Securing the Flow: Security and Privacy Tools for Flow-based Programming

Thodoris Ioannidis
InQbit Innovations SRL.
Bucharest, Romania
thodoris.ioannidis@inqbit.io

Vaios Bolgouras
Department of Digital Systems
University of Piraeus
Attica, Piraeus, Greece
vaios.bolgouras@ssl-unipi.gr

Ilias Politis
InQbit Innovations SRL.
Bucharest, Romania
ilias.politis@inqbit.io

Christos Xenakis
Department of Digital Systems
University of Piraeus
Attica, Piraeus, Greece
xenakis@unipi.gr

ABSTRACT

This paper presents a comprehensive collection of reusable artifacts for addressing security and privacy issues in the context of flow-based programming in Function-as-a-Service (FaaS) environments. With the rapid adoption of FaaS platforms, it becomes important to guarantee the security and privacy of applications. The presented artifacts incorporate a wide variety of nodes and techniques into the popular Node-RED architecture. They intend to improve the security and privacy of applications by addressing critical aspects such as secure data flow management, code authenticity and validation, access control mechanisms, and runtime monitoring and anomaly detection. Using these artifacts, developers can construct more robust and resilient applications in FaaS environments while mitigating potential security and privacy risks.

CCS CONCEPTS

• **Cloud Security Tools** → *Security and Privacy*; • **Privacy Preserving Security** → *Anonymization Techniques*; • **Flow Programming** → *Node-RED*.

KEYWORDS

flow-programming, security & privacy tools, Node-RED

1 INTRODUCTION

In recent times, the emergence of cloud computing and Function-as-a-Service (FaaS) platforms has brought about a revolutionary change in the manner in which software applications are conceived and implemented [1]. FaaS platforms provide numerous benefits, including scalability, cost efficiency, and decreased infrastructure management [2]. Embracing this groundbreaking methodology, flow-based programming perceives an application as an amalgamation of independent processes operating asynchronously. Instead of communicating via a stream of structured data chunks, these processes commence at a specific juncture and execute a series of sequential operations, handling one task at a time until the entire process is completed. This paradigm shift in application development enables enhanced flexibility and efficiency in utilizing FaaS platforms and cloud computing resources [3].

Node-RED is an open-source, flow-based programming tool and an execution platform that wraps the *Node.js* runtime environment into a visual asynchronous process representation [1]. The asynchronous processes built in Node-RED are referred to as *nodes* and define the building blocks for developing flow-based applications. Each node defines its own way of receiving data and performing an operation on that data, it then passes the data on to the next node. The collection of these asynchronous processes, referred to as a *flow*, facilitate a black-box network which manages the data exchange between nodes. This programming approach makes it simpler for users to access multiple layers and utilize the visual nature of this platform.

However, as the adoption of FaaS services increases, so does the concern for security and privacy. Developers today face numerous challenges and issues in ensuring the security and privacy of their applications. Traditional development approaches often struggle to address the unique requirements of FaaS services. These challenges include but are not limited to:

- Data Privacy: FaaS services typically operate on shared infrastructure, raising concerns about data confidentiality and integrity. Unauthorized access to sensitive data can lead to severe consequences, such as data breaches and privacy violations.
- Code Integrity: The dynamic and distributed nature of FaaS services makes it challenging to maintain code integrity. Ensuring that the executed code is unaltered and free from malicious modifications is crucial to prevent unauthorized access or tampering.
- Access Control: Managing access to functions and resources within a FaaS environment can be complex. Developers need effective mechanisms to control access permissions, authenticate users, and enforce fine-grained authorization policies.
- Runtime Monitoring: Real-time monitoring and anomaly detection are essential to detect and respond to security incidents promptly. Traditional monitoring approaches may not be suitable for the event-driven and distributed nature of FaaS services.

[1] https://nodered.org

To mitigate these challenges and address the security and privacy concerns in flow-based programming for FaaS services, we propose the utilization of NodeRED, which offers a visual interface for building event-driven applications. Its inherent design characteristics align well with the requirements of FaaS services, enabling developers to create secure and privacy-aware applications more efficiently.

In this paper, we present several reusable artifacts developed within the framework of the H2020 PHYSICS Project. These artifacts provide security and privacy services tailored specifically for flow-based programming in FaaS environments. Our contributions include:

- Secure Data Flow Management: We introduce a set of nodes and techniques that enable secure and encrypted data flow between functions, ensuring the confidentiality and integrity of sensitive information.
- Code Verification and Validation: We propose a novel approach for verifying the integrity of function code at runtime, mitigating the risk of code tampering and unauthorized modifications.
- Access Control Mechanisms: We present a comprehensive access control framework that allows developers to define and enforce fine-grained access policies within a FaaS environment, ensuring proper authentication and authorization.
- Runtime Monitoring and Anomaly Detection: We develop monitoring mechanisms specifically designed for flow-based programming in FaaS services, enabling real-time detection and response to security incidents.

By incorporating these artifacts into the NodeRED ecosystem, developers can enhance the security and privacy posture of their applications, enabling them to confidently leverage the benefits of flow-based programming in FaaS services.

In the subsequent sections of this paper, we provide detailed descriptions, implementation guidelines, and evaluation results of each artifact. We also discuss potential future research directions and conclude by emphasizing the significance of security and privacy considerations in the evolving landscape of FaaS-based applications.

2 RELATED WORK

The emergence of new FaaS (Function-as-a-Service) models and serverless architectures has significantly altered the landscape of software development, delegating system management responsibilities to cloud providers and leaving application-level security needs to developers [4]. To address the challenges of workflow design and management in this context, several tools have been developed, offering unique features and capabilities. Apache Airflow serves as a powerful workflow design and management tool, specifically tailored for diverse data flow scenarios [5]. It utilizes Directed Acyclic Graphs (DAGs) to define the flow of data, ensuring a unidirectional movement between nodes (tasks). With Python as its primary language, Airflow enables the dynamic creation of pipelines and provides extensibility options through custom operator creation. Furthermore, it seamlessly integrates with major cloud service providers such as AWS [2], Google Cloud [3], and Microsoft Azure [4], offering additional functionalities to the workflow environment. In contrast, Apache Taverna [6] focuses on scientific workflows, providing a set of components including the Taverna Engine, the Taverna Workbench, and the Taverna Server. It caters to scientists with limited programming knowledge, enabling them to construct complex data streams for analysis. Researchers from various domains can leverage Taverna to integrate data from multiple sources and perform analyses in fields like geography, medicine, and sports. The tool prioritizes simplicity and usability to empower domain experts in their workflow design process. Camunda is an open-source workflow management tool, which emphasizes streamlined and efficient workflow design [7]. Its architecture revolves around three core principles: design, automation, and improvement. Camunda offers continuous optimization of workflow processes, making it particularly suitable for complex organizations that require maximum information visibility for users. With a lightweight Java API stack, Camunda ensures reliability and scalability, and it seamlessly integrates into cloud environments accessible from various platforms. Cflow is a cloud-based and open-source workflow management software available on AWS [8]. It simplifies workflow design and maintenance by providing a vast library of components and predefined workflows, minimizing the need for extensive coding. Cflow's flexibility and ease of integration with third-party applications like SAP make it suitable for organizations of different sizes, enabling efficient workflow management. Knime primarily focuses on data science workflows, offering an intuitive, drag-and-drop graphical interface for building complex pipelines [9]. It allows users to seamlessly combine tools from different domains, such as machine learning, R & Python scripting, and connectors to Apache Spark. Knime's strength lies in its versatility and the ability to visually create workflows without the need for extensive coding.

Considering the various workflow tools available, NodeRED was chosen for our project due to its specific characteristics and alignment with our objectives. NodeRED provides a visual interface for flow-based programming, allowing developers to create secure and privacy-aware applications efficiently. Its integration capabilities with FaaS services, such as AWS, Google Cloud, and Microsoft Azure, make it well-suited for our needs. Additionally, NodeRED's extensibility and the availability of a vast ecosystem of nodes and contributed modules further enhance its appeal for developing secure and privacy-focused applications in the FaaS environment. While each workflow tool discussed has its unique features and areas of specialization, NodeRED's visual interface, integration capabilities, and extensibility make it a valuable choice for our project focused on security and privacy in flow-based programming for FaaS services. By leveraging NodeRED, we aim to provide reusable artifacts that address the challenges faced by developers in the context of FaaS and enhance the security and privacy of their applications.

[2] https://aws.amazon.com
[3] https://cloud.google.com
[4] https://azure.microsoft.com

2.1 VISUAL FLOW PROGRAMMING

Node-RED offers a visual programming environment that allows users to create and manage their programming flows through a drag-and-drop node palette interface. Flows can visually represent the data and the internal logic followed, making this programming approach easier for developers to comprehend, design and build flows, without extensive coding requirements, enabling rapid prototyping and speeding up the development process.

Despite the user-friendly interface, developing secure intricate flows can be difficult compared to conventional coding approaches. Multiple interconnected nodes in a flow may require extensive testing and logging, in order to identify the operational issues and how to resolve them. Proper error handling and exception management within flows can be crucial for robustness and reliability. Recognizing potential failure points, implementing error handling nodes, and effectively managing exceptions can be challenging, particularly in larger flows. The programming environment provides utilities for debugging, monitoring and identifying problems, but there is still a learning curve associated with grasping the principles and concepts of flow-based programming and how the nodes work. Proper documentation of flows, including the purpose, functionality, and usage instructions of nodes, is important for sharing knowledge and engaging new users. Keeping documentation up-to-date and providing clear explanations can be time-consuming, but they are essential for effective collaboration.

Node-RED has a broad range of pre-built nodes and services, facilitating easy connectivity solutions and interactions with other infrastructure components. As flows increase in size and complexity, performance optimization becomes important. It can be difficult to manage resources and comprehend the numerous interconnections between the nodes. Keeping track of data flow, logic, and dependencies within a complex flow can require meticulous documentation and organization. Identifying and mitigating performance bottlenecks, such as excessive computation, inefficient data processing, or network delays, requires careful analysis and optimization techniques, while also larger flows can consume more system resources impacting the scalability of these applications.

2.2 SECURE FLOW DEVELOPMENT

When it comes to the development of secure processes in flow-based programming (**FBP**), there are several challenges to consider and address:

- **Secure Data Handling**: FBP involves the flow of data between nodes, services and external components.Validating incoming data and ensuring its security within a flow is vital. Implementing data validation mechanisms, sanitizing inputs, and applying security practices, such as encryption and access control, is essential to implement measures to protect the privacy and security of the data throughout its journey within the flow. This includes encryption, secure storage, and secure communication channels to prevent unauthorized access or data leakage.
- **Integrating External Systems**: FBP environments often interact with external systems, APIs, and services, where it is essential to ensure secure integration, including validating and sanitizing inputs, implementing secure communication protocols, and securely managing authentication credentials for accessing external resources.
- **Privacy Compliance Regulations**: Depending on the industry or region, there might be specific security and privacy regulations that need to be adhered to. It is important to understand and comply with relevant standards (such as GDPR, HIPAA, or ISO 27001) when designing security and privacy requirements in Node-RED. Ensuring compliance with these regulations requires implementing privacy-preserving techniques, obtaining necessary user consent, and adopting privacy-by-design principles throughout the flow development process.
- **Authentication and Authorization**: FBP environments need to incorporate mechanisms for authenticating and authorizing users and components within the flow. Ensuring that only authorized users and components can access and manipulate sensitive data is necessary for maintaining security and preserving privacy.
- **Logging and Auditing**: Implementing comprehensive logging and auditing mechanisms within the FBP environment is quite important for security and privacy compliance. Logs can help track events, detect anomalies and potential security risks, and provide an audit trail for investigating potential security incidents or privacy breaches.
- **Vulnerability Management**: FBP environments may rely on various third-party components and libraries. It is crucial to monitor and address security vulnerabilities in these components through patching and updates. Regular vulnerability assessments and penetration testing could help identify and mitigate potential security risks.
- **Education and Awareness**: Flow-based programming introduces its own unique security and privacy considerations. Developers and users need to be educated and aware of these considerations to ensure they understand the potential risks, best practices, and how to implement security and privacy measures effectively.

Addressing these challenges requires a combination of security and privacy best practices, adherence to relevant regulations, and a thorough understanding of the specific security requirements within the context of the FBP environment.

3 SECURITY AND PRIVACY TOOLS

In this paper, we present a collection of reusable Node-RED nodes, with their corresponding subflows, for addressing the security needs and privacy requirements of FaaS services in flow-based programming. The nodes implement cryptographic and privacy preserving operations that provide fundamental security building blocks for building secure flows.

3.1 GENERATING ENCRYPTION KEYS

Generating strong and secure encryption keys using Password-Key Derivation (*PKD*) functions, organizations and individuals can enhance the security of their encryption keys, making it significantly more challenging for adversaries to compromise sensitive information. These PKD functions serve as a critical defense mechanism

against various password-related attacks, ensuring the confidentiality and integrity of data.

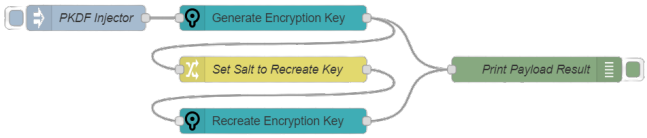

Figure 1: *Password Key Derivation Function Subflow*

The subflow of the figure utilizes the Password-Key Derivation Function Node-RED node that is used for generating and recreating encryption keys. It can be utilized for password-based encryption and decryption, ensuring that only authorized users with the correct password can access the protected data. Cryptographic PKD functions allow for flexible and adaptable key derivation schemes. They can be adjusted to balance security requirements with the computational resources available on the system. By tuning parameters like iteration count, memory usage, and parallelization factor, the key derivation function can be customized to match the desired security level without compromising the system's performance. The node can be configured for creating keys, along with their corresponding salt, of *128 bit*,*192 bit* and *256 bit* size and configuring the number of iterations the PKD function is going to be performed on the password.

3.2 ENCRYPTING DATA IN TRANSIT

Encrypting and decrypting data travelling in a flow using the Advanced Encryption Standard (*AES*). Employed in FaaS services, AES helps protect sensitive data, ensures compliance with regulatory requirements, and allows for efficient and secure data exchange. It is an essential security measure to safeguard sensitive information in the dynamic and distributed environment of FaaS architectures.

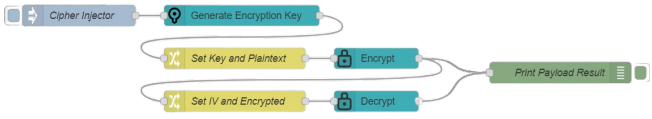

Figure 2: *Cipher Subflow*

In this figure the Cipher Node-RED node is utilized, along with Password-Key Derivation Function node, to generate secure encryption keys and use the generated keys to encrypt the message payloads of the flow. AES is a widely adopted and trusted symmetric encryption algorithm. It provides strong encryption, ensuring that sensitive data remains confidential even if it is stored or transmitted through FaaS services. By encrypting the data with AES, you can protect it from unauthorized access by anyone who does not possess the encryption key. Supported key and IV sizes include *128 bit*,*192 bit* and *256 bit*, while the node also offers several AES modes (*CBC, ECB, CFB, OFB, CTR*,*GCM*) to be used in the ciphering operation.

3.3 HASHING DATA IN TRANSIT

Generating hashes and hash-based message authentication code by utilizing the Secure Hash Algorithm (*SHA*). This can enhance data integrity, protect sensitive information, detect data tampering, and enable secure authentication mechanisms. SHA algorithms provide a reliable and efficient means to verify the integrity of data, making them an important component of secure data processing in FaaS environments.

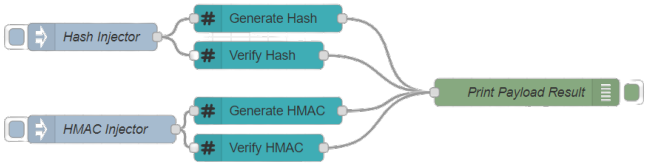

Figure 3: *Hasher Subflow*

The subflow of the figure makes use of the developed Hasher Node-RED node that generates SHA payload hashes and Hashed Message Authentication Codes (*HMAC*).SHA hashing ensures the integrity of data by generating a unique hash value for a given input. Even a minor change in the input data will produce a significantly different hash value. By storing and comparing the hash values, you can verify if the data has been tampered with or modified. This is particularly important in FaaS services where data may be processed and passed between different functions or services. The node allows configuring the size of the produced hash, the hash to check to verify against and, if the HMAC option is chosen, the key to use in the hashing operation.

3.4 AUTHENTICATING DATA IN TRANSIT

Key-pairs creation for data signing and verification that leverages the Public Key Infrastructure (*PKI*) model. Organizations can establish data authenticity, non-repudiation, secure communication channels and trust relationships. The node can enable secure data exchange, compliance with regulations, protection against unauthorized access or tampering, enhancing the overall security and reliability of the FaaS service.

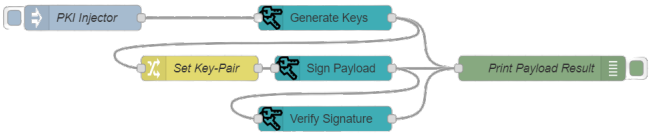

Figure 4: *Public Key Infrastructure Subflow*

In this subflow the PKI node is used to generate Ed25519 keypairs, sign message payloads and verify payload signatures.PKI signing and verification ensure the authenticity of data. With PKI, data can be signed using a private key, which can only be generated by the owner. The signature is unique to the data and the private key, providing a strong indication that the data has not been tampered with and that it originated from the expected sender. Verifying the signature using the corresponding public key establishes the authenticity of the data. The properties of the node can be

configured to generate new key-pairs, sign data payloads provided the private key, and verifying data payloads provided the public key. Both keys provided must be in *hex* format.

3.5 PRESERVING PRIVACY OF DATA

Preserving the privacy of data in FaaS services is vital for protecting sensitive information, complying with regulations, fostering user trust, mitigating security risks, safeguarding intellectual property, and adhering to ethical principles. It ensures that data remains confidential, secure, and used in a manner consistent with individuals' expectations and legal requirements.

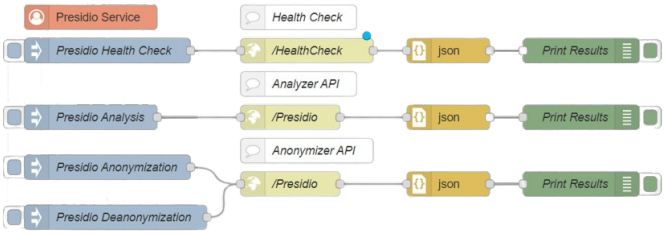

Figure 5: *Presidio Subflow*

The subflow of the figure integrates the external Microsoft Presidio[5] service through API calls, providing fast identification and anonymization of private data in text and images such as credit card numbers, names, locations, social security numbers, financial data and more [10]. FaaS services often process and store sensitive data, referred to as personally identifiable information (PII), financial records, health data, or trade secrets. Preserving the privacy of this data ensures that it remains confidential and inaccessible to unauthorized parties. It helps prevent identity theft, fraud, unauthorized disclosure, or misuse of sensitive information. The node allows configuring the Presidio Service endpoints for identifying sensitive payload data (*Presidio Analyzer*) and anonymization and deanonymization processes (*Presidio Anonymizer*) for the flow.

4 USE CASES

The nodes developed offer a range of security and privacy capabilities within the Node-RED environment, empowering developers to implement secure and privacy-preserving functionality in their applications. These use cases demonstrate how these nodes can be employed to protect sensitive information, ensure data integrity, enable secure communication, establish trust, and comply with privacy regulations.

The PKDF (*Password Key Derivation Function*) Node is valuable for securely deriving encryption keys from passwords. This can be crucial for securing user credentials, sensitive documents, or private communication channels.

The Cipher Node can be used for secure encryption and decryption of sensitive data, including encrypting and decrypting files or messages for secure transmission or storage, encrypting passwords or sensitive information in databases, or encrypting data in transit between FaaS services.

[5]https://microsoft.github.io/presidio/

The Hasher Node can be useful for generating cryptographic hash values of data. This can be applied in scenarios such as data integrity verification, password storage (by hashing passwords instead of storing them in plaintext), verifying the integrity of software packages or code, or generating unique identifiers for anonymization purposes.

The Public Key Infrastructure (*PKI*) Node enables operations related to digital certificates, public and private key pairs, and signing/verification processes. It can be employed for secure communication between FaaS services by encrypting data with recipient's public keys, verifying digital signatures for authenticity and integrity, or establishing trust relationships based on trusted certificate authorities.

The Presidio Anonymization Service Node provides functionalities for preserving privacy by identifying and anonymizing sensitive data. It can be used in use cases such as data anonymization for compliance with privacy regulations, generating synthetic datasets for research or analysis while protecting individuals' identities, or transforming personal data into pseudonymous identifiers to ensure privacy in data processing operations.

5 DISCUSSION

This collection's security-focused nodes introduces advanced cryptographic techniques, enabling secure data transmission, storage, and verification. The privacy-focused node integrates privacy enhancing technologies in addition to security measures. The incorporation of secure key management mechanisms could enable users to securely store and regulate access to their cryptographic keys, ensuring that only authorized parties could use and interact with them. The incorporation of web3 technologies into Node-RED could enable developers to enhance their flow-based application into through smart contract interactions. Smart contract nodes could enable interaction with blockchain networks in a seamless manner, allowing developers to create, deploy, and manage smart contracts directly within the Node-RED environment. These nodes could facilitate the automation of contractual agreements, removing the need for intermediaries and providing the application's logic and data with transparency and immutability. By integrating blockchain technologies and smart contract interactions using Node-RED nodes, developers can harness the power of decentralized networks while maintaining the highest levels of security and user privacy.

6 CONCLUSION

In conclusion, this paper has presented a comprehensive collection of reusable artifacts that address security and privacy challenges in flow-based programming for Function-as-a-Service (FaaS) environments. These artifacts, seamlessly integrated into the Node-RED framework, enhance the security and privacy posture of FaaS applications by focusing on secure data flow management, code verification and validation, access control mechanisms, and runtime monitoring and anomaly detection. By leveraging these artifacts, developers can build more robust and resilient applications while mitigating security and privacy risks in FaaS environments. The presented artifacts contribute to the ongoing efforts in ensuring the secure and privacy-aware adoption of FaaS platforms and provide valuable resources for future research and development in this area.

ACKNOWLEDGMENTS

This research has received funding from European Commission's Horizon Europe and Horizon 2020 research and innovation programs under grant agreements No. 824015 (INCOGNITO), No. 101017047 (PHYSICS)

REFERENCES

[1] J. Spillner, Serverless computing and cloud function-based applications, in: Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion, 2019, pp. 177–178.

[2] A. Wang, S. Chang, H. Tian, H. Wang, H. Yang, H. Li, R. Du, Y. Cheng, Faasnet: Scalable and fast provisioning of custom serverless container runtimes at alibaba cloud function compute, in: 2021 USENIX Annual Technical Conference (USENIX ATC 21), 2021.

[3] E. Paraskevoulakou, D. Kyriazis, Ml-faas: Towards exploiting the serverless paradigm to facilitate machine learning functions as a service, IEEE Transactions on Network and Service Management (2023).

[4] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. Netto, et al., A manifesto for future generation cloud computing: Research directions for the next decade, ACM computing surveys (CSUR) 51 (5) (2018) 1–38.

[5] B. P. Harenslak, J. de Ruiter, Data Pipelines with Apache Airflow, Simon and Schuster, 2021.

[6] S. Soiland-Reyes, I. Dunlop, A. R. Williams, Apache taverna: Sustaining research software at the apache software foundation, in: Bioinformatics Open Source Conference, 2015.

[7] J. Freund, B. Rücker, Real-life BPMN, Camunda, 2012.

[8] M. Li, C. Chen, C. Hua, X. Guan, Cflow: A learning-based compressive flow statistics collection scheme for sdns, in: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, 2019, pp. 1–6.

[9] S. Beisken, T. Meinl, B. Wiswedel, L. F. de Figueiredo, M. Berthold, C. Steinbeck, Knime-cdk: Workflow-driven cheminformatics, BMC bioinformatics 14 (2013) 1–4.

[10] I. Pilán, P. Lison, L. Øvrelid, A. Papadopoulou, D. Sánchez, M. Batet, The text anonymization benchmark (tab): A dedicated corpus and evaluation framework for text anonymization, Computational Linguistics 48 (4) (2022) 1053–1101.

Received XX June 2023; revised XX June 2023; accepted XX June 2023