# Adaptive Observability for Forensic-Ready Microservice Systems

Davi Monteiro, Yijun Yu, Andrea Zisman, and Bashar Nuseibeh

**Abstract**—Microservice-based applications may include multiple instances of microservices running on containerised infrastructures. These infrastructures pose challenges to digital investigations of security incidents because digital evidence can be destroyed when containers are terminated. Observability techniques are used to facilitate the investigation of incidents in microservice systems. However, existing observability approaches do not address security incidents when there is a need to perform digital forensic investigations. Furthermore, approaches to proactively support digital forensic investigations are limited to security incidents that are known a priori. In this paper, we propose an adaptive observability approach based on game theory. The approach addresses the challenge of implementing forensic-ready microservice systems while considering uncertainties in security incidents. Our approach provides evidence collection capabilities for microservice systems and continually adapts to improve the forensic readiness of microservices. Specifically, the approach uses game theory to model and reason about the interactions between users and microservices, determining the optimal time and manner for observing microservices before the occurrence of security incidents. The performance of the approach has been assessed and compared with other observability approaches. Results of the evaluation indicate that adaptive observability outperforms other observability approaches, with improvements ranging from 3.1% up to 42.50%.

**Index Terms**—microservices, forensic-ready systems, observability, game theory

## 1 Introduction

Microservices have emerged as an architectural style for building a single application as a collection of fine-grained, loosely coupled, and autonomous services, in which each service runs in its process and communicates through lightweight mechanisms [1]. This architectural style focus on isolating microservices around business capabilities, employing well-established software modularisation principles such as *separation of concerns* [2] and *bounded context* [3]. As a result, each microservice functions as an independent unit for development, testing, deployment, operation, and scalability [4], [5].

Microservice-based applications may include hundreds to thousands of instances of microservices running on containerised infrastructures (e.g., Kubernetes and Docker Swarm). These containerised infrastructures provide auto-scaling and self-healing capabilities to create multiple instances of microservices scaling up the application and replace malfunctioning microservices. However, these infrastructures pose challenges when a security incident occurs, and a digital forensic investigation needs to be conducted. Containers are ephemeral by nature - they have a specific lifetime and; when terminated, their internal storage is destroyed together with any potential digital evidence. Therefore, it is necessary to anticipate security incidents and prepare microservice-based systems to collect potential digital evidence for future investigations.

Traditionally, digital forensics is a reactive process initiated after the occurrence of security incidents [6]. In contrast, *digital forensic readiness* aims to support digital investigations proactively [7]; i.e., preparing software systems to collect potential digital evidence before security incidents occur. Some strategies and guidelines [7]–[9] have been proposed to collect digital evidence proactively. Other methods and techniques have been developed to implement forensic-ready systems focusing on the reduction of data necessary to investigate security incidents [10], [11]. However, these existing approaches depend on the assumption of prior knowledge about security incidents to synthesise evidence-gathering capabilities, which limits their effectiveness in uncertain scenarios. Examples of this type of scenario include cases where a forensic-ready system needs to collect possible digital evidence under: (i) uncertainty about the type of user (e.g., regular or malicious) who is interacting and (ii) uncertainty about the behaviour of malicious users.

More recently, a set of instrumentation techniques (e.g., metrics, logs, and distributed tracing), referred to as *observability*, have been used to facilitate the investigation of incidents in microservice systems [12]–[14]. According to Polyakov [15], there are three observability approaches, namely: (i) full observability: when all observability data is collected (metrics, logs, and traces); (ii) sampling observability: when observability data with a fixed probability is collected; and (iii) adaptive sampling observability: when a prior probability is adjusted to assure that the amount of data collected is stable based on levels of traffic workload. Full observability brings extra overhead to microservice systems, while sampling and adaptive sampling observability aim to reduce the cost of data collection [16]. However, in the context of a security incident, approaches (ii) and (iii) may

- *Davi Monteiro (davi.monteiro@ul.ie) is with Lero, the Irish Software Research Centre, University of Limerick, Limerick, Ireland.*
- *Yijun Yu (yijun.yu@open.ac.uk) and Andrea Zisman (andrea.zisman@open.ac.uk) are with the Open University, Milton Keynes, United Kingdom.*
- *Bashar Nuseibeh (bashar.nuseibeh@lero.ie) is with Lero, the Irish Software Research Centre, University of Limerick, Limerick, Ireland and the Open University, Milton Keynes, United Kingdom.*

*Manuscript received April 19, 2005; revised August 26, 2015.*

not record the data required to support digital investigations due to the sampling process.

In this paper, we propose a game-theoretical approach, named *adaptive observability*, for implementing forensic-ready microservice systems. Our approach uses game theory to formalise representations and reasoning of the interactions between users and microservices. In particular, game theory provides a framework for reasoning about security incidents involving uncertainty scenarios [17]. These representations model how malicious users and forensic-ready microservices should make decisions when interacting. In addition, the representation of such interactions as a game model helps us capture uncertainties around security incidents (e.g., the presence of malicious users, types of attacks that may occur, and possible impacts of these attacks). Therefore, we reason about these representations to determine when and how to observe microservices before the occurrence of security incidents.

Furthermore, we consider a set of requirements, known as *forensic readiness requirements* [18], in the implementation of our approach. Such requirements are important to bridge the gap between the concept of digital forensic readiness and its implementation in software engineering practices. For example, rather than providing forensics readiness guidelines, these requirements specify how forensics-ready systems must behave to achieve forensics readiness. We evaluated our approach, and the results indicate that adaptive observability outperforms full and sampling observability using the F-measure as a criterion to perform the comparisons. Specifically, our approach shows significant improvement (up to 42.50% better performance) in scenarios with high uncertainty of malicious users. The main contributions of this work are the following:

- Representations of malicious users and forensic-ready microservices;
- A game-theoretical model to reason about interactions of users and microservices;
- An empirical evaluation of adaptive observability compared to other observability approaches.

The remainder of this paper is organised as follows. In section 2 we describe a motivating example. In section 3 we provide an overview of adaptive observability. In section 4 we define representations of malicious users and forensic-ready microservices. In section 5 we formalise our game-theoretical model. In section 6 we describe how the adaptive observability approach can be used in practice. In section 7 we evaluate our approach and discuss. In section 8 we provide an account of existing work. Finally, in section 9 we describe some concluding remarks and future work.

## 2 Motivating Example

Our work is motivated by an example of a microservice system for intelligent video surveillance, inspired by one of Amazon's machine learning sample solutions, and used as a running example to illustrate self-adaptive microservice systems [19]. The application aims to inform users about the presence of humans via real-time analysis of video frames captured by security cameras.

The architecture of this application consists of two types of microservices: business and infrastructure. A business microservice refers to a service that supports a specific business function, such as face recognition and video playback; whereas an infrastructure microservice encompasses more generic and reusable services that handle non-functional tasks, such as service discovery and service orchestration.

**Scenario.** Consider a scenario where two users, Bob and Alice, have access to the video surveillance application. Bob is a malicious user who intends to compromise the system availability, whereas Alice is a regular user who has no intentions of attacking the system. In order to achieve his malicious objective, Bob can employ various types of attacks. However, the potential impact of an attack depends on the type of microservice targeted by Bob.

The system encounters two forms of uncertainty: the type of user (malicious or regular) involved in the interactions and the behaviour of malicious users (attack or no attack). If the microservice system decides to observe an interaction from Alice, which represents an ordinary interaction with the system, the data collected is irrelevant to support digital investigations. Otherwise, if the system decides to observe an interaction from Bob, the data collected could be relevant for future digital investigations. There is a cost (networking, storage, or overhead) associated with each action performed by the system that our approach aims to reduce.

Bob is aware that the surveillance application consists of a collection of microservices, but he is unsure about the specific type of microservice with which he is interacting. Additionally, he is unsure about the observability techniques employed by the microservice system or whether the system has chosen not to observe his interaction. In his efforts to compromise the system availability, Bob can employ two types of denial-of-service attacks: distributed denial-of-service (DDoS) and application-layer DDoS. The former proves effective in targeting infrastructure microservices by exhausting system resources, while the latter is effective against business microservices as it causes the system to attack itself, leading to a cascade of system failures [20].

Lastly, Bob may change the type of attack employed depending on his perception of the impact against a specific type of microservice. In addition, Bob prefers to maintain his privacy and avoiding the likelihood of being traced back in case of a digital forensic investigation. Thus, he prefers to attack the system when his interactions are ignored by the system. In summary, Bob aims to maximise the impact of the attack while minimise the possibility of privacy loss.

Existing approaches for implementing forensic-ready systems [10], [11] may not be effective in scenarios where there is no prior knowledge of security incidents. To collect digital evidence, we can use existing observability approaches, but they may not meet forensic readiness requirements such as relevance, completeness, and minimality. For example, full observability can collect all necessary digital evidence, but may also collect irrelevant data, compromising the relevance and minimality requirements if a regular user interacts with the system. Sampling approaches such as fixed or adaptive may collect relevant data, but they may not satisfy the forensic readiness requirements, as the sampling process can occur during interactions of regular users.

Our approach models uncertainties from the perspec-

tives of users and microservices to reason about their interactions and determine when and how to observe microservices. In addition, our approach uses collected data to improve and adapt the proposed game model, enhancing the forensic-ready capabilities of microservice systems. Game theory provides an effective solution for this work because it allows the representation and analysis of strategic interactions of self-interest players (users and microservices) prior to the occurrence of security incidents. In such interactions, the best action of a player depends on expectations about what other players will do. Self-interest means that players have preferences over the outcomes of their interactions, and they would behave to maximise them.

## 3 Approach: Adaptive Observability

Adaptive observability provides mechanisms for instrumenting, collecting, and preserving observability data such as metrics, logs, and traces. Figure 1 shows the architecture of our approach with two phases: *design time* and *runtime*.

During the *design time* phase, software engineers must specify interactions between users and microservices using the proposed game model. This specification includes defining the types of users and microservices, available actions, and uncertainties. Additionally, software engineers must provide model parameters to instantiate the game model. These parameters represent the possible impacts and costs of attacks constrained by the preference specifications of malicious users and forensic-ready microservices. The model defined in this phase is used in the runtime phase.

During the *runtime* phase, the approach uses a MAPE-based feedback loop [21] to provide adaptations to the microservice system. Such adaptations are strategies for forensic-ready microservices in response to potential interactions of malicious users. As shown in the figure, in the runtime phase the architecture is divided into *managing system* and *managed system*. In the following, we explain the components of this phase.

*Managing System*. This system is organised into strategic reasoning and MAPE-based components. The strategic reasoning components support the steps in the feedback loop. It consists of four components, namely: (i) **game solver**: responsible for computing the game solution as a set of strategies to be executed by forensic-ready microservices (ii) **runtime model**: responsible for maintaining updates on the game model at runtime, including player types, available actions, and uncertainties; (iii) **learning model**: responsible for maintaining the histories of the game and learning rules to formulate future strategies; and (iv) **strategy builder**: responsible for building strategies for forensic-ready microservices. Such strategies are descriptions of when and how to observe and collect observability data from microservices.

In the MAPE-based components, the **Monitor** provides an *API* for the instrumentation of microservices and *sensors* to collect data from the managed system. The collected data is used to update the runtime model and learning model components. The **Analyser** component performs game-theoretic reasoning using the game solver component and computes strategies used by the managed system. The **Planner** component synthesises strategies for forensic-ready microservice using the strategy builder component. The **Executor** performs the microservice strategies in the managed system. This process is executed by *actuators* distributed among the microservices of the managed system.

*Managed system*. A microservice system interacts with users to realise business functionalities and generates outputs (metrics, logs, and traces) stored in a datastore.

In this paper, we focus on the descriptions of the preference specifications and the game-theoretical model for adaptive observability. A complete description of the adaptive observability approach, including the learning and strategy builder components, will be addressed in future work.

## 4 Preference Specifications

The representations of malicious users and forensic-ready microservices are essential for determining how they should make decisions during interactions. To formalise these representations, we use the concept of *preferences* from rational choice theory [22]. This theory consists of three elements: (i) a set of actions; (ii) a set of outcomes; and (iii) a specification of preferences over outcomes. To illustrate this theory, consider the case of Bob, a malicious user trying to compromise a microservice-based system. Bob can perform two types of attacks (actions), resulting in different consequences (outcomes), but he prefers to act when the system ignores his interactions. In this scenario, we can define Bob's preference as preserving privacy (preferences over outcome).

Preference is the order that an agent gives to a set of outcomes [23]. Formally, preferences are specified by binary relations to express *strict* ($\succ$), *indifferent* ($\sim$), and *weak* ($\succeq$) preference relations. For example, let $O$ be a finite set of outcomes, for any $o_1, o_2 \in O$, $o_1 \succ o_2$ when an agent strictly prefers $o_1$ to $o_2$; $o_1 \sim o_2$ when an agent is indifferent between $o_1$ and $o_2$; and $o_1 \succeq o_2$ when an agent weakly prefers (is indifferent or prefers) $o_1$ to $o_2$.

### 4.1 Preferences of malicious users

Previous research in the fields of security and privacy has used concepts from economics and game theory to represent attackers [24], [25]. However, these studies have primarily focused on specific attack-defence scenarios, and how to formalise the attacks using game models. Examples are the zero-sum games [26], repeated games [27], and stochastic games [28]. To provide a more generalised framework for understanding attacker behaviour, researchers have proposed an incentive-based model of attacker intent, objectives, and strategies (AIOS) [29]. In this model, attackers aim to maximise their incentives and minimise the costs associated with their actions.

We use the AIOS model to formalise the behaviour of malicious users in our work. This model is suitable for the proposed adaptive observability because it allows us to consider the incentives and costs of users without relying on specific attack scenarios. Under this model, we consider that (i) the incentive for malicious users is to impact the confidentiality, integrity, or availability (CIA) of a system; and (ii) the cost is the possibility of potential privacy loss resulting from a particular outcome. Based on these assumptions, we formulate a set of outcomes from the perspective of malicious users described as follows:

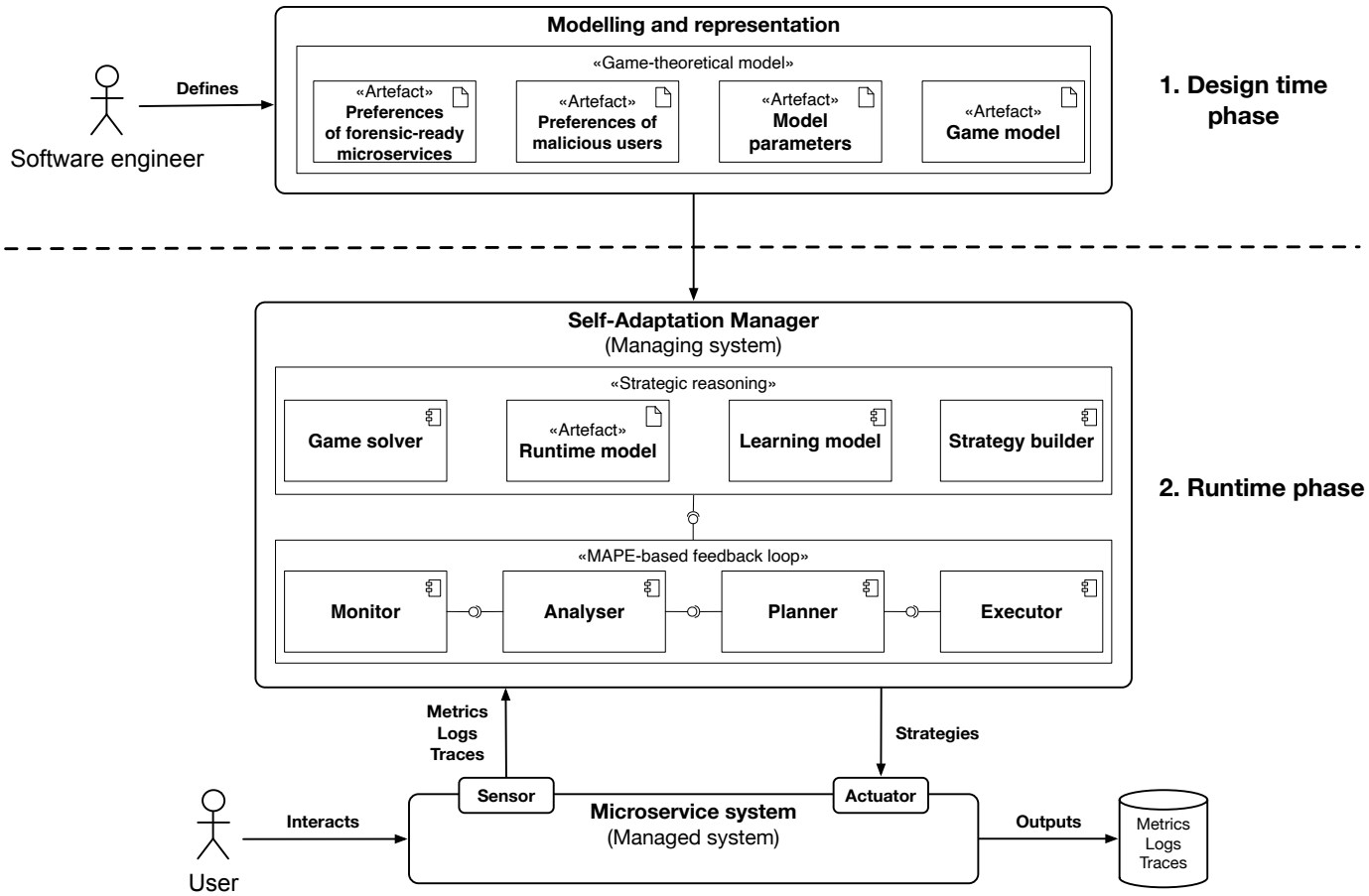


Fig. 1. An architectural overview of the adaptive observability approach in a microservice-based system.

$o_1$: **successful attack and potential privacy loss** - impact on CIA and data is preserved by the system.
$o_2$: **successful attack and no privacy loss** - impact on CIA and no data is preserved by the system.
$o_3$: **unsuccessful attack and potential privacy loss** - no impact on CIA and data preserved by the system.
$o_4$: **unsuccessful attack and no privacy loss** - no impact on CIA and no data is preserved by the system.

After specifying the outcomes, we perform pairwise comparisons to establish a preference ordering for malicious users. To perform these comparisons, we use the preference ranking method [30], which defines preferences over $n$ outcomes in a $n \times n$ matrix $A = (a_{ij})$, where:

$$a_{ij} = \begin{cases} 1 & \text{if outcome } i \text{ is preferred to } j \\ \text{-1} & \text{if outcome } j \text{ is preferred to } i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

In the pairwise comparisons, we assume that malicious users prefer to maintain their privacy over causing an impact on the CIA of microservice systems. In other words, the most important aspect for malicious users is to preserve their privacy, which can be represented as $o_2 \succeq o_1$ and $o_2 \succeq o_3$ and $o_4 \succeq o_3$ and $o_4 \succeq o_1$. The most preferred outcome is $o_2$, which represents a successful attack where the microservice system ignores the interactions from malicious users. This can be expressed as $o_2 \succeq o_1$, $o_2 \succeq o_3$, and $o_2 \succeq o_4$. In contrast, the least preferred outcome is $o_3$, which denotes an unsuccessful attack where the microservice system collects data from the interactions of malicious users. Therefore, $o_1 \succeq o_3$, $o_2 \succeq o_3$, and $o_4 \succeq o_3$. Matrix 2 presents the results of these comparisons.

$$\mathbf{A_1} = \begin{array}{c} \begin{array}{cccc} o_1 & o_2 & o_3 & o_4 \end{array} \\ \begin{bmatrix} 0 & \text{-1} & 1 & \text{-1} \\ 1 & 0 & 1 & 1 \\ \text{-1} & \text{-1} & 0 & \text{-1} \\ 1 & \text{-1} & 1 & 0 \end{bmatrix} \end{array} \begin{array}{c} \\ o_1 \\ o_2 \\ o_3 \\ o_4 \end{array} \tag{2}$$

Based on the results of Matrix 2, we can establish a preference order that ranks the outcomes from most to least preferable in an ordinal manner. This order provides foundations to analyse how malicious users should behave when interacting with forensic-ready microservices. The preference ordering of malicious users is defined as follows:

$$o_2 \succeq o_4 \succeq o_1 \succeq o_3 \tag{3}$$

### 4.2 Preferences of forensic-ready microservices

Digital forensic readiness in the context of software engineering is a property that encapsulates the capabilities of the software to collect digital evidence before the occurrence of security incidents. In this view, Pasquale et al. [18] propose a set of requirements for forensic-ready systems such as availability, relevance, minimality, traceability,

completeness, non-repudiation, data provenance, and legal compliance. In the work in this paper, we focus on the following requirements:

- **Relevance (REQ-1)**: preserved data should be essential to support a digital forensic investigation and explain how security incidents occurred;
- **Completeness (REQ-2)**: preserved data should have all the necessary or appropriate elements to support a digital forensic investigation;
- **Minimality (REQ-3)**: preserved data should be minimal and not include any information that is unnecessary for an investigation.

A forensic-ready system needs to decide if data is relevant or non-relevant. This task is similar to information retrieval and binary classification problems, in which a model or a classifier, produces a discrete class label (positive or negative), indicating the predicted class of an instance [31]. However, the predicted class may be different from the actual class. Thus, additional labels (true and false) need to be considered. Given a model and an instance, there are four possible outcomes: (i) **true positive**: positive instances correctly classified as positive; (ii) **false positive**: negative instances incorrectly classified as positive; (iii) **true negative**: negative instances correctly classified as negative; and (iv) **false negative**: positive instances incorrectly classified as negative. We map this idea from binary classification to formulate a set of outcomes from the perspective of forensic-ready microservices, described as follows:

$tp$: **true positive** - relevant data preserved; e.g., data collected from the interactions of malicious users;
$fp$: **false positive** - non-relevant data preserved; e.g., data collected from the interactions of regular users;
$tn$: **true negative** - non-relevant data not preserved; e.g., data not collected from the interactions of regular users;
$fn$: **false negative** - relevant data not preserved; e.g., data not collected from the interactions of malicious users.

For each pair of outcomes, we use the forensic readiness requirements to assess the most preferable outcome. In addition, we made two assumptions in these comparisons: (i) collecting relevant data is preferred to ignoring irrelevant data (i.e., $tp \succeq tn$), and (ii) collecting irrelevant data is preferred to ignoring relevant data (i.e., $fp \succeq fn$).

- For REQ-1 and REQ-3, the system should preserve only relevant data, discarding unnecessary information ($tp \succeq fp$, $tp \succeq tn$, and $tp \succeq fn$).
- For REQ-2, the system should preserve all the necessary or appropriate data ( $tp \succeq fp$, $tn \succeq fp$, and $fp \succeq fn$).
- For RQ-3, the system should not preserve irrelevant data ($tp \succeq tn$, $tn \succeq fp$, and $tn \succeq fn$).

The outcome $fn$ violates all the forensic readiness requirements; for this reason, $tp \succeq fn$, $fp \succeq fn$, and $tn \succeq fn$. Matrix 4 and preference ordering 5 exhibit the results of pairwise comparisons and preferences of forensic-ready microservices, respectively.

$$\mathbf{A_2} = \begin{array}{c} \begin{array}{cccc} tp & fp & tn & fn \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ -1 & 0 & -1 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 0 \end{bmatrix} \end{array} \begin{array}{c} \\ tp \\ fp \\ tn \\ fn \end{array} \tag{4}$$

$$tp \succeq tn \succeq fp \succeq fn \tag{5}$$

The preference orderings presented in this section are utilised as constraints in the parameterisation of the proposed game model. This process instantiates the game model to express the impacts and costs of potential attacks. These constraints are necessary to ensure that the specified orders of preference in Formulas 3 and 5 are maintained.

## 5 Adaptive Observability Game

The fundamental assumption of game theory is that all agents (or players) have common knowledge. However, this assumption restricts the representation of the scenario described in Section 2, as Bob and the microservice system have uncertainties regarding each other. Therefore, we use Bayesian games to formalise our game-theoretical model. Bayesian games, or games of incomplete information, are models of interactive decision situations in which the game's elements (e.g., players, actions, or preferences) are not common knowledge.

### 5.1 Game-theoretical model

We formalise a game-theoretical model for forensic-ready microservice systems addressing the uncertainties from the perspectives of users and microservices. We consider that all users and microservices are rational players in the game; i.e., they are assumed to maximise their utility. The uncertainties of a player are captured by the notion of an epistemic type, which describes a player's private knowledge about other players [32]. Formally, we define the adaptive observability game as a tuple $G = \langle N, A, \Theta, p, u \rangle$, where:

- $N = \{1, \cdots, n\}$ is a finite set of players in which $i$ represents the index for the players, and we use $-i$ to denote all the other players except $i$;
- $A = \{A_1, \cdots, A_n\}$ is a finite set of actions, where $A_i$ is the set of actions available for player $i$;
- $\Theta = \{\Theta_1, \cdots, \Theta_n\}$ is a finite set of types, where $\Theta_i$ is the type space of player $i$;
- $p : \Theta \mapsto [0, 1]$ is the common prior over types;
- $u = \{u_1, \cdots, u_n\}$ is a finite set of utility functions, where $u_i : A \times \Theta \mapsto \mathbb{R}$ is the function for player $i$.

*Example*. Consider a simplified version of the motivating example described in Section 2. There are two players, $N = \{1, 2\}$, in which player 1 represents the users and player 2 represents the microservices. For each player, there are two types: $\Theta_1 = \{\theta_{1,1}, \theta_{1,2}\}$ is the type space of player 1, and $\Theta_2 = \{\theta_{2,1}, \theta_{2,2}\}$ is the type space of player 2. Type $\theta_{1,1}$ represents the malicious user (Bob), and type $\theta_{1,2}$ represents the regular user (Alice). Type $\theta_{2,1}$ represents the infrastructure microservices, and $\theta_{2,1}$ represents the business microservices. We denote $A_1$ as player 1's set of actions formed of A (DDoS attack) and N (no attack), and $A_2$ as player 2's set of actions composed of L (log) and I (ignore). The probability distribution on these types is as follows: (i) $p$ is the probability that player 1's type is $\theta_{1,1}$, and $1 - p$ is the probability that player 1's type is $\theta_{1,2}$; (ii) $q$ is the probability that player 2's type is $\theta_{2,1}$, and $1 - q$ is the probability that player 2's type is $\theta_{2,2}$.

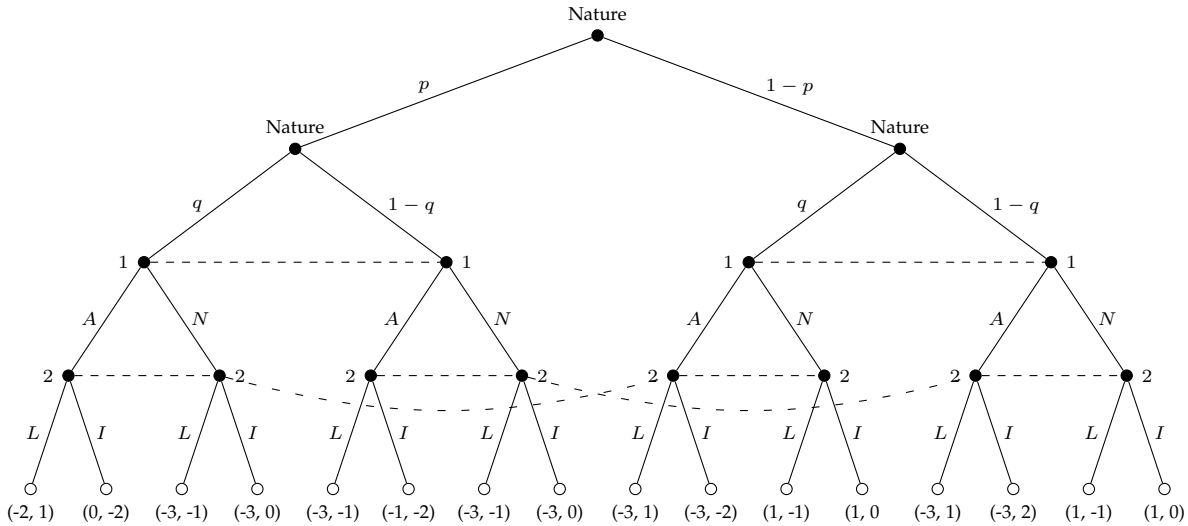


Fig. 2. The game tree of a simplified version of the motivating example, containing two players, four types, and four actions.

Figure 2 shows the game tree that represents the model described above. In a game tree, each node represents a player, the edge represents a possible action, and the leaves denote the outcomes of utility functions. The Nature node is a player who has no utility function; instead, this node provides exogenous randomisation to decide players' types. Nodes 1 and 2 represent players 1 and 2, respectively. The dashed lines connecting the nodes denote four information sets. Specifically, an *information set* is a set of decision nodes that a player cannot distinguish when making a decision [33]. For instance, Bob is uncertain whether he is interacting with a business or infrastructure microservice. This uncertainty is captured by the information set.

### 5.2 Strategies and utility functions

Each player (users or microservices) can choose an action deterministically or probabilistically. A *pure strategy* is a deterministic choice of action for player $i$ that maps from every type of $i$ to the action. On the other hand, a *mixed strategy* is a probabilistic choice of action, defined as a probability distribution over pure strategies.

The AIOS model encompasses concepts such as attacker intent, objectives, and strategies. Attacker intent and objectives are used to specify the preferences over outcomes, as described in Section 4.1. In this section, we focus on defining the strategies for malicious users. For example, Bob aims to compromise the system availability, as described in Section 2. To achieve this goal, Bob can use two attacks: DDoS and application-layer DDoS. We incorporate these attacks as strategies for malicious users in the game model, denoting user $i$'s strategy $s_i \in S_1$, where $S_1$ is the set of (pure and mixed) strategies for player 1.

For instance, the edges of node 1 in Figure 2 represent the actions available to player 1 (users): A (attack) and N (no attack). In this context, a pure strategy for player 1 is the choice to play A or N, while a mixed strategy for player 1 is a choice to play A with a probability of 1/2 and N with a probability of 1/2.

The process of collecting evidence in microservice systems is accomplished through observability instrumentation techniques, including metrics, logs, and distributed tracing. We model these techniques as strategies for forensic-ready microservices. Formally, we denote microservice $i$'s strategy as $s_i \in S_2$, where $S_2$ is the set of (pure and mixed) strategies for player 2. For example, the actions available to player 2 (microservices) represented as the edges of node 2 in Figure 2 are L (log) and I (ignore). An example of a pure strategy for player 2 is the choice to play L or I. On the other hand, a mixed strategy is the choice to play L with a probability of 2/3 and to play I with a probability of 1/3.

In game theory, players choose strategies based on their preferences over outcomes and those of the other players. Preferences can be represented by the concept of *utility function*, which measures players' level of satisfaction with given outcomes. Formally, a utility function for player $i$ is a real-valued function $u_i : O \mapsto \mathbb{R}$, where $O$ is a set of outcomes. The results of utility functions are called utilities or payoffs, where larger numbers describe better outcomes and smaller numbers denote less favourable outcomes. The leaves in Figure 2 represent utilities. In the remainder of this section, we define the utility functions for malicious users and forensic-ready microservices. These utility functions are numerical representations of the preferences specified in Section 4.1 and 4.2. We use the model parameters to calculate the results of these utility functions and allow adjustment of these functions based on the preferences of malicious users and forensic-ready microservices.

**Utility function for malicious users.** For malicious user $i$, given the preferences described in Section 4.1, the $i$'s utility function is defined as follows:

$$u_i(s_i, s_{-i}, \theta_{-i}) \stackrel{\text{def}}{=} i_i(s_i, \theta_{-i}) - c_i(s_i, s_{-i}), \tag{6}$$

where function $i_i(s_i, \theta_{-i})$ captures the impact metrics of the CIA (confidentiality, integrity, and availability) triad on microservices of type $\theta_{-i}$, defined as:

$$i_i(s_i, \theta_{-i}) \stackrel{\text{def}}{=} \sum_{\{i \in N | i=2, \Theta_2 = \theta_{-i}\}} [I_C(s_i, i) + I_I(s_i, i) + I_A(s_i, i)], \tag{7}$$

where functions $I_C(s_i, i)$, $I_I(s_i, i)$, and $I_A(s_i, i)$ denote the impact of confidentiality, integrity, and availability, respectively, on all microservices of type $\theta_{-i}$, given that malicious user $i$ chosen to play strategy $s_i$.

These functions are measured using the Common Vulnerability Scoring System (CVSS),[1] which assigns a score from 0 (no impact) to 10 (critical impact) for each impact metric. We refer the reader to the official specification of CVSS for a complete description of the impact metrics.[2] We used CVSS in the definition of utility functions because it captures the effects of a successful attack from the interactions described in this work. Furthermore, CVSS is used worldwide as a primary source for assessing the severity of computer system security vulnerabilities [34]. We do not consider the correctness and accuracy of CVSS scores.

The cost of strategies employed by malicious users depends on the probability of privacy loss. More specifically, the function $c_i(s_i, s_{-i})$ represents the privacy cost of user $i$, which is modelled as $\varepsilon v_i$, where $\varepsilon \in [0, 1]$ is the probability of privacy loss resulting from a particular outcome, and $v_i \in [1, \infty)$ is user $i$'s private valuation of privacy [35]. To ensure that this utility function represents the preference relations for malicious users described in Section 4.1, the privacy cost $\varepsilon v_i$ must be greater than $i_i(s_i, \theta_{-i})$. Therefore, we use the preference ordering for malicious users as constraints to tune the parameters of the model.

**Utility function for forensic-ready microservices.** Considering the preference relations specified in Section 4.2, forensic-ready microservice $i$'s utility function is defined as:

$$u_i(s_i, s_{-i}) \stackrel{\text{def}}{=} r_i(s_i, s_{-i})e_i - c_i(s_i), \tag{8}$$

where $r_i(s_i, s_{-i})$ is the relevance function which expresses the importance of data collected by microservice's strategy $s_i$ from user's strategy $s_{-i}$. This is a discrete function that returns 1 if the data collected is relevant, and 0 if the data preserved is non-relevant, expressed as:

$$r_i(s_i, s_{-i}) = \begin{cases} 1 & \text{if } s_i \text{ preserves relevant data from } s_{-i} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where parameter $e_i \in [1, \infty)$ represents the scale of importance of potential digital evidence, specified as *probative value*. In particular, probative value is defined as the degree of significance of digital evidence used to decide whether the evidence supports a particular hypothesis in a legal case [36]. Formally, let $E_i$ denote the set of probative values for microservice $i$. We use the notation $e_i \in E_i$ to denote the probative value which microservice $i$ declares to define the utility function $u_i(s_i, s_{-i})$. Finally, the cost function $c_i(s_i)$ represents the cost of microservice $i$'s strategy $s_i$ (e.g., measure, log, or trace). This cost can be estimated based on measures of monitoring, evidence collection, storage, and network usage.

### 5.3 Expected utility function

In the motivating example, we describe a scenario where Bob (malicious user) knows his type, but does not know the types of microservices. Similarly, microservices (business and infrastructure) have information about their types but do not have information about the types of users (malicious and regular). To reason about this setting, we need to consider the expected value of utility functions relative to the probability of other players' types and actions. This reasoning is captured by the concept of *expected utility function*. We use the notation $\mathbb{E}_i$ to denote player $i'$ expected utility function. Furthermore, a pair of strategies of players $i$ and $-i$ is called a strategy profile, written as $s = (s_i, s_{-i}) \in S$, where $S$ is the Cartesian product of $S_1$ (set of strategies for player 1) and $S_2$ (set of strategies for player 2). In the adaptive observability game, the expected utility function $\mathbb{E}_i$ of player $i$ for playing the strategy profile $s = (s_i, s_{-i}) \in S$ with type $\theta_i$ is defined as follows:

$$\mathbb{E}_i(s, \theta_i) = \sum_{\theta_{-i} \in \Theta_{-i}} p(\theta_{-i} \mid \theta_i)\mathbb{E}_i(s, (\theta_{-i}, \theta_i)) \tag{10}$$

In other words, the expected utility function depends on (i) the probability of other players having types $-\theta$, given that player $i$ has type $\theta$, expressed by the conditional probability $p(\theta_{-i} \mid \theta_i)$; and (ii) the expected utility function $\mathbb{E}_i(s, \theta_{-i}, \theta_i)$ which is detailed as follows:

$$\mathbb{E}_i(s, (\theta_{-i}, \theta_i)) = \sum_{a \in A} \left( \prod_{j \in N} s_j(a_j \mid \theta_j) \right) u_i(a, \theta_{-i}, \theta_i) \tag{11}$$

The expected utility function $\mathbb{E}_i(s, (\theta_{-i}, \theta_i))$ depends on (i) the probability under mixed strategy $s_j$ that player $j$ plays action $a_j$, given that $j$'s type is $\theta_j$, expressed by the notation $s_j(a_j \mid \theta_j)$; and (ii) the evaluation of $i$'s utility function given action $a$, other player's type $\theta_{-i}$, and $i$'s type $\theta_i$, written as $u_i(a, \theta_{-i}, \theta_i)$.

### 5.4 Equilibrium analysis

Rational players are assumed to maximise their expected utility, and their decisions are guided by the principle of the *best response*. Specifically, a player formulates the best responses that generate more expected utility, given the strategies of the other players. For instance, suppose that player 2 (microservice) knows that player 1 (user) will choose N (no attack). In this case, player 2 will choose I (ignore) because that is the best response to N. Formally, player $i$'s best response to strategy profile $s_{-i}$ is defined as:

$$BR_i(s_{-i}) = \arg\max_{s'_i \in S_i} \mathbb{E}_i(s'_i, s_{-i} \mid \theta_i). \tag{12}$$

Given the definition of best response, the Nash equilibrium (NE) [37] of a game is a strategy profile $s = (s_1, ..., s_n)$ where, for all players $i$, $s_i$ is the best response to $s_{-1}$; i.e., no player has incentives to change the strategy, and each player is playing the best response to the strategies of the other players. In the context of the adaptive observability game, the equilibrium concept of interest is Bayesian Nash equilibrium (BNE) [32], a refinement of NE. Formally, a strategy profile $s = (s_1, ..., s_n)$ is BNE if $s_i$ is a best response to $s_{-i}$ for every player $i \in N$. We use Gambit software

1. https://www.first.org/cvss
2. https://www.first.org/cvss/specification-document

tools for game theory to obtain the BNE, and simulate the behaviour of players in an equilibrium setting.[3]

## 6 Adaptive Observability in Action

We describe how to support the adaptive observability approach to be used in existing microservice-based applications. The implementation of adaptive observability in microservices systems relies on having instrumented microservices. However, existing observability solutions lack standardisation in the code instrumentation of the APIs. As a result, the use of observability solutions requires re-instrumenting the code of microservices. To overcome this challenge, the cloud community has developed an observability framework (OpenTelemetry), which provides a unified and vendor-agnostic way of instrumenting, collecting, and exporting observability data across different platforms.[4]

We use Micrometer as an abstraction layer over OpenTelemetry to instantiate the concepts proposed in adaptive observability. In particular, Micrometer provides a facade over the instrumentation APIs of observability solutions, thereby allowing software engineers to instrument JVM-based applications using a single API.[5] This API, called Observation API, defines concepts such as meters, counters, gauges, timers, and tracing. The implementation of these concepts is determined at runtime by the observability solution added to the application *classpath*.

We extended the Observation API to enable adaptive observability in existing microservice-based applications. Specifically, we created two Java annotations, namely `@EnableAdaptiveObservability` and `@MicroserviceType`, which must be added to the application's main class. The former annotation enables adaptive observability in a microservice by creating necessary configurations for executing microservice strategies. The latter annotation specifies the type of microservice based on the model of the adaptive observability game. Listing 1 shows an example of a class using these annotations. Therefore, to enable adaptive observability in an existing microservice-based application, only two lines of code, one for each annotation, need to be added to each microservice.

```
@EnableAdaptiveObservability
@MicroserviceType(value = "infra")
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Listing 1. `@EnableAdaptiveObservability` and `@MicroserviceType`

The code instrumentation in microservices is performed by implementations of the Observation interface. An observation, in this context, refers to the act of viewing and recording a fact or occurrence for further analysis. To perform an observation, we need an implementation of ObservationRegistry that we provided through the auto-configuration of `@EnableAdaptiveObservability`. Our implementation of ObservationRegistry comprises instances of ObservationHandler, responsible for carrying out the microservice strategies prescribed by the game-theoretical model. Listing 2 shows an example, where an observation is created and initiated in line two, followed by the opening of the observation's scope using a try-with-resources pattern. Events are generated within the scope to signal various aspects of the user creation process.

```
1  public User createUser(UserRequest userRequest) {
2      Observation obs = Observation.start(...);
3      try (Scope scope = obs.openScope()) {
4          obs.event(Event.of("Start of user creation"));
5          return userService.save(userRequest);
6      } catch (Exception exception) {
7          obs.event(Event.of("Error in user creation"));
8          obs.error(exception);
9          throw exception;
10     } finally {
11         obs.event(Event.of("End of user creation"));
12         obs.stop();
13     }
14 }
```

Listing 2. Instrumentation using the Observation API

In addition to code instrumentation, a new microservice is required to manage the game model, perform strategic reasoning, and dispatch strategies to other microservices. To enable auto-configuration on this new microservice, we created `@EnableAdaptiveObservabilityServer`. Listing 3 provides an example of a class using the server annotation.

```
@EnableAdaptiveObservabilityServer
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Listing 3. `@EnableAdaptiveObservabilityServer`

To use a different observability approach (e.g., full and sampling), it is necessary to remove the configuration for adaptive observability described in this section and add a property to a configuration file. This property represents the probability of sampling and can take a value between 0.0 and 1.0. In terms of code instrumentation, no modifications need to be made, as long as the Observation API is used. Thus, the separation of the instrumentation API from the observability solutions helps to reduce coupling between a microservice and any particular observability framework. However, the choice of an observability approach impacts the evidence collection process in microservices systems; specifically, there are implications for the amount of data collected and its relevance. For this reason, there is a need for assessments of this impact to serve as recommendations to software engineers in choosing an observability approach.

3. http://www.gambit-project.org
4. https://opentelemetry.io
5. https://micrometer.io

# 7 EVALUATION

The work was evaluated in terms of a comparison of the performance of the adaptive observability approach with the full and sampling observability. To achieve this goal, we designed and conducted an experiment to investigate the impact of different observability approaches on the evidence collection process in microservice-based applications. We selected an open-source benchmark microservice system called TrainTicket [38], which consists of 41 microservices adhering to microservice design principles and reflecting industrial practices. We chose this system due to its complexity and size, which represents a challenging scenario for the evaluation of our approach.

We assessed the performance of the observability approaches using metrics such as precision, recall, and F-measure. Such metrics are used for evaluating the performance of information retrieval methods and binary classifiers [31], [39]. In our context, these metrics are appropriate for quantitatively measuring whether forensic readiness requirements (relevance, completeness, and minimality) have been satisfied. For example, precision measures how much data preserved is relevant; i.e., an observability approach that collects only essential data has a precision of 1.0 (perfect precision) and satisfies the requirements of relevance and minimality. Recall measures the percent of all possible relevant data preserved; i.e., an observability approach that collects all the necessary data has a recall of 1.0 (perfect recall) and satisfies the requirement completeness.

Ideally, an observability approach would have both perfect precision and perfect recall. Thus, we decided to use F-measure as a criterion to perform the comparisons since it is a single metric that takes into account both precision and recall. In particular, F-measure is calculated by taking the harmonic mean of recall and precision. The formulas to compute precision, recall, and F-measure are defined, respectively, as follows.

$$Precision = \frac{tp}{tp + fp} \tag{13}$$

$$Recall = \frac{tp}{tp + fn} \tag{14}$$

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{15}$$

## 7.1 Research questions

The evaluation is based on the following research question:

**RQ1:** What is the performance of the observability approaches in terms of precision, recall, and F-measure?
***Rationale**: This research question aims to evaluate the performance of observability approaches under various scenarios of uncertainty related to the type of users (e.g., regular and malicious) and varying the number of simultaneous users interacting with the microservice-based system.*

**RQ2:** What is the best observability approach in terms of F-measure to perform evidence collection in microservice-based systems?
***Rationale**: This research question aims to determine which observability approach performs better in terms of F-measure and to provide recommendations to software engineers in choosing an observability approach.*

## 7.2 Hypotheses

As part of the experiment, we formulated and tested the following hypotheses:

$\boldsymbol{H_{01}}$ There is no difference in performance (measured as F-measure) between the observability approaches. i.e., $H_{01} : F_1(full) = F_1(sampling) = F_1(adaptive)$.

$\boldsymbol{H_{11}}$ There is difference in performance (measured as F-measure) between the observability approaches. i.e., $H_{11} : F_1(full) \neq F_1(sampling) \neq F_1(adaptive)$.

## 7.3 Variables

Variables in experiments are divided into two categories: independent and dependent. The former refers to those variables that we can control and modify in the experiment. The latter refers to those variables that we want to study to see the effect of the modifications in the independent variables. In this experiment, the independent variables are the uncertainty of the type of user and the number of simultaneous users interacting with the microservice-based system. The dependent variable is the performance metric F-measure defined in Formula 15.

## 7.4 Experimental design

To conduct the experiment, we simulated different types of users interacting with the system using the following strategies: no attack, attack 1, and attack 2. In the first strategy, no attack represents a use case for booking train tickets. In the second strategy, attack 1 represents a DDoS attack targeting infrastructure microservices. In the third strategy, attack 2 represents a SQL injection attack on business microservices. To assess the impact of attacks 1 and 2, we use CVSS as the primary source. The summary of the users' strategies containing their respective impacts on the system is presented in Table 1.

We modelled the strategies as a set of execution scenarios using an open-source load testing tool, named Gating.[6] Each execution scenario, representing the interaction between users and the TrainTicket system, is described in Table 2. The execution scenarios describe situations of different levels of uncertainty about the type of user, ranging from 10% to 90% of malicious users. The number of simultaneous users interacting with the system are $10^2$, $10^3$, $10^4$, and $10^5$. The uncertainty about the type of microservice is represented as 50% infrastructure and 50% business. Finally, we apply all the observability approaches (full, sampling, and adaptive) to perform the process of evidence collection during the simulation.

Each execution scenario is repeated 30 times to ensure statistical significance of the performance metrics (e.g., precision, recall, and F-measure) with a confidence interval between 95% and 99%, as recommended by Arcuri and Briand [40]. The result of the evidence collection performed by the observability approaches is stored in three different datasets: $G_1$: data collected by full observability; $G_2$: data collected by sampling; and $G_3$: data collected by adaptive observability.

We created a gold standard dataset to establish a common ground to assess the performance of the observability

6. https://gatling.io

TABLE 1
Summary of user strategies and their impact on the CIA.

| Strategy | Description | Attack vector | Impact metrics | | | CVE-ID |
|---|---|---|---|---|---|---|
| | | | Confidentiality | Integrity | Availability | |
| No attack | Booking train tickets | - | None (I:N) | None (A:H) | None (A:H) | - |
| Attack 1 | DDoS attack | Infrastructure microservice | None (C:N) | None (I:N) | High (A:H) | 2022-1728 |
| Attack 2 | SQL injection | Business microservice | Low (C:L) | Low (I:L) | None (A:N) | 2013-0375 |

TABLE 2
Set of execution scenarios and their proportion of type of users and microservices, given a number of simultaneous users.

| | Type of user | | Type of microservice | | |
|---|---|---|---|---|---|
| | Malicious | Regular | Business | Infrastructure | Number of users |
| $s_1$ | 10% | 90% | 50% | 50% | $10^2, 10^3, 10^4, 10^5$ |
| $s_2$ | 20% | 80% | 50% | 50% | $10^2, 10^3, 10^4, 10^5$ |
| $s_3$ | 30% | 70% | 50% | 50% | $10^2, 10^3, 10^4, 10^5$ |
| $s_4$ | 40% | 60% | 50% | 50% | $10^2, 10^3, 10^4, 10^5$ |
| $s_5$ | 50% | 50% | 50% | 50% | $10^2, 10^3, 10^4, 10^5$ |
| $s_6$ | 60% | 40% | 50% | 50% | $10^2, 10^3, 10^4, 10^5$ |
| $s_7$ | 70% | 30% | 50% | 50% | $10^2, 10^3, 10^4, 10^5$ |
| $s_8$ | 80% | 20% | 50% | 50% | $10^2, 10^3, 10^4, 10^5$ |
| $s_9$ | 90% | 10% | 50% | 50% | $10^2, 10^3, 10^4, 10^5$ |

approaches. A gold standard is a benchmark created as a set of accurate annotations which provide perfect precision and recall. To create this dataset, we directly observed the data collected to verify the identity of the user and the strategy performed. We then compared the results of $G_1$, $G_2$, and $G_3$ to this gold standard dataset.

### 7.5 Results and hypothesis testing

The results presented in this section are part of a more comprehensive evaluation, and the supplementary material is available on GitHub.[7] In the following, we revisit and answer our research questions.

**RQ1:** What is the performance of the observability approaches in terms of precision, recall, and F-measure?

The results of the precision, recall, and F-measure of the observability approaches (full, sampling, and adaptive) are shown in Figure 3. Full observability provided a perfect recall (score of 1.0) for all execution scenarios. However, the precision was low (e.g., 0.03, 0.07, and 0.10) in scenarios with high uncertainty about malicious users (e.g., $s_1$, $s_2$, and $s_3$). On the other hand, in scenarios of low uncertainty (e.g., $s_7$, $s_8$, and $s_9$), the precision scores increase as there are more malicious users interacting with the system. This was expected, as the full observability approach collects all data during the evidence collection process.

Unlike full observability, the results for sampling and adaptive observability approaches shown in Figure 3 demonstrate that these approaches do not provide perfect recall. Instead, the recall score was approximately 0.5 on average, while the precision score tends to increase in scenarios with low uncertainty about malicious users ($s_7$, $s_8$, and $s_9$).

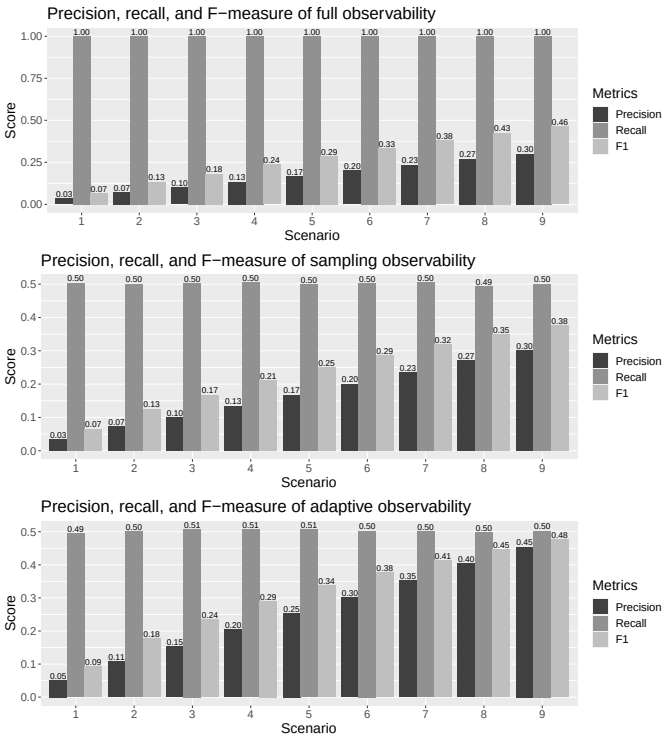


Fig. 3. Precision, recall, and F-measure of the observability approaches.

**RQ2:** What is the best observability approach in terms of F-measure to perform evidence collection in microservice-based systems?

To answer this research question, we use F-measure as a criterion to decide which observability approach performs better. However, F-measure may not be sufficient to thoroughly evaluate the performance difference between the observability approaches, as observed results may be affected by chance. To address this limitation, we decided to perform a statistical analysis to assess whether there is enough empirical evidence to claim that there is a difference between them. To conduct the statistical analysis and hypothesis testing, we use the Wilcoxon signed-rank test, which is a non-parametric test used to compare two paired samples when each individual in one sample also appears in the other sample [41], [42]. Specifically, we use this test because the data in $G_1$, $G_2$, and $G_3$ does not follow a normal distribution, violating the assumption for a parametric test.

7. https://github.com/davimonteiro/tsc-evaluation

TABLE 3
The F-measure of datasets $G_1$, $G_2$, and $G_3$ and their relative difference.

| | Datasets ($F_1$) | | | Relative difference | |
|---|---|---|---|---|---|
| | $G_1$ | $G_2$ | $G_3$ | $G_3$ vs $G_1$ | $G_3$ vs $G_2$ |
| $s_1$ | 0.0670 | 0.0653 | 0.0931 | 38.96% | 42.50% |
| $s_2$ | 0.1339 | 0.1254 | 0.1775 | 32.51% | 41.52% |
| $s_3$ | 0.1824 | 0.1675 | 0.2351 | 28.91% | 40.37% |
| $s_4$ | 0.2364 | 0.2121 | 0.2905 | 22.91% | 36.95% |
| $s_5$ | 0.2862 | 0.2510 | 0.3383 | 18.18% | 34.76% |
| $s_6$ | 0.3339 | 0.2868 | 0.3773 | 12.99% | 31.58% |
| $s_7$ | 0.3794 | 0.3202 | 0.4132 | 08.93% | 29.03% |
| $s_8$ | 0.4261 | 0.3493 | 0.4461 | 04.69% | 27.71% |
| $s_9$ | 0.4629 | 0.3758 | 0.4772 | 03.10% | 26.97% |

Our null hypothesis $H_{01}$ states that there is no difference in performance, measured as F-measure, between the observability approaches; i.e., we have no assumption about which approach is better in terms of F-measure. The results of the Wilcoxon test, reported in the supplementary material, indicate a statistically significant difference in F-measure between the observability approaches. Therefore, at the 5% significance level, we reject the null hypothesis $H_{01}$ and conclude that the approaches' performances are significantly different.

We used Vargha and Delaney effect size Â [43], a non-parametric effect size measure, to evaluate the magnitude of the performance difference. This type of effect size is recommended in software engineering when randomised algorithms are involved [40], [44], [45]. In this experiment, the Vargha and Delaney effect size $\hat{A}_{AB}$ measures the expected probability of observability approach A producing a higher F-measure score than approach B. This measure is useful to assess the practical significance of the observed difference in performance expressed as F-measure.

In the supplementary material, we present the results of Vargha and Delaney effect size comparing $G_3$ versus $G_1$, denoted as $\hat{A}_{31}$, and $G_3$ versus $G_2$, written as $\hat{A}_{32}$. The values of this effect size range from 0 to 1, in which a value of 0.5 indicates that two approaches are stochastically equivalent and values closer to 0 or 1 indicate a large stochastic difference between approaches. For example, in scenario $s_3$ for $10^3$ users $\hat{A}_{31}$ = 0.99 (large); i.e., we would obtain better results 99% of the time with adaptive observability.

Lastly, to compare the performance of the observability approaches in terms of F-measure, we calculated the relative difference between $G_3$ and $G_1$, as well as between $G_3$ and $G_2$. Table 3 presents the F-measure of datasets $G_1$, $G_2$, and $G_3$, along with their relative differences. The results indicate that the adaptive observability approach outperforms the other observability approaches in all execution scenarios. Specifically, adaptive observability is better than sampling observability (from 26.97% up to 42.50%) and full observability (from 3.1% up to 38.96%). When considering the uncertainty regarding malicious users, the performance of adaptive observability improves as the uncertainty of attacks occurring increases. In other words, adaptive observability performs better in scenarios with high uncertainty about malicious users (e.g., $s_1$, $s_2$, and $s_3$).

### 7.6 Threats to validity

Threats to validity can be classified into the following categories: conclusion, internal, construct, and external validity. We discuss the threats that potentially impacted our work and the ways in which we attempted to mitigate them.

**Conclusion validity** concerns the factors that affect the statistical analysis and the conclusion from relations between the variables of an experiment. In order to address this threat, we first verify whether the datasets analysed in this experiment follow a normal distribution via a histogram and the Shapiro-Wilk test [46]. Since we found that the normality assumption was violated, we decided to use a non-parametric test which does not rely on that assumption. Next, to mitigate the risk of the *fishing and the error rate* threat, we decided to use only one dependent variable as described in Section 7.3. Finally, this experiment is based on the assumption that analysed datasets are correctly labelled. To address threats to this assumption, we created a gold standard as described in Section 7.4.

**Internal validity** evaluates the influences or biases that affect the causality of the independent variables. We expressed different scenarios to simulate interactions between users and the system, varying the number of simultaneous users and the uncertainty about the type of user. However, we did not change the uncertainty of the type of microservice by fixing an even probability distribution (50% business and 50% infrastructure). The latter type of uncertainty may not reflect reality, and we may need to consider different levels of uncertainty. However, this would increase the number of possible execution scenarios, making this experiment infeasible.

**Construct validity** concerns generalising the results based on the concept or theory behind it. The metrics used to assess the performance of the observability approaches may not be sufficient to cover all aspects of the evidence collection process. Other factors such as cost and scalability may also be important to consider. Moreover, our study focused on a single microservice-based system, which may limit the generalisability of our findings.

**External validity** assesses the extent to which the findings can be generalised to other settings. The main concern is the choice of the benchmark system used in this experiment, as a single microservice-based system may limit the generalisability of our findings. Prior studies also used the same benchmark system [47]–[51].

## 8 Related Work

**Microservices and Observability.** The concept of observability originated in control theory and refers to the measure of how the internal states of a system can be determined from its external outputs [52], [53]. In microservice systems, observability involves a set of instrumentation techniques that helps developers reason about incidents in microservices [13], [14]. Existing studies use data collected by observability to automate debugging processes [54], [55], perform fault localisation [48], [56], and investigate the effectiveness of existing industrial debugging practices [47]. However, these studies assume that data, relevant or not, is available

for investigating incidents. Our work aims to use observability to perform evidence collection of relevant data before a security incident occurs, reducing the cost of observability.

**Digital forensic readiness.** The notion of digital forensic readiness was formulated in the context of organisations to support proactive digital investigations of software systems [7], [57], [58]. Existing research literature has initially focused on identifying guidelines [7]–[9] that can enhance organisations' capabilities to achieve digital forensic readiness [18]. In software engineering, however, little attention has been given to the challenge of how to implement software systems that are forensic-ready. Existing works that address this problem is ineffective for microservices because they introduce unnecessary system overhead. For example, there are approaches [59]–[62] that focus on proactive evidence collection of all potential digital evidence. However, this strategy consumes additional system resources (e.g., CPU and I/O) [63], especially in highly optimised microservices that tend to be latency sensitive [12]. Furthermore, excessive evidence collection can produce irrelevant data that hides the relevant digital evidence required to establish factual and reliable conclusions during digital forensic investigations [10], [63].

Researchers have proposed techniques and methods to perform evidence collection selectively. For example, Pasquale et al. [10] propose using attack scenarios [64] to modify evidence collection mechanisms for cloud infrastructures. Alrajeh et al. [11] introduce the concept of evidence preservation requirements to model known incidents in advance and synthesise evidence collection capabilities for forensic-ready systems. These studies, however, consider that security incidents and the attacker behaviour are known a priori. This means that if an attacker exploits an unknown vulnerability or uses other strategies to compromise software systems, the evidence collection mechanisms may not collect the digital evidence needed; i.e., loss of digital evidence. This scenario can be aggravated when there are uncertainties associated with the presence of attackers or in adaptations of their behaviour. For instance, attackers may modify their strategies to compromise software if they are ineffective to breach security capabilities [17].

**Game theory.** To address the uncertainties associated with potential security incidents and the adaptive nature of attacker behaviour, the security research community has explored game-theoretical models for analysing and planning strategies to mitigate security threats. Within this context, the interactions between attackers and defenders are modelled to respond to various types of attacks, including DDoS attacks [24], [65], stealthy attacks [66], bandwidth amplification attacks [67], and advanced persistent threats [17], [68]–[71].

Game theory provides an appropriate framework for reasoning about security incidents involving strategic decision-making by attackers and defenders when there is uncertainty about potential security incidents [17]. However, prior game theory-based approaches have not considered situations when the security mechanisms are insufficient to protect the system assets or when there is a need to design forensic-ready software systems. Furthermore, those approaches address two-player games (attacker and defender). However, in the context of microservices, other models are required to accommodate players, such as types of users (e.g., regular or malicious) and types of microservices (e.g., business and infrastructure services).

Our work differs from earlier studies in forensic-ready systems by addressing the uncertainty of security incidents. We model the interactions between malicious users and microservices as a game-theoretical model to decide when and how to observe microservice systems. In particular, we represent the uncertainties from both perspectives (attacker and microservices). As opposed to game-theoretical approaches in security, we consider the forensic readiness requirements (e.g., relevance, minimality, and completeness) to design forensic-ready microservice systems.

## 9 Conclusion

In this paper, we addressed the problem of implementing forensic-ready microservice systems considering uncertainties about security incidents. To illustrate this problem, we presented a motivating example that highlights interactions between users and microservices. This example provides context for the concepts we introduce in our approach. Our approach, named adaptive observability, focuses on providing evidence collection mechanisms for microservice systems. We primarily describe two aspects of our approach: representation and reasoning. The former involves formalising the representations of malicious users and forensic-ready microservices. The latter involves reasoning about these representations to decide when and how to observe the microservices before a security incident occurs.

The work was evaluated by using a benchmark microservice system. In the evaluation, we assessed the performance of adaptive observability and compared it to existing observability approaches. The results indicate that adaptive observability is better than sampling observability (from 26.97% up to 42.50%) and full observability (from 3.1% up to 38.96%). In particular, adaptive observability performs better in scenarios with high uncertainty about malicious users.

In future work, we intend to extend the capabilities of adaptive observability by incorporating adaptations of microservices strategies based on the risk preferences of malicious users. Specifically, we plan to investigate how different risk attitudes (e.g., risk averse, risk neutral, and risk seeking) may influence the behaviour of malicious users. In terms of evaluation, we intend to assess our approach using an industrial case study, addressing the limitations of the generalisability of results. Additionally, we recognise that performance metrics may not be sufficient for evaluating observability approaches. Therefore, we intend to consider other factors, such as cost and scalability, when assessing the effectiveness of adaptive observability.

## Acknowledgment

This work was supported in part by the Science Foundation Ireland grant 13/RC/2094_P2 and the UK EPSRC grant EP/R013144/1.

## REFERENCES


[1] J. Lewis and M. Fowler, "Microservices," 2014. [Online]. Available: http://www.martinfowler.com/articles/microservices.html

[2] P. Tarr, H. Ossher, W. Harrison, and S. M. Sutton, "N degrees of separation: multi-dimensional separation of concerns," in *Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No.99CB37002)*, May 1999, pp. 107–119.

[3] E. Evans, *Domain-Driven Design: Tacking Complexity In the Heart of Software*. Boston, MA, USA: Addison-Wesley, 2003.

[4] J. Thönes, "Microservices," *IEEE Software*, vol. 32, no. 1, pp. 116–116, Jan 2015.

[5] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Software*, vol. 35, no. 3, pp. 24–35, May 2018.

[6] E. Casey, *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet*, 3rd ed. Orlando, FL, USA: Academic Press, Inc., 2011.

[7] R. Rowlingson, "A ten step process for forensic readiness," *International Journal of Digital Evidence*, vol. 2, no. 3, pp. 1–28, 2004.

[8] A. Valjarevic and H. Venter, "A harmonized process model for digital forensic investigation readiness," in *Advances in Digital Forensics IX*, G. Peterson and S. Shenoi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 67–82.

[9] M. Elyas, A. Ahmad, S. B. Maynard, and A. Lonie, "Digital forensic readiness: Expert perspectives on a theoretical framework," *Computers & Security*, vol. 52, pp. 70 – 89, 2015.

[10] L. Pasquale, S. Hanvey, M. Mcgloin, and B. Nuseibeh, "Adaptive evidence collection in the cloud using attack scenarios," *Computers & Security*, vol. 59, pp. 236 – 254, 2016.

[11] D. Alrajeh, L. Pasquale, and B. Nuseibeh, "On evidence preservation requirements for forensic-ready systems," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2017. New York, NY, USA: ACM, 2017, pp. 559–569.

[12] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, "Dapper, a large-scale distributed systems tracing infrastructure," Google, Inc., Tech. Rep., 2010. [Online]. Available: https://research.google.com/archive/papers/dapper-2010-1.pdf

[13] C. Sridharan, *Distributed Systems Observability*. Sebastopol, CA, EUA: O'Reilly Media, 2018.

[14] R. Miles, *Chaos Engineering Observability*. Sebastopol, CA, EUA: O'Reilly Media, 2019.

[15] L. Polyakov, *Mastering Distributed Tracing: Analyzing performance in microservices and complex systems*. Birmingham, United Kingdom: Packt Publishing, 2019.

[16] J. Dogan, "Sampling," 2018. [Online]. Available: https://medium.com/@rakyll/sampling-in-observability-db0142cdda5b

[17] C. Kinneer, R. Wagner, F. Fang, C. L. Goues, and D. Garlan, "Modeling observability in adaptive systems to defend against advanced persistent threats," in *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design*, ser. MEMOCODE '19. New York, NY, USA: ACM, 2019, pp. 10:1–10:11.

[18] L. Pasquale, D. Alrajeh, C. Peersman, T. Tun, B. Nuseibeh, and A. Rashid, "Towards forensic-ready software systems," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '18. New York, NY, USA: ACM, 2018, pp. 9–12.

[19] N. C. Mendonça, P. Jamshidi, D. Garlan, and C. Pahl, "Developing self-adaptive microservice systems: Challenges and directions," *IEEE Software*, pp. 0–0, 2019.

[20] S. Behrens and B. Payne, "Starting the avalanche," 2017. [Online]. Available: https://medium.com/netflix-techblog/starting-the-avalanche-640e69b14a06

[21] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[22] S. Tadelis, *Game theory: an introduction*. Princeton university press, 2013.

[23] D. M. Kreps, *Microeconomic foundations I: choice and competitive markets*. Princeton university press, 2013, vol. 1.

[24] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Bacşar, and J.-P. Hubaux, "Game theory meets network security and privacy," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 25:1–25:39, Jul. 2013.

[25] L. Zhang, T. Zhu, P. Xiong, W. Zhou, and P. S. Yu, "More than privacy: Adopting differential privacy in game-theoretic mechanism design," *ACM Comput. Surv.*, vol. 54, no. 7, jul 2021.

[26] T. Alpcan and S. Buchegger, "Security games for vehicular networks," *IEEE Transactions on Mobile Computing*.

[27] G. Kol and M. Naor, "Cryptography and game theory: Designing protocols for exchanging information," in *Theory of Cryptography*, R. Canetti, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 320–339.

[28] T. Alpcan and T. Basar, "An intrusion detection game with limited observations," in *12th Int. Symp. on Dynamic Games and Applications, Sophia Antipolis, France*, vol. 26, 2006.

[29] P. Liu, W. Zang, and M. Yu, "Incentive-based modeling and inference of attacker intent, objectives, and strategies," *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 1, p. 78–118, feb 2005.

[30] J. Kemeny and J. Snell, *Mathematical Models in the Social Sciences*, ser. A Blaisdell book in the pure and applied sciences. MIT Press, 1978.

[31] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006, rOC Analysis in Pattern Recognition.

[32] J. C. Harsanyi, "Games with incomplete information played by bayesian players, i–iii part i. the basic model," *Management science*, vol. 14, no. 3, pp. 159–182, 1967.

[33] M. Maschler, S. Zamir, and E. Solan, *Game theory*. Cambridge University Press, 2020.

[34] H. Howland, "Cvss: Ubiquitous and broken," *Digital Threats*, vol. 4, no. 1, feb 2022. [Online]. Available: https://doi.org/10.1145/3491263

[35] A. Ghosh and A. Roth, "Selling privacy at auction," *Games and Economic Behavior*, vol. 91, pp. 334–346, 2015.

[36] R. E. Overill and J. Collie, "Quantitative evaluation of the results of digital forensic investigations: a review of progress," *Forensic Sciences Research*, vol. 6, no. 1, pp. 13–18, 2021.

[37] J. Von Neumann and O. Morgenstern, "Theory of games and economic behavior," in *Theory of games and economic behavior*. Princeton university press, 2007.

[38] X. Zhou, X. Peng, T. Xie, J. Sun, C. Xu, C. Ji, and W. Zhao, "Poster: Benchmarking microservice systems for software engineering research," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, 2018, pp. 323–324.

[39] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *arXiv preprint arXiv:2010.16061*, 2020.

[40] A. Arcuri and L. Briand, "A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering," *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.

[41] L. A. Marascuilo and R. C. Serlin, *Statistical methods for the social and behavioral sciences*. WH Freeman/Times Books/Henry Holt & Co, 1988.

[42] S. Sidney, "Nonparametric statistics for the behavioral sciences," *The Journal of Nervous and Mental Disease*, vol. 125, no. 3, p. 497, 1957.

[43] A. Vargha and H. D. Delaney, "A critique and improvement of the cl common language effect size statistics of mcgraw and wong," *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.

[44] N. L. Leech and A. J. Onwuegbuzie, "A call for greater use of nonparametric statistics." 2002.

[45] S. Poulding and J. A. Clark, "Efficient software verification: Statistical testing using automated search," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, p. 763–777, nov 2010.

[46] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[47] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.

[48] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, D. Liu, Q. Xiang, and C. He, "Latent error prediction and fault localization for microservice applications by learning from system trace logs," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2019. New York, NY, USA: ACM, 2019, pp. 683–694.

[49] C. Zhang, X. Peng, C. Sha, K. Zhang, Z. Fu, X. Wu, Q. Lin, and D. Zhang, "Deeptralog: Trace-log combined microservice anomaly detection through graph-based deep learning," in *2022 IEEE/ACM*

*44th International Conference on Software Engineering (ICSE)*, 2022, pp. 623–634.

[50] D. Liu, C. He, X. Peng, F. Lin, C. Zhang, S. Gong, Z. Li, J. Ou, and Z. Wu, "Microhecl: High-efficient root cause localization in large-scale microservice systems," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2021, pp. 338–347.

[51] X. Guo, X. Peng, H. Wang, W. Li, H. Jiang, D. Ding, T. Xie, and L. Su, "Graph-based trace analysis for microservice architecture understanding and problem diagnosis," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 1387–1397.

[52] R. E. Kalman, "On the general theory of control systems," *IFAC Proceedings Volumes*, vol. 1, no. 1, pp. 491 – 502, 1960, 1st International IFAC Congress on Automatic and Remote Control, Moscow, USSR, 1960.

[53] ——, "Mathematical description of linear dynamical systems," *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, vol. 1, no. 2, pp. 152–192, 1963.

[54] X. Zhou, X. Peng, T. Xie, J. Sun, W. Li, C. Ji, and D. Ding, "Delta debugging microservice systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE 2018. New York, NY, USA: ACM, 2018, pp. 802–807.

[55] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Delta debugging microservice systems with parallel optimization," *IEEE Transactions on Services Computing*, pp. 1–1, 2019.

[56] C. Pham, L. Wang, B. C. Tak, S. Baset, C. Tang, Z. Kalbarczyk, and R. K. Iyer, "Failure diagnosis for distributed systems using targeted fault injection," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 503–516, 2017.

[57] J. Tan, "Forensic readiness," *Cambridge, MA:@ Stake*, pp. 1–23, 2001.

[58] J. Sachowski, *Implementing Digital Forensic Readiness: From Reactive to Proactive Process*. Syngress, 2016.

[59] C. Shields, O. Frieder, and M. Maloof, "A system for the proactive, continuous, and efficient collection of digital forensic evidence," *Digital investigation*, vol. 8, pp. S3–S13, 2011.

[60] F. van Staden and H. Venter, "Implementing forensic readiness using performance monitoring tools," in *Advances in Digital Forensics VIII*, G. Peterson and S. Shenoi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 261–270.

[61] D. J. Ras and H. S. Venter, "Proactive digital forensics in the cloud using virtual machines," in *2015 International Conference on Computing, Communication and Security (ICCCS)*, Dec 2015, pp. 1–6.

[62] C. Liu, A. Singhal, and D. Wijesekera, *Identifying Evidence for Cloud Forensic Analysis*. Singapore: Springer Singapore, 2017, pp. 371–391.

[63] J. Zhu, P. He, Q. Fu, H. Zhang, M. R. Lyu, and D. Zhang, "Learning to log: Helping developers make informed logging decisions," in *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ser. ICSE '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 415–425.

[64] C. Sarraute, "Automated attack planning," *arXiv preprint arXiv:1307.7808*, 2013.

[65] Y. Liu, D. Feng, Y. Lian, K. Chen, and Y. Zhang, "Optimal defense strategies for ddos defender using bayesian game model," in *Information Security Practice and Experience*, R. H. Deng and T. Feng, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 44–59.

[66] M. van Dijk, A. Juels, A. Oprea, and R. L. Rivest, "Flipit: The game of "stealthy takeover"," *Journal of Cryptology*, vol. 26, no. 4, pp. 655–713, Oct 2013.

[67] T. Deshpande, P. Katsaros, S. A. Smolka, and S. D. Stoller, "Stochastic game-based analysis of the dns bandwidth amplification attack using probabilistic model checking," in *2014 Tenth European Dependable Computing Conference*, 2014, pp. 226–237.

[68] S. Rass, S. König, and S. Schauer, "Defending against advanced persistent threats using game-theory," *PLOS ONE*, vol. 12, no. 1, pp. 1–43, 01 2017.

[69] J. Pawlick, T. T. H. Nguyen, E. Colbert, and Q. Zhu, "Optimal timing in dynamic and robust attacker engagement during advanced persistent threats," in *2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, 2019, pp. 1–8.

[70] L. Xiao, D. Xu, C. Xie, N. B. Mandayam, and H. V. Poor, "Cloud storage defense against advanced persistent threats: A prospect theoretic study," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 3, pp. 534–544, March 2017.

[71] L. Xiao, D. Xu, N. B. Mandayam, and H. V. Poor, "Attacker-centric view of a detection game against advanced persistent threats," *IEEE Transactions on Mobile Computing*, vol. 17, no. 11, pp. 2512–2523, Nov 2018.

**Davi Monteiro** is a PhD student in Computer Science from the University of Limerick, Ireland, working within Lero, the Science Foundation Ireland Research Centre for Software. His research interests are in software engineering for distributed systems and self-adaptive systems, with emphasis on the following topics: service-oriented and microservices architectures, model-driven engineering, digital forensics, game theory, and cloud computing.

**Yijun Yu** is a full professor in Software Engineering at The Open University, UK. His research focuses on the exploration and development of automated techniques for improving software development processes, in terms of performance of both software engineers and the software artefacts that they produce.

**Andrea Zisman** is Professor of Computing at the Open University (OU) with expertise in the areas of software and service engineering. Her research interests are in adaptation of software systems; traceability of software artefacts; secure software engineering; cloud computing; and IoT-based systems and its applications to food security and circular economy. Andrea has been principal and co-investigator in various EPSRC, European, and industry funded research projects. She holds a Parnas Fellowship from the Irish Software Research Centre (Lero). She is a member of the EPSRC's Information and Communication Technologies (ICT) Strategic Advisory Team (SAT). Andrea has been programme co-chair of main conferences in software and service engineering, she is vice-chair of the IFIP Working Group 2.9, and associate editor of several journals.

**Bashar Nuseibeh** is Chief Scientist of Lero - The Irish Software Research Centre, a Professor of Software Engineering at the University of Limerick, Ireland, and a Professor of Computing at the Open University, UK. His research interests lie at the intersection of requirements engineering, adaptive systems, and security and privacy. He served as Editor-in-Chief of IEEE Transactions on Software Engineering, of the Automated Software Engineering Journal, and of ACM Transactions on Autonomous and Adaptive Systems. He is currently an Associate Editor of IEEE Security Privacy Magazine. Bashar is a Member of Academia Europaea and the Royal Irish Academy.