled elsewhere and the reason for this is that such concern is solved a lot easier on a different level than the one where main formalization of the method happens in. One possible approach is to use Secure Sell (SSH) [13], which is a software-based approach to network security that encrypts data when it is sent by computer to the network and then decrypts it on the receiving end. AWS does provide the ability to add this form of communication to a virtual machine by providing a pair of SSH-keys, but proper management and rotation of those keys is a complex task and, if lost or stolen, can be a dangerous vulnerability point of the system (additionally, SSH generally works on a well-known port which can also provide a way for an attacker to break in or disrupt user sessions on an instance). As an alternative, Amazon cloud provides another special service called Session Manager [14], that can be used to establish secure connection to an EC2 instance without opening any ports and relying on any file-based keys. It works by installing a special client software on the machine that needs to be controlled "from outside" and only opening outbound connections required for this client to communicate with the server part of the Session Manager. Then the connection and action permissions are managed by special abstractions in AWS cloud, like users, roles and policies, and using a browser based interface it becomes possible to connect to a virtual machine. It has a huge range of capabilities, such as session tracking using general log generating and processing mechanisms, access control based on aforementioned roles or users, session expiration mechanism and finally is highly portable because of its web based implementation. Such rich variety of features (most of which are crucial for confident management of remote machines) would overcomplicate the basics of an adaptive logging method, which is why those were designed to be handled by some external mechanisms. Additionally, the exclusion of such mechanism from the method itself means that there are basically no restrictions on how tight the security of inbound connections should be and each implementation can decide on the appropriate level it requires.

This setup is demonstrated in Fig. 2:

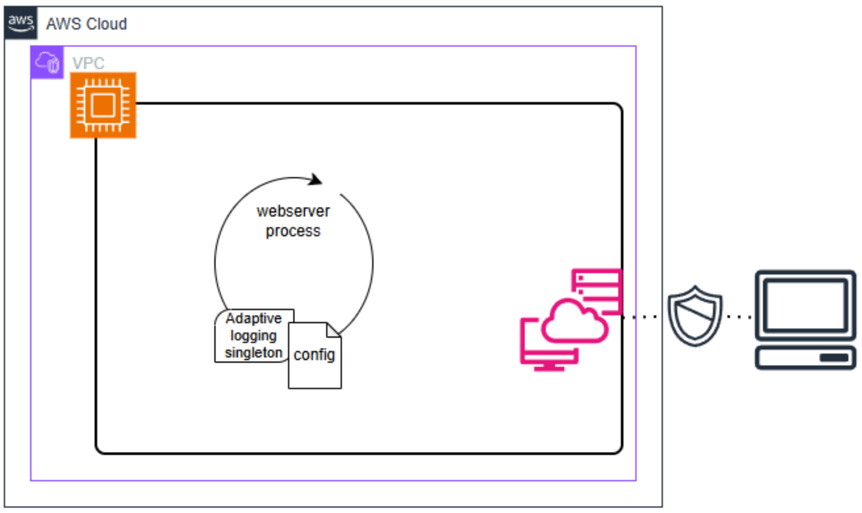

Fig. 2. EC2 architecture with file-based adaptive logging configuration and Session manager client with secure connection to an administrative machine

Together with the EC2 setup shown in Figure 1 this approach allows to finally describe how adaptive logging can be used in a typical web-server scenario. Generally, web-servers operate using HTTP routes, in our case "/auth", "/me" and "/users", which provide corresponding business logic functionality, add layers of protection (if necessary) such as authentication, client address blacklisting, etc., may add constructs that allow to improve performance and finally allow to introduce higher levels of observability for a given endpoint. A typical approach used in software development is to have log reporting be based on different severity levels and fine-tune it in development or testing stages, but leave only the most severe and dangerous cases to be reported in the production environment. While this does make sense, the approach is not ideal: for example, if all three of web-server in an example routes are exhaustively covered with debugging statements, which are properly marked with corresponding severity, then when lower severity level reporting is enabled, using levels like "info" or "debug", it automatically starts generating information from everywhere, even if the current task at hand does have such requirement. In Fig. 1 if an issue arises in "/users" endpoint and "debug" severity level reporting is activated, it could potentially trigger logging from "/auth" route on each request (if it serves both as a first time login endpoint and as an authentication revalidation method, for example), which consequently leads to more noise and unwanted details that need to be filtered either automatically using some special tools or manually. This is where log tagging provides a much better precision when inspecting a particular part of the software system. However, while it is very well possible to simply restart the web-server with the new configuration, there are occasions where it is not desirable, such as when an issue might be caused by some ephemeral state inside the operating system process itself (memory, temporary files, etc). Using the setup from Fig. 2 it becomes possible to securely connect from an administrator controlled machine into an active cloud-based compute instance and trigger reinitialization procedure inside an adaptive logging method implementation (by utilizing process signals in an operating system to initiate reload of currently active configuration object and to pull the updated one from text-based configuration file nearby), persisting most (if not all) of the stateful parts of the runtime.

The resulting mechanism has some resemblance with existing tooling for client-side development in modern web industry: tools that are called “bundlers”, which are used to combine and process source code before shipping it to end users sometimes offer the ability to replace parts of the code base “on the fly” without the need to reload the page and erasing the danger to lose temporary state of components or modules. This feature is called “hot module replacement” and is a common technique in renown development tools such as webpack and vite [15]. In a way, the reinitialization part of the adaptive logging approach can be thought of as “hot module replacement”, but for observability related concerns, where “modules” are not exactly software components that comprise main functionality, but rather helper segments that make debugging easier.

CONCLUSIONS AND PROSPECTS FOR FURTHER RESEARCH

This paper presented an architectural model of real life application of adaptive logging method using infrastructure services provided by one of the world’s largest cloud providers — Amazon Web Services. Using Elastic Cloud Compute service to host a webserver, combined with the ability to establish a secure connection to a virtual machine using session manager service, allowed to bring theoretical aspects of adaptive logging method into the real world. This setup is demonstrated to not only be able to adapt to changing reporting requirements during the development phase, but also to achieve this without the need to restart processes, and as a result — without losing internal process state. Serializable nature of the configuration parameters and reliance on a widely available mechanism of operating system’s process signals made it simple enough to find appropriate components needed to design a proper implementation. While the example is mainly focused on a web-server use case (because of its simplicity and clear separation of different parts of the system dictated by HTTP routes), this method can as well be used in other applications that run on AWS provided infrastructure (and possibly in other cloud provider environments as well, but with slightly adjusted architectures). Possible prospects for further research include comparing the differences between simple web-server based adaptive logging setups in different major cloud providers (such as Microsoft Azure or Google Cloud), as well as looking into more advanced use cases such as implementations for prebuilt software products like native programs and mobile applications, or it might be worth looking into how software products written in different programming languages can introduce adaptive logging method into their code base.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Feldmann, A., Gasser, O., Lichtblau, F., Pujol, E., Poese, I., Dietzel, C., Wagner, D., Wichtlhuber, M., Tapiador, J., Vallina-Rodriguez, N., Hohlfeld, O., & Smaragdakis, G. (2021). A year in lockdown: how the waves of COVID-19 impact internet traffic. *Communications of the ACM, 64(7),* 101–108. https://doi.org/10.1145/3465212123
2. Fuchs, K. (2023). Exploring the opportunities and challenges of NLP models in higher education: is Chat GPT a blessing or a curse? *Frontiers in Education, 8*. https://doi.org/10.3389/feduc.2023.1166682
3. Sengar, S., Hasan, A., Kumar, S. & Caroll, F. (2024). Generative artificial intelligence: a systematic review and applications. *Multimedia Tools And Applications*. https://doi.org/10.1007/s11042-024-20016-1
4. Guembe, B., Azeta, A., Misra, S., Osamor, V. C., Fernandez-Sanz, L., & Pospelova, V. (2022). The Emerging Threat of Ai-driven Cyber Attacks: A Review. *Applied Artificial Intelligence, 36(1).* https://doi.org/10.1080/08839514.2022.2037254
5. Aydin, H. (2021). A Study of Cloud Computing Adoption in Universities as a Guideline to Cloud Migration. *Sage Open, 11(3).* https://doi.org/10.1177/21582440211030280

6. Gopireddy, S. (2023). Compliance automation in azure: ensuring regulatory compliance through DevOps. *International Journal of Core Engineering & Management, 7(7).*
7. Suprunenko, I., 7 Rudnytskyi, V. (2024). On specifics of adaptive logging method implementation. *Bulletin of Cherkasy State Technological University, 29(1),* 36–42. https://doi.org/10.62660/bcstu/1.2024.36
8. Suprunenko, I., & Rudnytskyi, V. (2024). Validation of dynamic message variant in adaptive logging method. *International scientific-technical journal "Measuring and computing devices in technological processes", 4*. https://doi.org/10.31891/2219-9365-2024-80-5
9. Kewate, N., Raut, A., Dubekar, M., Raut, Y., & Patil, A. (2022). A Review on AWS - Cloud Computing Technology. *International Journal for Research in Applied Science & Engineering Technology (IJRASET), 10(1).* https://doi.org/10.22214/ijraset.2022.39802
10. Dancheva, T., Alonso, U., & Barton, M. (2024). Cloud benchmarking and performance analysis of an HPC application in Amazon EC2. *Cluster Computing, 27,* 2273–2290. https://doi.org/10.1007/s10586-023-04060-4
11. Wright, A., Andrews, H., Hutton, B., & Dennis, G. (2022). *JSON Schema: A Media Type for Describing JSON Documents.* https://json-schema.org/draft/2020-12/json-schema-core.
12. Suprunenko, I., & Rudnytskyi, V. (2024). Comparison of message passing systems in context of adaptive logging method. *Visnyk of Kherson National Technical University, 2(89),* 228–234. https://doi.org/10.35546/kntu2078-4481.2024.2.32
13. Barrett, D. J., Silverman, R. E., & Byrnes, R. G. (2005). *SSH, the Secure Shell: The Definitive Guide*. O'Reilly Media, Inc.
14. Wittig, A., & Wittig, M. (2023). *Amazon Web Services in action: an in-depth guide to AWS (Third edition.).* Manning Publications.
15. Nguyen, T. A. (2024). *A comparative analysis of Webpack and Vite as build tools for JavaScript.* Haaga-Helia University of Applied Sciences. Business Information Technology. Bachelor's Thesis.

Супруненко Ілля Олександрович
аспірант кафедри інформаційної безпеки та комп’ютерної інженерії
Черкаський державний технологічний університет, Черкаси, Україна
ORCID ID: 0000-0002-1188-4804
i.o.suprunenko.asp22@chdtu.edu.ua

Рудницький Володимир Миколайович
д.т.н., професор кафедри інформаційної безпеки та комп’ютерної інженерії
Черкаський державний технологічний університет, Черкаси, Україна
головний науковий співробітник
Державний науково-дослідний інститут випробувань і сертифікації
озброєння та військової техніки, Черкаси, Україна
ORCID ID: 0000-0003-3473-7433
rvn_2008@ukr.net

ХМАРНО-ОРІЄНТОВАНА АРХІТЕКТУРА ІМПЛЕМЕНТАЦІЇ МЕТОДУ АДАТИВНОГО ЛОГУВАННЯ

Анотація. Програмні технології є рушійною силою відчутної кількості повсякденних процесів, змінюючи звичний спосіб виконання задач та навіть можуть суттєво допомогти в кризові часи. Від приватних віртуальних мереж, що допомогли впоратись із викликами моделей віддаленої роботи, які були необхідні в період перших масових проявів COVID-19, до моделей штучного інтелекту, що трансформують процеси навчання та пошуку інформації, та моделей хмарних обчислень з їх впливом на написання програмного забезпечення — зміни повсюди. Однак разом з ними з’являються нові дилеми, включно з тими, що стосуються безпеки програмного забезпечення та його користувачів, що в свою чергу вимагає належного захисту та контролю над комп’ютерними програмами. Фокус уваги цієї статті спрямований на аспект спостережності як складової частини кібербезпеки, в межах якого представлена модель реалізації теоретичних аспектів методу адаптивного логування при розміщенні на реальному прикладі веб-сервера. Основою для даної моделі є інфраструктура одного із найбільших провайдерів обчислювальних потужностей в хмарі, на реальних сервісах якого продемонстрована реалізація двох важливих формальних компонентів методу. В ході моделювання була перевірена можливість застосування адаптивного підходу, а також продемонстровано, що в загальному значна частина обчислювальних платформ має необхідні компоненти, що робить можливим реалізацію в різних застосунках. Додатково продемонстровано, що виключення спеціалізованого механізму безпеки із формальних визначень методу адаптивного логування є життєздатним підходом, оскільки необхідний рівень безпеки може бути забезпечений сервісами провайдера хмарних потужностей, завдяки чому додавання цього компоненту проходить не впливаючи безпосередньо на імплементацію методу.

Ключові слова: кібербезпека; спостережність; логування; дебагінг; хмарна інфраструктура; архітектурна модель.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Feldmann, A., Gasser, O., Lichtblau, F., Pujol, E., Poese, I., Dietzel, C., Wagner, D., Wichtlhuber, M., Tapiador, J., Vallina-Rodriguez, N., Hohlfeld, O., & Smaragdakis, G. (2021). A year in lockdown: how the waves of COVID-19 impact internet traffic. *Communications of the ACM, 64(7),* 101–108. https://doi.org/10.1145/3465212123
2. Fuchs, K. (2023). Exploring the opportunities and challenges of NLP models in higher education: is Chat GPT a blessing or a curse? *Frontiers in Education, 8*. https://doi.org/10.3389/feduc.2023.1166682
3. Sengar, S., Hasan, A., Kumar, S. & Caroll, F. (2024). Generative artificial intelligence: a systematic review and applications. *Multimedia Tools And Applications*. https://doi.org/10.1007/s11042-024-20016-1

4. Guembe, B., Azeta, A., Misra, S., Osamor, V. C., Fernandez-Sanz, L., & Pospelova, V. (2022). The Emerging Threat of Ai-driven Cyber Attacks: A Review. *Applied Artificial Intelligence, 36(1).* https://doi.org/10.1080/08839514.2022.2037254
5. Aydin, H. (2021). A Study of Cloud Computing Adoption in Universities as a Guideline to Cloud Migration. *Sage Open, 11(3).* https://doi.org/10.1177/21582440211030280
6. Gopireddy, S. (2023). Compliance automation in azure: ensuring regulatory compliance through DevOps. *International Journal of Core Engineering & Management, 7(7).*
7. Suprunenko, I., 7 Rudnytskyi, V. (2024). On specifics of adaptive logging method implementation. *Bulletin of Cherkasy State Technological University, 29(1),* 36–42. https://doi.org/10.62660/bcstu/1.2024.36
8. Suprunenko, I., & Rudnytskyi, V. (2024). Validation of dynamic message variant in adaptive logging method. *International scientific-technical journal "Measuring and computing devices in technological processes", 4*. https://doi.org/10.31891/2219-9365-2024-80-5
9. Kewate, N., Raut, A., Dubekar, M., Raut, Y., & Patil, A. (2022). A Review on AWS - Cloud Computing Technology. *International Journal for Research in Applied Science & Engineering Technology (IJRASET), 10(1).* https://doi.org/10.22214/ijraset.2022.39802
10. Dancheva, T., Alonso, U., & Barton, M. (2024). Cloud benchmarking and performance analysis of an HPC application in Amazon EC2. *Cluster Computing, 27,* 2273–2290. https://doi.org/10.1007/s10586-023-04060-4
11. Wright, A., Andrews, H., Hutton, B., & Dennis, G. (2022). *JSON Schema: A Media Type for Describing JSON Documents.* https://json-schema.org/draft/2020-12/json-schema-core.
12. Suprunenko, I., & Rudnytskyi, V. (2024). Comparison of message passing systems in context of adaptive logging method. *Visnyk of Kherson National Technical University, 2(89),* 228–234. https://doi.org/10.35546/kntu2078-4481.2024.2.32
13. Barrett, D. J., Silverman, R. E., & Byrnes, R. G. (2005). *SSH, the Secure Shell: The Definitive Guide*. O'Reilly Media, Inc.
14. Wittig, A., & Wittig, M. (2023). *Amazon Web Services in action: an in-depth guide to AWS (Third edition.).* Manning Publications.
15. Nguyen, T. A. (2024). *A comparative analysis of Webpack and Vite as build tools for JavaScript.* Haaga-Helia University of Applied Sciences. Business Information Technology. Bachelor's Thesis.