IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 May 2023, accepted 11 June 2023, date of publication 14 June 2023, date of current version 21 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3286376

APPLIED RESEARCH

RLAuth: A Risk-Based Authentication System Using Reinforcement Learning

CLAUDY PICARD AND SAMUEL PIERRE, (Senior Member, IEEE)
Mobile Computing and Networking Research Laboratory (LARIM), Polytechnique Montréal, Montreal, QC H3T 1J4, Canada
Corresponding author: Claudy Picard (claudy.picard@polymtl.ca)

ABSTRACT Conventional authentication systems, that are used to protect most modern mobile applications, are faced with usability and security problems related to their static and one-shot nature. Indeed, one-shot authentication mechanisms challenge the user at the beginning of a session leaving them vulnerable to attacks on lost/stolen devices or session hijacking. In addition, static authentication mechanisms always use the same challenges to authenticate the user without considering the dynamic nature of the risk related to the authentication context. To mitigate these challenges, we propose RLAuth, a risk-based authentication system that can automatically adapt the level of challenge presented to the user on each authentication request based on the current context. RLAuth is based on binary anomaly detection, which is solved using a deep reinforcement learning agent that acts as the classifier. To cope with the high class imbalance in the anomaly detection problem, we propose to use a balanced sampling technique during experience replay and an imbalanced correction factor during reward computation. We evaluate RLAuth on a public dataset using the G-mean metric which is the square root of the product of sensitivity with specificity. This metric is efficient to measure the classification performance of a model under class imbalance since it does not overfit to the majority class. Finally, RLAuth obtained a G-Mean of 92.62%. In addition, the reinforcement learning agent can be trained offline for acceptable results in about 130 s and can then be periodically retrained to improve its performance over time.

INDEX TERMS Anomaly detection, deep reinforcement learning, imbalanced classification, risk-based authentication.

I. INTRODUCTION

The sanitary crisis caused by the COVID-19 pandemic has shown the great benefits linked to the recent growth in mobile applications' offer and mobile device usage. These mobile applications, which range from entertainment to personal health, leverage a safe and contactless society. We now see more than ever the crucial need for reliable, secure, and efficient applications. The large majority of available mobile applications rely on traditional explicit authentication systems using static and one-shot knowledge-based methods (i.e., password, PIN and graphic patterns) or physiological biometrics, which are known to be vulnerable to various presentation attacks [5], [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Zijian Zhang.

Static systems rely on a static set of authentication methods, where the same challenges are presented to the user on each request. In contrast, dynamic authentication systems were proposed to eliminate the problems linked to static authentication [9] by dynamically choosing the authentication challenge on each request. An early example of dynamic authentication system is proposed in [11] where an upgraded multi-factor authentication mechanism is used based on a question-response challenge. In this system, the main authenticator as well as the second layer challenge are chosen dynamically based on the assessed security level. This mechanism can effectively increase the security level of the authentication system but, in its basic form, still falls into one-shot authentication category.

One-shot authentication mechanisms initiate an authentication request when the user starts a session that stays open until manually closed or by an application [13].

This behavior makes applications protected by one-shot authentication vulnerable to attacks on lost/stolen devices or session hijacking [16], [17].

Continuous authentication can be applied to mitigate the vulnerabilities related to one-shot authentication [19]. Most recent works in continuous authentication on behavioral biometrics [20], [21], [22], context-awareness [25], [26] or hybrid approaches [29], [30] can effectively increase authentication conviviality, since they authenticate the user using implicit methods. However, they often rely on complex machine learning algorithms that can be greedy in resource consumption because of their continuous nature.

A dynamic approach to mitigate one-shot authentication's vulnerabilities is risk-based authentication, which asserts the current level of risk based on surrounding context information. On authentication request, it dynamically chooses the challenge presented to the user based on that evaluated risk [9]. In recent years, strong interest has been shown towards risk-based authentication in the research community to balance security and usability. This authentication approach has the potential to increase user conviviality and security while remaining usable in resource-constrained environments [31], [32]. Indeed, without overflowing the system, a risk-based authentication system increases the level of difficulty of the authentication challenge when a high risk is detected. Conversely, it reduces the need for obtrusive authentication in low-risk environments.

However, risk-based authentication systems proposed in the literature face applicability problems. Indeed, some works do not take into consideration the sensitivity of the protected service or asset which directly informs on the potential risk occurring [3], [24]. Other works rely on static components like reauthentication methods or methods based on static contextual values [8], [18], [28]. Finally, most authors did not integrate mobile deployment in their design, leading to greater delays of authentication and privacy issues added by network communication [4], [27].

In this paper, we propose RLAuth, a durable and dynamic cross-application risk-based authentication system that efficiently reinforces mobile applications' authentication methods without compromising user's privacy. RLAuth uses an anomaly detection deep reinforcement learning (DRL) agent to predict if the current context is related or not to an anomaly on each application access. The system then dynamically chooses an authentication method based on the context status and the sensitivity of the application accessed. An explicit authentication is only triggered when an anomaly is detected. To the best of our knowledge, this is the first time that reinforcement learning is used in user authentication systems.

The use of deep reinforcement learning in decision-making problems compared to other machine learning algorithms offers several important advantages: 1) it intrinsically enables online learning and fast real-time inference, 2) it does not require prior knowledge of the environment and 3) it efficiently self-adapts to new observations by learning by trial and error [33], [34]. Hence, reinforcement learning is a good candidate to manage the dynamic nature of security problems in mobile environments. The contributions of this research work are detailed as follows:

1. We design RLAuth, a risk-based authentication system that dynamically adapts the challenge presented to the user on authentication requests based on the evaluated risk of the authentication context. The proposed system can be deployed directly on mobile devices leading to better privacy and using commonly used authentication methods available on most modern mobile devices.
2. We design a risk modeling technique based on the current authentication context, the last successful authentication data, and the data sensitivity of the accessed resource.
3. We design a binary anomaly detection module that uses a DRL agent as a classifier. The reinforcement learning agent continuously learns how to classify the current state by trial and error, which leads to periodic retraining and dynamic context discovery.
4. We propose a balanced training batch sampling technique to increase the imbalanced classification performances.
5. We evaluate RLAuth on a public dataset where data were collected without constraints on mobile devices, user activity or user environment.

The rest of this paper is organized as follows. In Section II, we present an overview of the related works in risk-based authentication and recent applications of reinforcement learning in cyber-security. In Section III, we introduce a brief background on reinforcement learning frameworks analyzed in this paper. We detail the main components of the proposed risk-based authentication system in Section IV. In Section V, we discuss the experimental results. We finally conclude in Section VI.

II. RELATED WORKS

A. RISK-BASED AUTHENTICATION SYSTEMS

Several sources of information are used in the literature to compute the level of risk in risk-based authentication, mainly historic information on the identity of the user, contextual information on the current environment of authentication and data sensitivity. Each of these contextual information provides important information and when combined, they can lead to a well-balanced risk engine and overall risk-based authentication system. Data sensitivity is the most informative element because it directly impacts the computation of the risk of potential attacks. Indeed, if the context is untrusted but the data that we try to protect is of non-importance, then there is no need to protect it and vice versa.

One of the important early work in risk-based authentication was presented in [2] with the introduction of Context-Aware Scalable Authentication (CASA). This system was designed to dynamically choose an active authentication factor for user authentication on mobile devices based on passive factors. A major contribution of this work was to highlight the importance of location in the daily routine

TABLE 1. Comparison of related works.

Related Work	Risk modeling	Context information				Automated context discovery	Periodic adaptation	On-device deployment
		Location	Time	Identity confidence	Data sensitivity			
Progressive auth [1]	SVM			X	X			X
CASA [2]	Naive Bayes (NB)	X						X
[3]	Spatial entropy with cosine similarity	X	X			X	X	X
[4]	Fuzzy inference system				X			
[7]	Fuzzy inference	X			X			
[8]	Statistical framework	X				X	X	
[10]	Bayesian inference		X	X		X	X	X
[12]	KNN	X	X	X		X		X
[14]	OCSVM	X	X	X		X		
[15]	Static policies				X			
[18]	Mathematical model	X	X		X	X		
[23]	Rules			X	X			
Cormorant [24]	Rules	X	X	X				X
RSA [27]	NB	X	X		X	X	X	
Google smart lock [28]	Policies	X						X
RLAuth (ours)	Reinforcement Learning	X	X	X	X	X	X	X

of a typical user. While their method could be extended to use multiple passive factors, only location was used in this work as context information which is not sufficient to accurately represent the context of an authentication request. The system proposed in [3] also uses location as context data but considers it as a spatio-temporal feature in which users profiles are formed based on mobility patterns. These mobility patterns are represented as spatial entropy vectors ensuring user privacy. While informative, both works did not consider data sensitivity when choosing the appropriate authenticator.

In recent years, some domain-specific risk-based user authentication systems were proposed to secure various applications. The authors in [4] proposed a risk engine based on a fuzzy inference system for online banking that considers the cost and benefits of each action. The authors of [7] proposed a risk engine based on fuzzy logic applicable to energy management tasks in smart homes. In [15], a system is proposed to dynamically choose the appropriate authenticator for the authorization process of payment at the point of service (POS) based on a predefined set of rules. While these systems show that risk-based authentication systems are usable in real-life applications, they are not transferable to other domains because the proposed risk engines are too intertwined with their respective application domain. Therefore, in this paper, we propose a risk engine that is application agnostic and can be used to protect any mobile applications, even intra-application processes.

Interesting works were proposed to enable risk-based authentication in a multi-device context. Progressive Auth [1] was designed to secure mobile applications by using information collected from multiple devices to determine the level of confidence in the identity of the user. CORMORANT [24] is similar to Progressive Auth, but it offers more complex features. It dynamically determines the available behavioral biometrics and trusted devices. In CORMORANT, a risk threshold is computed based on location, time, and proximity between the trusted devices. When a device is used, the user confidence is compared to the risk threshold and an explicit authentication is asked if needed. While an interesting concept, these mobile authentication systems based on multi-device context assume that users are, at all times, using multiple devices simultaneously or at the very least carrying multiple mobile devices that are not realistic. Google Smart Lock [28] is an Android locking mechanism that keeps the device unlocked when a known context is detected. The user can specify trusted locations and devices that represent the trusted contexts. In contrast to progressive auth and CORMORANT, Google Smart Lock does not entirely rely on multi-device context. However, the trusted contexts cannot be automatically inferred. Indeed, it requires that the user manually identifies trusted contexts, which increases the burden of the users instead of decreasing it. In addition, an attacker who takes control of the devices can easily change these trusted contexts at will.

Some researchers in risk-based authentication concentrate on re-authentication after a login attempt. In [8], a risk engine evaluates the contextual information, e.g., IP address and user agent, surrounding the login attempt to determine if further information is needed to safely assess the online user's identity. More recently, the author of [10] proposed a risk engine based on Bayesian inference that assess the level of confidence of a login attempt using the historic number of failed login attempts. In [12], the authors proposed a risk-based authentication system for mobile passenger at land/sea border controls based on novelty detection machine learning. While these methods show

good features, they do not take into consideration the data sensitivity. In addition, they are designed to adapt the level of challenge presented after a performed authentication. Thus, the first layer of authentication is always the same no matter what the surrounding context is, leading to static authentication.

Few works consider features related to the user's identity confidence level in the evaluated context. The authors of [14] proposed a risk engine based on anomaly detection usable in risk-based authentication systems for online browsing. The risk is derived from the context classifier confidence on each classification step and an authenticator is chosen according to the level of risk inferred. The authors used a one-class SVM (OCSVM) classifier to solve the anomaly detection problem. Context information, software and device fingerprinting, location, time and last authentication data are used by the proposed risk engine. However, the sensitivity of the data accessed on authentication requests is not considered when choosing the authenticator. More recently, a risk-based authentication for intra-process anomaly detection was proposed in [23]. The system dynamically chooses the authentication method based on the sensitivity of the action taken. However, it only relies on the identity confidence obtained from biometric authentication methods in a continuous mode. Therefore, it relies more on the continuous authentication paradigm which is more computationally greedy.

The authors of [18] proposed a risk engine that is based on formal mathematical modeling of the risk. This model describes a Level of Security that is attributed to a given context and a computed Level of Risk based on the current context, the data sensitivity, and the probability of attack. The components involved in the risk analysis can take values in a predetermined pool of discrete values. In this paper, we use a similar risk modeling. However, our work differs in the fact that the components used in risk analysis are computed dynamically leading to better usability because it can cover an infinity of scenarios without needing the user's involvement. Finally, the system that is closest to our work is RSA Adaptive Authentication [27], a commercial system that acts as an authentication plugin to secure either websites or mobile applications. A Naïve Bayes classifier acts as a risk engine for fraud detection and computes a risk score based on current context data, such as location, time, and device fingerprinting. The organization can choose custom policies that will be used to mitigate the risk based on data sensitivity. The fraud detection model self-adapts over time to new context data. The major difference with our work is that our anomaly detection module is designed to operate directly in the mobile environment. This limits the exchange of sensitive information related to the user, leading to better privacy, and decreasing the delay of authentication. In addition, our system works with any installed application without registration needed. Therefore, we focus on securing the user's assets rather than the organization's assets.

Table 1 shows a brief comparison of the related works presented in this section with our proposed method. In addition to previously identified challenges, we can see that few related works proposed a mechanism for periodic retraining. This feature is important because it ensures the durability of the system by adapting to previously unseen contexts.

B. DEEP REINFORCEMENT LEARNING APPLIED TO CYBER-SECURITY

In recent years, applied reinforcement learning has gained popularity in the cyber-security research community in many security applications such as intrusion detection [35], [36], [37], [38], attack detection [39], [40] or signal and device authentication [41], [42]. In this section, we present an overview of the reinforcement learning techniques used in the most recent works in cyber-security.

Authentication is an important security requirement in any computer system. Furthermore, achieving accurate and efficient authentication is a main challenge in resource constraint environments. A signal and devices authentication model in massive IoT systems was proposed in [41] to increase authentication efficiency. In this work, multi-agent deep reinforcement learning (DRL) using neural fictitious self play algorithm was proposed to choose between authenticating or not an IoT device. Similarly, a PHY authentication mechanism in Vehicular Ad-Hoc Networks (VANETs) was proposed in [42] to increase authentication accuracy while decreasing security overhead. In this work, the best authentication mode and the spoofing test threshold are chosen with a DRL agent based on Neural Episodic Control.

Reinforcement learning applied to attack detection is beginning to raise interest in the research community. In [39], a face anti-spoofing method using REINFORCE was proposed to improve spoofing detection accuracy. In this work, local extracted features of face pictures are passed to a policy gradient agent that infers the position of suspicious areas that are investigated for spoofing detection. In [40], an efficient and adaptive DDoS detection agent in Internet of Vehicle environment is proposed based on transfer learning and reinforcement learning using DDQN framework.

To the best of our knowledge, reinforcement learning was never used for anomaly detection in user authentication systems. However, recent studies on Intrusion Detection Systems (IDS) based on anomaly detection using reinforcement learning are greatly relevant in the context of our research work, since they acknowledge the challenges introduced by imbalanced classification. An adversarial environment reinforcement learning (AE-RL) algorithm for IDS based on Double Deep Q-Networks (DDQN) was proposed in [35] to enable fast inference and accurate online training with class imbalance. In [36], the authors built upon AE-RL to propose AESMOTE, an adversarial reinforcement learning algorithm that uses Synthetic Minority Over-

sampling Technique (SMOTE) to cope with class imbalance. In this work, the authors effectively increased anomaly detection accuracy but at the cost of a much higher training time. Finally, authors of [37] proposed a DDQN agent for IDS based on anomaly detection. They compared DDQN with Deep D-Networks (DQN), Policy Gradient and Actor Critic frameworks. They proved that DQN and DDQN are more appropriate for solving anomaly detection problems. Moreover, it was shown that low values for the discount factor are more suited for classification problems since the objective is to predict the class for the current state only. Finally, in [38], the authors proposed an Anomaly Network Intrusion Detection System (ANIDS) based on Deep SARSA framework to increase attack detection accuracy in highly imbalanced environment. In this work, the DRL agent acts as an attack detector where each action is assigned to different attacks or normal behavior. They compared their method with the State-of-the-art and showed that Deep Sarsa is effective to solve classification problems under high class imbalance.

III. BACKGROUND

In this section, we clarify some concepts used in the rest of the paper. We also detail the assumptions on which the research rests as well as the adversary model.

Deep reinforcement learning (DRL) framework is a relatively new branch of reinforcement learning that was first introduced in [43] and later popularized in [44] with the introduction of Deep Q-Networks (DQN). DRL enables the use of reinforcement learning in continuous state and action spaces by introducing a neural network (NN) as a function approximator in the framework [33]. In this section, we present four model-free reinforcement learning frameworks that can be used to solve continuous state space and discrete action space problems: DQN, DDQN, Deep Sarsa (DS) and Deep Expected Sarsa (DES).

We chose DQN and DDQN because they were proven to be effective in anomaly detection problems [37]. DS and DES are deep reinforcement frameworks less frequently used than Q-learning frameworks. However, the nature of our problem increases the cost of an error in anomaly detection since it would lead to a security breach. Hence, the stability provided by Sarsa learning makes DS and DES attractive candidates to solve the anomaly detection problem at hand [45].

These frameworks optimize their parameters by minimizing the loss function denoted by (1).

$$L(\theta) = (\delta)^2, \tag{1}$$

where δ (2) is the temporal difference (TD) error computed for each transition:

$$\delta = r + \gamma \cdot Q'\left(s', a'\right) - Q_\theta(s, a), \tag{2}$$

where r is the reward, γ is the discount factor, $Q_\theta(s, a)$ is the approximated Q-values for the current state-action and $Q'\left(s'_i, a'_i\right)$ is the approximated Q-values for the next state-action which varies for each framework.

Reinforcement learning agents are based either on off-policy or on-policy. On-policy agents use the same policy to determine the next action and to learn from the reward. Conversely, off-policy agents use two different policies: one to determine the next action and another one to learn from the reward [33]. In this article, we explore both types of agents.

A. Q-LEARNING-BASED FRAMEWORKS

DQN [46] is an off-policy framework that trains a deep convolutional neural network, called the *online network*, using Q-Learning algorithm [47] and experience replay [43]. A separate Q-network, called the *target network*, is used to estimate the target Q-values. For each transition $\left\{s_i, a_i, r_i, s'_i\right\}$ of the randomly sampled batch of size B, the TD error δ is computed using (2) with $Q'\left(s'_i, a'_i\right)$ computed as $max_{a'} Q_{\bar{\theta}}\left(s'_i, a'_i\right)$ here $Q_{\bar{\theta}}$ and Q_θ are the Q-values predicted by the *target network* and *online network*, respectively.

The weights of the *target network* are updated for each T learning step by copying the *online network* weights, leading to a more stable learning. The periodic updating of the *target network* in DQN can increase the learning overhead and increases the memory used by the agent. Therefore, this can negatively impact its usability in resource-constrained environments.

DDQN is an extension of DQN that was proposed in [48] to cope with its maximization bias. The difference with DQN lies in the computation of the next state-action Q-value as $Q_{\bar{\theta}}\left(s'_i, argmax Q_\theta(s'_i, a'_i)\right)$. The maximization step is performed to estimate the next action on the *online network* which is then used to estimate the action-value on *target network*. Therefore, it reduces the overestimation problems by separating the action selection and its evaluation.

B. SARSA-BASED FRAMEWORKS

Deep Sarsa learning [45] is an on-policy DRL method based on Sarsa algorithm. It updates its Q-Network based on the estimated Q-Value $Q_\theta\left(s', a'\right)$ of the next transition $\left\{s', a'\right\}$ where a is the action that will effectively be taken. Sarsa learning is considered more stable than Q-Learning [45] because it does not rely on a purely greedy estimation of a.

Deep Expected Sarsa (DES) is a DRL algorithm that can be on- or off-policy. In this paper, we consider its on-policy version as introduced in [49] as well as its off-policy version. In DES, the expected value of the next transition $Q'\left(s', a'\right)$ is computed as $\sum_a \left(\pi\left(a \mid s'_i\right) \cdot Q_\theta\left(s'_i, a\right)\right)$ according to the policy $\pi\left(a \mid s'_i\right)$ for each action. In this work, the action is chosen with an ε-greedy method. Hence, the action probabilities can be computed using (3).

$$\pi\left(a \mid s'_i\right) = \begin{cases} 1 - \varepsilon, & a = argmax\left(Q_\theta\left(s'_i\right)\right) \\ \frac{\varepsilon}{|A|}, & else \end{cases} \tag{3}$$

With this formulation, the policy $\pi\left(a \mid s'_i\right)$ resolves to Q-Learning when ε equals 0. Thus, Expected Sarsa learning produces an updating process that offers a good balance between Sarsa and Q-Learning [50].

IV. PROPOSED MODEL

In this paper, we present a risk-based authentication designed to complement existing authentication methods. The idea of this authentication system is to add a layer of security when accessing a mobile application by authenticating the user directly on the mobile device. Indeed, in the proposed model, information on the behavior of the user is kept on the device and is never shared with accessed applications leading to better privacy.

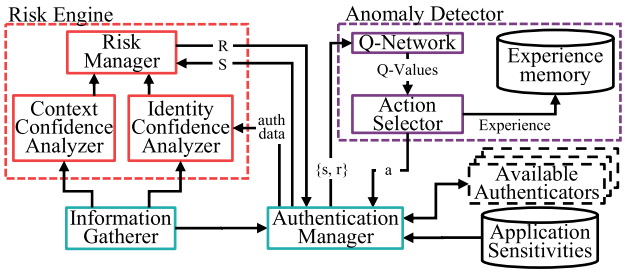


FIGURE 1. System architecture.

In this section, we detail the different components of the proposed risk-based authentication system. Fig. 1 presents an overview of the complete system architecture. The context information is gathered by the Information Gatherer in a continuous mode and is propagated to the risk engine and the authentication manager as new information is available. When the user accesses a new application, an authentication request is triggered in the Authentication Manager. The Authentication Manager then sends context and foreground application information to the Anomaly Detector, which then returns the status of the current state. Finally, based on the state status and the level of sensitivity of the application, the Authentication Manager can decide if further authentication is needed and the authenticator that will be used if necessary. Hence, the proposed system is composed of four main components:

1. The **Information Gatherer** is an active listener that continuously runs in the background. It actively listens to changes in context and is triggered when new information on the current context is available, such as changes in WiFi connection, location, screen locking/unlocking, and foreground application. This information is then relayed to either the Authentication Manager or the Risk Engine for further processing.
2. The **Authentication Manager** is the core module of RLAuth. It oversees user authentication and the post-authentication process. Moreover, it acts as a bridge between the Risk Engine and the Anomaly Detector.
3. The **Risk Engine** computes the current risk based on identity and context information.
4. The **Anomaly Detector** determines the status of the current context by flagging it as either normal or abnormal.

The model proposed in this research paper is based on some assumptions detailed as follows:

1. We assume that humans tend to be predictable in their living patterns and are more likely to visit the same places and do the same actions on a regular basis.
2. Based on the previous assumption, we assume that most of the time, the level of risk is quite low. Indeed, we assume that sensitive applications (such as banking applications) are usually accessed in safer environments and are rarely accessed compared to entertainment applications. Thus, causing a great imbalance between the number of abnormal events and regularities.

A detailed description of the Authentication Manager, the Risk Engine and the Anomaly Detector is provided in the subsequent sections.

A. RISK ENGINE

The risk engine computes the current estimated risk based on the evaluated identity and context confidence level. Identity confidence and context confidence are continuously updated as new information are made available by the Information Gatherer and the Authentication Manager. However, the risk is computed only when needed. Fig. 1 presents an overview of the Risk Engine's components and its relations with other modules of the proposed system.

1) IDENTITY CONFIDENCE ANALYSER

The Identity Confidence Analyzer evaluates the confidence level in the identity of the current user. The identity confidence level $C_{id} \in [0, 1]$ is directly proportionate to the confidence level of the last successful authentication $C_{la} \in [0, 1]$, which is weighted by a dynamic factor $\sigma \in [0, 1]$. σ is a function of the delay $d_{la} \in \mathbb{R}^+$ between the current confidence evaluation and the last authentication and is computed using (4).

$$\sigma = 1 - \frac{d_{la}}{MAX_{d_{la}}}, d_{la} \leq MAX_{d_{la}} \tag{4}$$

Therefore, σ decreases when it gets older and as the time lapse reaches the maximum allowed delay $\text{MAX}_{d_{la}}$, the identity confidence level reaches 0. We choose to design the decrease of σ as a linear decrease because it is a simple and effective relation.

2) CONTEXT CONFIDENCE ANALYSER

The context confidence level $C_{Co} \in [0, 1]$ is evaluated with a simple technique where the frequency of appearance of a given context f_{co} is divided by a constant $\text{MAX}_{f_{co}}$ which denotes the threshold for contexts regularity.

The Context Confidence Analyzer keeps track of the user's context history and maps each encountered context to the frequency $f_{co} \in \mathbb{Z}^+$ that is incremented each time the user is correctly authenticated. The context confidence level reaches its maximum value when f_{co} equals $\text{MAX}_{f_{co}}$.

$\text{MAX}_{f_{co}}$ is a hyperparameter that directly impacts the balance between usability and security and must be chosen carefully. Indeed, if $\text{MAX}_{f_{co}}$ is too high, the usability of the system will decrease because the Context Confidence

Analyzer will take too much time to reach confidence in regular contexts. Such a situation would lead to an increase in explicit authentication requests. Inversely, if $\text{MAX}_{f_{co}}$ is too low, the security of the system will decrease since the Context Confidence Analyzer will reach confidence in unregular contexts too quickly.

3) RISK MANAGER

In this section, we take the classic formulation of the risk as defined in [51] based on the probability of unsatisfactory outcome and the cost of an unsatisfactory outcome. Thus, the risk manager computes the actual risk $R \in [0, 1]$ associated to the current state based on the sensitivity $S \in [0, 1]$ of the application accessed and the probability of a successful attack derived from the current confidence in the identity of the user:

$$R = (1 - (\mu C_{Id} + (1 - \mu) C_{Co})) \cdot S, \tag{5}$$

where C_{Id} is the identity confidence level, C_{Co} is the context confidence level and $\mu \in [0, 1]$ is an hyperparameter of the model that weights the impact of both confidence levels on the overall user confidence. The sensitivities are stored in a database that maps each encountered application to a level of sensitivity. In this work, three levels of sensitivities were defined: low (0), medium (0.5) and high (1). The level of sensitivity of an application is chosen automatically based on the category assigned by Google Play. For example, applications associated with Finance category are more sensitive and are assigned to the highest level of sensitivity. In contrast, application associated to Entertainment are assigned to the lowest level of sensitivity. These sensitivities should be customizable by the user to reflect their personal needs in security. It is worth noting that these levels of sensitivities were chosen to ease the evaluation of the model and can be changed as needed.

B. ANOMALY DETECTOR

The anomaly detection process is modeled as a binary classification problem where the Anomaly Detector classifies the current context as normal or abnormal. To solve this classification problem, we propose to use a DRL agent with experience replay [52]. As stated in Section I, DRL allows an organic form of learning by trial and error that fluidly adapts to new environments. Hence, it is a well-adapted framework to solve a classification problem embedded in a dynamic environment. Fig. 1 presents an overview of the proposed Anomaly Detector.

1) STATE

The state s_t contains the contextual data represented as the vector (conn, loc, day, month, period, la_M, σ, S), where conn, loc, day, month, period and la_M are discrete variables that respectively describes the internet connection status of the device; the current location of the user, the day of the week, the month of the year, the period of the day (either night, morning, afternoon or evening) and the authenticator used on last authentication. σ and S are continuous variables that describes the identity confidence weighting factor (4) and the application sensitivity respectively.

2) ACTION

The problem is modeled as anomaly detection using binary classification where at each time t, an action $a_t \in \{0, 1\}$ is taken. Action 0 flags the current context as abnormal which means that the device could be under attack. Action 1 flags the current context as normal when the risk is low enough to allow the user to continue to use the application without requesting an explicit authentication. The selected action a_t directly influences the next state $s_t^{'}$. Indeed, it dictates how the next authenticator will be chosen which directly impacts variables la_M and σ.

Reward: The anomaly detection of contextual data faces a major challenge linked to class imbalance because the expected number of anomalies is much lower than the expected number of regularities, which makes the learning task harder. To tackle this challenge, the authors of [53] proposed to balance the reward by assigning a lower value to the majority class or action 1 in our design which is linked to anormal context. Therefore, the reward r is modeled based on the risk R_s associated to a given state as well as the imbalance correction factor $\lambda \in R^+$, and is denoted by:

$$r = \begin{cases} 1 & , a = a_e \text{ and } a_e = 0 \\ \lambda & , a = a_e \text{ and } a_e = 1 \\ |C_{ca} - R_s| & , a \neq a_e \text{ and } a_e = 0 \\ -\lambda\,|C_{ca} - R_s| & , a \neq a_e \text{ and } a_e = 1 \end{cases}, \tag{6}$$

where the expected action a_e is obtained by comparing the current risk R_s to a risk threshold $R_{Tr} \in [0, 1]$. Therefore, if $R_s > R_{Tr}$, then a_e is set to 0 else, a_e is set to 1. In this formulation, the reward is balanced using the imbalance correction factor λ only if the expected action a_e corresponds to the majority class. This mechanism reduces the impact of observations related to a normal context to give more place to observations related to an abnormal context leading to a more balanced overall training process. The risk is involved in the reward computation only in case of a misclassification. When a classification error occurs, the reward is the difference between the chosen authenticator's confidence level C_{ca} and the evaluated risk R_s. This penalty does not only inform the agent that an error was made, but it provides information of its impact on the security as well as the usability level.

When $C_{ca} < R_s$, the misclassification leads to a security problem since the confidence of the chosen authenticator was not high enough to mitigate the evaluated risk. In contrast, $C_{ca} > R_s$ leads to a usability problem because the confidence of the chosen authenticator was too high for the evaluated risk. Therefore, the absolute value in the reward computation on misclassification allows to equally penalize the agent on security or usability errors.

Algorithm 1 Anomaly Detector Training Process With Balanced Experience Replay

Input: experience tuple $(s_t, a_t, r_t, s'_j, a'_j)$ and expected action a_e
store $(s_t, a_t, r_t, s'_j, a'_j)$ in mem_{a_e}
evaluate b_0 and b_1 with (8) and (9)
for j=1, b_0 **do**
  $\text{batch}_j \leftarrow$ sample random experience from mem_0
end for
for j= b_0, b_1 **do**
  $\text{batch}_j \leftarrow$ sample random experience from mem_1
end for
for j=1, B **do**
  $s_j, a_j, r_j, s'_j, a'_j \leftarrow \text{batch}_j$
  $Y_j \leftarrow \text{NN}(\theta)$ prediction for s_j
  update $Y_j(s_j, a_j)$ using next state-action Q-value estimation (2)
  $X_j \leftarrow s_j$
end for
train NN(θ) on X and Y with loss function of (1)
$\varepsilon \leftarrow \varepsilon * decay$
update NN($\bar{\theta}$) if needed

3) Q-NETWORK

It is a fully connected layered neural network using back-propagation. The activation function used for each hidden layer is RELU and the output layer is linear. The input layer is composed of eight nodes that correspond to the size of the state vector. The output layer is composed of two nodes that correspond to the size of the action space. The Q-Network outputs the estimated Q-values $Q(s_t; \theta)$ given a state vector s_t.

4) ACTION SELECTOR

At each time t, the action is chosen based on ε-greedy method [46]:

$$a_t = \begin{cases} argmaxQ(s_t; \theta), & p_t > \varepsilon, p_t \in [0, 1] \\ rand(L), & else \end{cases} \quad (7)$$

where L is the set of possible actions that is equal to {0, 1}. To enable exploration, the action selector randomly chooses the next action with a probability of ε,. The agent is created with an ε value of 1 that is slowly decreased throughout the training phase by a chosen decay factor. This decay process allows the agent to perform more exploration iterations at the beginning of the training phase. While the agent gains confidence, the frequency of exploration slowly decreases, and the frequency of exploitations increases until the actions are completely chosen based on the policy.

5) EXPERIENCE MEMORY

The agent uses a process known as Experience Replay [52] that uses past experiences to train the agent. On each training iteration t, the experience tuple $\left(s_t, a_t, r_t, s'_j, a'_j\right)$ is stored in the experience memory. The training batch is then created by sampling B experiences from the experience memory. Because of the presence of great class imbalance in our classification problem, if all experiences are stored in the same memory, the probability of sampling experiences from the minority class is too small. To increase the probability of sampling experiences from the minority class, we could increase the batch size. We could also use over and under sampling algorithms to artificially decrease the class imbalance in the training batch [54]. However, these techniques would considerably increase the training time of the agent [36]. To ensure a good class balance in the training batch, we propose to divide the experiences in two memories of size N, one for each class. Thus, when sampling the training batch, the number of experiences from each class will be derived from the batch size B:

$$b_0 = \begin{cases} \min(B \cdot (1 - bf), mem_0), & bf < 1 \\ \min\left(\frac{B}{bf}, mem_0\right), & else \end{cases}, \quad (8)$$

$$b_1 = B - b_0, \quad (9)$$

where $b_0 \in \mathbb{Z}^{0+}$ and $b_1 \in \mathbb{Z}^{0+}$ are the batch sizes for the minority class and majority class respectively, mem_0 is the number of experiences in the experience memory of the minority class and bf $\in [0, B[$ is the balance factor that determines the balance between the minority and the majority class in the training batch. To ensure that b_0 and b_1 are integers, the value of bf must respect the constraint $B \bmod \text{bf} = 0$. In this design, if bf is less than 1, the proportion of experiences that are drawn from the minority class is larger than the proportion of experiences drawn from the majority class. If bf is larger than 1, we see the opposite behavior. In both cases, the number of experiences drawn from the minority class is always floored by mem_0.

Combining all these elements lets us define the training process of the Anomaly Detector detailed by **Algorithm 1**. Through the training process, this algorithm describes the process that is run on each training iteration. The first step is to store the current experience vector in the appropriate memory. Then, b_0 and b_1 are computed using (8) and (9) and experiences are sampled from their respective database. For each sampled experience, Q-values are predicted and updated using the chosen action-value estimation technique. Finally, the Q-Network is trained on the B pairs of data associating states to Q-values and ε is decayed. Note that with this formulation, any action-value estimation technique can be applied in the update step.

C. AUTHENTICATION MANAGER

The Authentication Manager is the brain of the proposed user authentication system. When a new application is accessed by the user, a new authentication request is triggered. The Authentication Manager is then responsible to choose the appropriate authenticator $auth_t$ in a pool of available authenticators given chosen action a_t (7) and the application's sensitivity:

$$auth_t = \begin{cases} argmin_i\left(\left|C_{A_i} - S_t\right|\right), & if\ a_t = 1, i \in I \\ argmin_e\left(\left|C_{A_e} - S_t\right|\right), & if\ a_t = 0, e \in E \end{cases}, \quad (10)$$

where A is the set of available authenticators on the user's mobile device, I is the set of implicit authenticators, $I \subseteq A$, and E is the set of explicit authenticators, $E \subseteq A$. Therefore, if the status of the current state is flagged as normal, the system chooses an authenticator in the pool of available implicit authenticators. Inversely, if the current state is flagged as abnormal, then the system chooses an authenticator in the pool of available explicit authenticators. This is based on the assumptions that explicit authentication mechanisms are generally more established and thus more accurate than implicit authentication techniques. An even more secure authenticator would combine both methods, but multimodal authentication is not taken into consideration in this work. In both cases, the best authenticator is the one with the confidence level closer to the application's sensitivity.

Each authentication request is resolved in six steps:

1. **Observe state**: The Authentication Manager gathers the current context and authentication confidence information from the Risk Engine.
2. **Select action**: The state vector is sent to the Anomaly Detector to be classified as normal or abnormal.
3. **Choose authenticator**: The appropriate authenticator is chosen using (10).
4. **Perform user authentication**: The user is then asked to perform the required authentication. The user has three attempts to authenticate themselves. This mitigates problems related to brute force attacks by restricting the number of tries that an attacker can perform. If they are correctly authenticated, the Authentication Manager can then process to post-authentication resolution. If they cannot be authenticated after three attempts, then the authentication process restarts from the beginning. Ultimately, if the system reaches a point where no authentication method is adequate to correctly authenticate the user given the current state s, the access to the application should be refused.
5. **Resolve post-authentication**: Each authentication event is resolved in the background. The authentication information is sent to the Risk Engine to update the last authentication information and retrieve the risk R computed using (5) giving the current state s along with the new state s. The reward r is then computed using (6).
6. **Complete iteration**: Depending on the current phase (which are detailed below), the experience tuple is either stored in a transition buffer in the operational phase or directly sent to the Anomaly Detector to complete the RL iteration in the training phase.

This authentication process is represented in Fig. 2 and Fig. 3. Finally, during its lifetime, the proposed authentication system passes through two main phases:

1. **Training phase**: When the user starts the authentication service for the first time, a training process is triggered. This phase is necessary to train the Q-Network parameters to a level of satisfaction where the system can be comfortably used by the user without affecting its usability. If the Q-Network is not previously trained before the operational phase, then the system could be too intrusive or could be unsecure. During this phase, we assume that the user is legitimate. Therefore, if the Authentication Manager chooses an explicit authenticator on a given iteration, it immediately assumes that the user was correctly authenticated by this authenticator. In this phase, any change in the context or any application accesses triggers an authentication request and thus a training iteration. This is done to increase the number of states that are used in the training process. When the Anomaly Detector reaches an appropriate level of accuracy, the system goes into the operational phase.
2. **Operational phase**: In this phase, the system is actively waiting for a new application access. Each authentication request is resolved by the Authentication Manager as described above. The experience tuples stored in the local buffer are periodically sent to the Anomaly Detector to train the RL agent with fresh observations.

The main algorithm of RLAuth is illustrated in Fig. 3. The computational time complexity of the proposed model is mostly affected by the neural network's training and inference

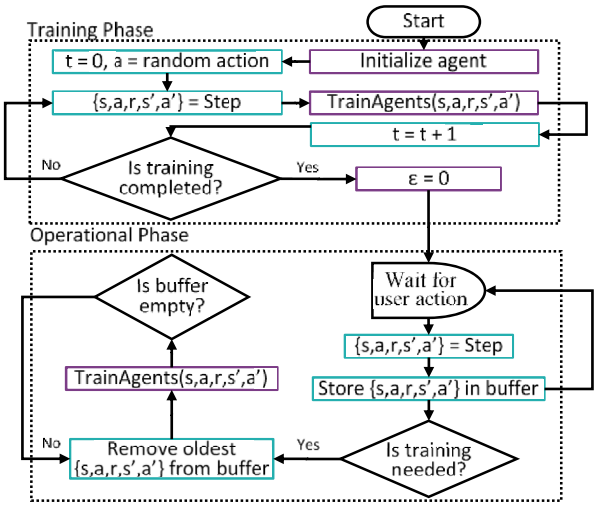


FIGURE 2. Main algorithm detailing training and operational phases.

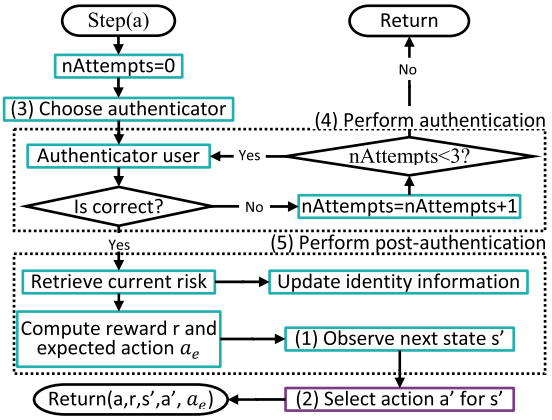


FIGURE 3. Step algorithm performed on each authentication request.

tasks occurring in the anomaly detector. The computational time complexity of a neural network is polynomial and is a function of its architecture's complexity. Of course, training the neural network is greedier than the simple feedforward pass occurring on inference. Indeed, the time complexity of the neural network's training is also dependent on the number of epochs, the number of training samples and the size of the training batch. If we take into consideration that training only occurs in offline settings, the inference is the main contributor to the real-time complexity of the proposed model. Therefore, since we chose a neural network with low complexity, the proposed model is efficient and can be considered usable in a real-life setting.

V. EVALUATION

In this section, we first detail the experimental setup for the evaluation of RLAuth. Then, we evaluate the impact of different hyperparameters of the Anomaly Detector on the system performances. Finally, we compare our proposed method with popular machine learning algorithms for anomaly detection.

A. EXPERIMENTAL SETUP

1) ENVIRONMENT

A prototype of the reinforcement learning agent was implemented on a PC with a processor Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz 2.70GHz. The Q-Networks were implemented using Keras based on Tensorflow. Using the Tensorflow Lite converter, the trained agent was converted to a Tensorflow Lite model, which was successfully embedded in an Android application for on-device inference. In our design, training and retraining is always performed on a server.

2) PARAMETERS SETUP

Table 2 details the model constant and the default parameters used during the experiments. Two explicit authenticators were used in this evaluation: the default OS authenticator, which could be PIN, password or pattern on an Android device, and the fingerprint authenticator available on most Android devices. As an implicit authentication method, we only use the context. The Risk Engine and Authentication Manager constants were chosen to maximize the security, but they could eventually be changed to fit the user's needs. Because of the low dimensionality of the state in the current formulation, we chose to use a simple neural network with only one fully connected hidden layer of 8 neurons. The reinforcement learning constants and default parameters were set as described in [53] except for λ that was set to 1 to accurately evaluate its impact on system performances.

3) DATASET

We use a portion of the Mobile Phone Use (MPU) public dataset from CRAWDAD [55] to evaluate the performances of our system. This dataset contains data collected over four weeks from multiple mobile phone sensors. During the data collection phase, users were instructed to use their personal smartphone as usual, without restrictions on device model or user activity. Hence, this dataset contains data collected in-the-wild that provides realistic information. Therefore, 42 users form our validation (25 users) and testing (17 users) sets with a total of 886 915 samples. The data for each user were cleaned to fit our needs. Only information on foreground application, Wi-Fi connection, location, and time were kept. In the following experiments, all data are sampled continuously from the beginning to ensure that the model for each user is trained and evaluated under realistic conditions.

TABLE 2. Model constant and default parameters.

Risk Engine constants and parameters	
Identity confidence weight (μ)	0.25
$MAX_{f_{co}}$	100
$MAX_{d_{lq}}$	5 min
Authentication Manager	
Authenticators and their confidence level	$\{DEFAULT: 0.8, FINGERPRINT: 0.95\}$
Anomaly risk threshold (R_{Tr})	0.25
Anomaly Detector	
ε decay	0.999
minimum ε	0.001
Stack size (N)	10000
Discount factor (γ)	0.1
Learning rate	0.001
Number of neurons (n)	8
Number of hidden layers (l)	1
Batch size (B)	8
Balance factor (bf)	2
Imbalance correction factor (λ)	1
Training size (ts)	4000
Update target intervals	1000
Agent	DDQN

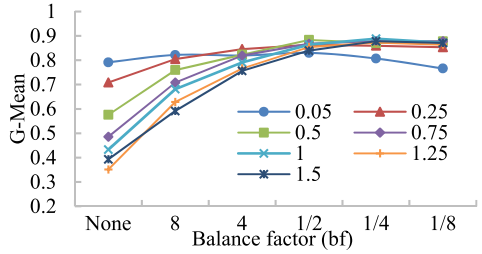
4) PERFORMANCE MEASURES

Accuracy is commonly used to evaluate the performances of classification models. However, this metric cannot be used appropriately in case of imbalanced classification problems. Indeed, accuracy tend to favor the majority class, which is highly problematic [56]. Hence, in this paper, we use the G-mean that has been accepted as a good performance metric to evaluate classifiers for problems with class imbalance [54], [56]. The G-Mean $\left((sensitivity \cdot specificity)^{1/2}\right)$ is computed based on sensitivity $(TP/(TP + FN))$ and specificity $(TN/(FP + TN))$. We also evaluate the efficiency of the model in terms of training computation time and testing computation time.

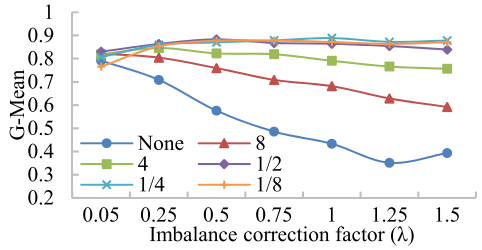
B. IMPACT OF BALANCE AND IMBALANCE CORRECTION FACTORS

We evaluate the proposed techniques to mitigate the problems related to class imbalance. To this end, we study the joint

effect of the balance factor (bf) and the imbalance corrector factor (λ) in terms of the G-Mean. The authors of [53] proposed to use the class imbalanced ratio ρ between the minority and majority classes to compute λ. However, it is not possible in this model because we do not know the value of ρ at the beginning of the training phase. Therefore, we evaluate the effect of fixed values of λ where $\lambda \in 0.05, 0.25, 0.5, 0.75, 1, 1.25, 1.5\}$. The minimum value of λ was chosen based on the average imbalance ratio of the data used in this work. In addition, to respect the constraint $B \bmod \text{bf} = 0$ with $B = 8$, we evaluate the effect of the balance factor where $\text{bf} = \{8, 4, 1/2, 1/4, 1/8\}$. Moreover, we evaluate the performances of the system when no balancing is applied on the training batch sampling. The results are depicted in Fig. 4.



(a) Impact of balance factor (bf) given different values of λ



(b) Impact of imbalance correction factor (λ) given different values of bf

FIGURE 4. Impact of imbalance correction parameters on agent performances.

Fig. 4. (a) shows that when bf decreases, better performances are obtained with higher values of λ. This is explained by the fact that with lower values of bf, more samples from the minority class are sampled. Thus, in this situation, if λ is too low, too much weight is applied to the minority class and the model loses its ability to correctly classify samples from the majority class. Moreover, we observe that when no class balancing is applied on the batch sampling, λ can effectively improve the quality of the classifier when its value is close to the class imbalance ratio. However, the effectiveness of λ is outperformed by bf when $\text{bf} < 1$. This effect can better be observed in Fig. 4. (b), where λ has a considerable impact on the model performances only when $\text{bf} > 1$ or is not applied. In addition, we can see that the use of bf considerably improves the classification performances with $\lambda \geq 0.25$.

Finally, apart from the case where $\text{bf} = 1/8$ and $\lambda = 0.05$, the G-Mean is improved by the application of the balanced sampling technique whatever are the values of bf and λ. The best average G-Mean of 88.86% on the validation set was obtained with $\text{bf} = 0.25$ and $\lambda = 1$.

C. IMPACT OF TRAINING SIZE

The optimization of the training size (ts) is important since it greatly impacts the security of the system. Indeed, when the training process is too long, the mobile applications are left unprotected for too long after the service is started by the user, which can negatively impact its adoption. However, if the training process is not long enough, the authentication service could switch to operational mode too quickly, which can have a great impact on security and/or usability. Therefore, we study the impact of the number of samples used in the training process on the performances of the model with $\text{ts} \in \{1000, 2000, 3000, 4000, 5000, 6000, 7000\}$, where a thousand samples represent about a day of context data. The results for this experiment are shown in Fig. 5.

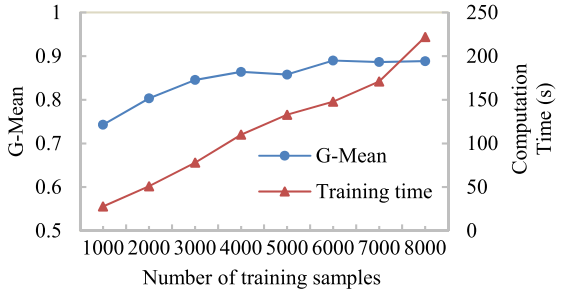


FIGURE 5. Impact of training size on agent performances.

We can see that the G-Mean is improved when ts increases from 1000 to 6000 with average values on validation set ranging from 74.29% to 88.99%. However, no more improvements are observed when ts takes values greater than 6000. Therefore, the anomaly detection agent can effectively converge to a satisfying solution in about 6000 training iterations, which represents less than a week of observations. This means that under these conditions, the system would be usable after approximately one week of training data collection.

As expected, the total training time increases linearly with the training size, but it does not impact the training time per sample, which is ≈25 ms for each value of ts. Hence, a greater value of ts only impacts the delay before the authentication system is usable to protect the user's assets.

D. IMPACT OF PERIODIC RETRAINING

One of the main advantages of using a reinforcement learning agent as a classifier is its ability to continuously adapt to new observations while interacting with the environment. Therefore, we evaluate the impact of the retraining interval (n_{rt}) on the performances of the agent with $n_{rt} \in \{1, 500, 1000, 1500, 2000\}$. The retraining interval corresponds to the number of observations between each retraining process. Hence, when $n_{rt} = 1$, the agent is continuously trained on each observation. In the current

implementation of the system, the agent is retrained on each n_{rt} observations only if the G-Mean on these observations is less than 80%. This ensures that the agent is retrained only when needed. Moreover, we compare the performances of the agent with and without retraining. The results obtained for this experiment are shown in Fig. 6.

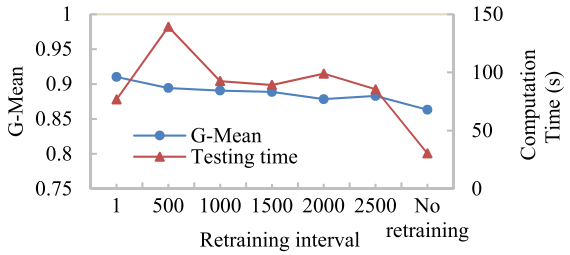


FIGURE 6. Impact of retraining intervals on agent performances.

As expected, the G-Mean slowly decreases as n_{rt} increases with obtained values ranging from 86.64% without retraining to 91.01% with continuous retraining. This behavior is also observable on testing time when $n_{rt} \in [1000, 2500]$. However, when n_{rt} goes from 1000 to 500, the testing time considerably increases with very little variations on the G-Mean. This could be explained by the fact that the retraining threshold is related to the classification performance of the agent on the last n_{rt} observations. Indeed, when the interval between each retraining step is too low, not enough data is available to correctly evaluate the classification performance. Therefore, unnecessary retraining steps are performed, leading to an increase in testing time. Interestingly, we can see a drop in the testing time when the agent is retrained on each new observation. This is related to the fact that the agent is always up to date with the newest context information, which leads to better overall classification performances. Thus, fewer retraining steps are performed on a long-term run.

While the best performances are obtained on a continuous retraining basis, it is not usable when the agent is embedded on the device because it would lead to too much communication traffic on the training server. In addition, constant communication with the server would consume too many resources on the device. However, if the agent is hosted by a remote server, then continuous retraining could be considered. Therefore, we propose to retrain the agent every 1000 observations when the agent is embedded directly on the device. This configuration led to an average G-Mean of 89.06% and a testing time per sample of 4.946 ms.

E. IMPACT OF DIFFERENT DRL LEARNING ALGORITHMS

The choice of the appropriate learning framework to solve the classification problem is of great concern while designing a DRL agent classifier. It was already shown in [37] that DQN and DDQN frameworks are more appropriate than Policy Gradient or Actor-Critic to solve anomaly detection problems. However, to the best of our knowledge, DS and DES were never evaluated in this context.

TABLE 3. Performances comparison of different reinforcement learning algorithms.

	RL learning algorithm	G-Mean	Training time/ sample (ms)	Testing time/sample (ms)
On-policy	DS	0.9024 (±0.0578)	20.915	3.734
	DES	0.9107 (±0.0485)	21.139	3.528
Off-policy	DES	0.9172 (±0.0414)	21.124	3.371
	DQN	0.9086 (±0.0657)	21.139	4.206
	DDQN	0.9185 (±0.0403)	26.204	3.969

TABLE 4. Optimized configuration for model parameters.

minimum ε	0.1
Discount factor (γ)	0.001
Learning rate	0.0055
Balance factor (bf)	0.25
Retraining interval (n_{rt})	1000
Training size (ts)	6000

In this study, we compare the use of on-policy (DS and DES) and off-policy (DES, DQN and DDQN) DRL agent as anomaly classifiers. Table 3 shows the performances of each agent with optimized configuration presented in Table 4. Overall, as expected, the performances obtained from each agent are quite similar. The major difference is observable in the computation time, which is greater for the DDQN agent. Indeed, the added prediction step in DDQN learning increases the computation time on each learning iteration. Moreover, we can see that agents that act off-policy outperformed those that act on-policy. This is due to the fact that off-policy algorithms continuously try to learn the optimal policy while on-policy algorithms learn a near-optimal policy.

As expected, DES can effectively increase the classification performance over DS by slightly increasing the training time. Surprisingly, the agent using off-policy DES was able to reach a G-Mean of 91.72%, which is comparable to the G-Mean obtained by the DDQN agents that is 91.85%. Therefore, DES is the best candidate to act as the anomaly detection agent because it reaches high performances without the complexity added by the target network used in DQN and DDQN frameworks.

F. COMPARISON WITH MACHINE LEARNING ALGORITHMS

Using deep reinforcement learning provides many advantages over conventional machine learning techniques. Indeed, a DRL agent learns online as it interacts with its environment and can continuously adapt to new observations. However, it comes with major drawbacks, such as high training time and high memory consumption. Therefore, to justify the use of a complex DRL agent in RLAuth architecture, we must ensure that it can effectively outperform conventional

TABLE 5. Performances comparison of different machine learning algorithms.

Algorithm Type	RL learning algorithm	G-Mean	Recall		Training time (s)	Testing time/sample (ms)
			Minority class	Majority class		
Supervised	SVM	0.8229 (±0.0954)	0.7769 (±0.1741)	0.8879 (±0.0894)	7.404	0.906
	DNN	0.7062 (±0.1667)	0.5657 (±0.2449)	**0.9376 (±0.0678)**	25.896	1.533
	GNB	0.5873 (±0.3909)	0.9266 (±0.2404)	0.5602 (±0.3884)	**6.660**	**0.867**
	2-NN	0.6315 (±0.0984)	0.4862 (±0.1785)	0.8678 (±0.1150)	14.184	3.121
Unsupervised	Isolation Forest	0.5580 (±0.2251)	0.8075 (±0.2753)	0.4994 (±0.2795)	7.271	39.665
	OCSVM	0.5239 (±0.1004)	0.7698 (±0.1985)	0.4091 (±0.1933)	10.944	1.556
Semi-supervised	RLAuth (ours)	**0.9262 (±0.0523)**	**0.9811 (±0.0350)**	0.8776 (±0.0941)	128.038	3.659

machine learning classifiers in terms of classification performance.

To this end, we compare our proposed Anomaly Detector using a DES agent with supervised (SVM, DNN, Gaussian Naïve Bayes (GNB) and k-Nearest Neighbors (k-NN) and unsupervised (Isolation Forest and OCSVM) classifiers commonly used to solve anomaly detection problems. We use the optimized configuration to perform this experiment and each classifier was evaluated under the same conditions. The DES agent was trained on each observation in an online mode and the other classifiers were trained in an offline mode once all training observations were available. The SVM classifier was initialized to consider the class imbalance in training and the neural network used by the DNN classifier has the same architecture as the Q-Network of the DRL agent. This experiment was conducted with data from the testing set. The results obtained are shown in Table 5.

Unsupervised classifiers generally used in one-class anomaly detection were unable to correctly learn how to distinguish anomalies from regularities. This is not surprising in the case of OCSVM, since it is known to be too sensitive to anomalies when trained with contaminated data. Moreover, we evaluate the performances of OCSVM when only samples from normal states were used to train the classifier and similar results were obtained. The performances of Isolation Forest and OCSVM could probably be improved by optimizing their respective hyper-parameters. However, the major problem with these classifiers is that we need to specify the ratio of outliers anticipated in our data that cannot be known in the context of a real-life application.

Supervised classifiers obtained better results than unsupervised classifiers. A closer analysis to the recall obtained for each class reveals that GNB is the only classifier that can adequately recall samples from the minority class, apart from RLAuth. However, it may poorly recall samples from the majority class, which makes it too sensitive to the minority class. In contrast, DNN and k-NN classifiers can adequately recall samples from the majority class but are unable to correctly recall samples from the minority class. SVM stands out as the best conventional machine learning classifier with a G-Mean of 82.29%, which represents a non-negligible difference of about 10% with the proposed DRL classifier.

Finally, the computation time needed to train our anomaly detection agent is much higher than for other classifiers. This is mostly related to the batch training performed on each iteration. The training of the agent is done in approximately 130 s, which is acceptable considering that the training iterations are performed asynchronously. RLAuth successfully outperformed all other classifiers in terms of G-Mean with an average value of 92.62%.

VI. CONCLUSION

In this paper, we proposed RLAuth, a risk-based authentication system that uses a deep reinforcement learning agent to classify the authentication context as an anomaly or a regularity. To this end, in addition to the Anomaly Detector, we designed a Risk Engine that computes the risk on authentication requests based on the context, the last authentication information and the application's sensitivity. Moreover, an Authentication Manager was proposed to deal with each authentication request and dynamically choose an appropriate authentication method, implicit or explicit, based on the application's sensitivity and the state status inferred by the DRL agent.

The proposed system faces some limitations that we plan to correct in future works. The main limitation with RLAuth is related to the fact that it does not protect the user in familiar contexts. Therefore, the proposed authentication system is vulnerable to attacks performed by coworkers at work or by family members at home. To solve this problem, we could use the proximity of the device in the identity confidence evaluation. Another limitation is related to the choice of the confidence level for each authenticator used in this work and the choice of the application sensitivities. Indeed, these values were arbitrarily chosen based on our domain knowledge. However, mathematical or machine learning models could be used to choose such values more accurately. Finally, in this implementation, all context information is kept on the device memory that could lead to high memory consumption on long-term utilization. To solve this problem, context information could be modeled with machine learning to reduce the memory consumption on the device.

RLAuth was designed to reinforce mobile application's authentication methods by adding a transparent and dynamic layer of security without compromising user's privacy. By taking into consideration the sensitivity of the protected services, we ensure that the proposed system can balance usability and security by design. Moreover, by enabling

periodic retraining and dynamic context discovery, we ensure that the system will be able to adapt to new environments in its lifetime. Finally, we have shown overall that deep reinforcement learning can effectively be applied in the context of user authentication. Indeed, our Anomaly Detector obtained a G-Mean of 92.62% on the testing set with its final configuration.

ACKNOWLEDGMENT

The authors would like to sincerely show their gratitude for the useful comments and insights that were provided by Dre. Franjieh El Khoury throughout the course of this research. They are grateful for the language review that was provided by Marc-André Cyr who greatly improved the quality of this paper. Finally, they thank their industrial partner, Flex Groups, for their support.

REFERENCES

[1] O. Riva, C. Qin, K. Strauss, and D. Lymberopoulos, "Progressive authentication: Deciding when to authenticate on mobile phones," in *Proc. 21st USENIX Secur. Symp. (USENIX Secur.)*, 2012, pp. 301–316.

[2] E. Hayashi, S. Das, S. Amini, J. Hong, and I. Oakley, "CASA: Context-aware scalable authentication," in *Proc. 9th Symp. Usable Privacy Secur.*, Jul. 2013, p. 3.

[3] J. Xiong, J. Xiong, and C. Claramunt, "A spatial entropy-based approach to improve mobile risk-based authentication," in *Proc. 1st ACM SIGSPATIAL Int. Workshop Privacy Geographic Inf. Collection Anal.*, Nov. 2014, p. 3.

[4] W. Han, C. Sun, C. Shen, C. Lei, and S. Shen, "Dynamic combination of authentication factors based on quantified risk and benefit," *Secur. Commun. Netw.*, vol. 7, no. 2, pp. 385–396, Feb. 2014.

[5] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *Proc. WOOT*, vol. 10, 2010, pp. 1–7.

[6] K. Patel, H. Han, and A. K. Jain, "Secure face unlock: Spoof detection on smartphones," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 10, pp. 2268–2283, Oct. 2016.

[7] U. S. Premarathne, "Reliable context-aware multi-attribute continuous authentication framework for secure energy utilization management in smart homes," *Energy*, vol. 93, pp. 1210–1221, Dec. 2015.

[8] D. Freeman, S. Jain, M. Dürmuth, B. Biggio, and G. Giacinto, "Who Are You? A statistical approach to measuring user authenticity," in *Proc. NDSS*, 2016, pp. 1–15.

[9] S. Gupta, A. Buriro, and B. Crispo, "Demystifying authentication concepts in smartphones: Ways and types to secure access," *Mobile Inf. Syst.*, vol. 2018, pp. 1–16, Mar. 2018.

[10] M. Liu, V. Sachidananda, H. Peng, R. Patil, S. Muneeswaran, and M. Gurusamy, "LOG-OFF: A novel behavior based authentication compromise detection approach," in *Proc. 19th Annu. Int. Conf. Privacy, Secur. Trust (PST)*, Aug. 2022, pp. 1–10.

[11] B. Sathish Babu and P. Venkataram, "A dynamic authentication scheme for mobile transactions," *Int. J. Netw. Secur.*, vol. 8, no. 1, pp. 59–74, 2009.

[12] M. Papaioannou, G. Zachos, I. Essop, G. Mantas, and J. Rodriguez, "Toward a secure and usable user authentication mechanism for mobile passenger ID devices for land/sea border control," *IEEE Access*, vol. 10, pp. 38832–38849, 2022.

[13] P. Arias-Cabarcos, C. Krupitzer, and C. Becker, "A survey on adaptive authentication," *ACM Comput. Surveys*, vol. 52, no. 4, pp. 1–30, Jul. 2020.

[14] M. Misbahuddin, B. S. Bindhumadhava, and B. Dheeptha, "Design of a risk based authentication system using machine learning techniques," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug. 2017, pp. 1–6.

[15] A. Wójtowicz and J. Chmielewski, "Technical feasibility of context-aware passive payment authorization for physical points of sale," *Pers. Ubiquitous Comput.*, vol. 21, no. 6, pp. 1113–1125, Dec. 2017.

[16] Y. Liang, S. Samtani, B. Guo, and Z. Yu, "Behavioral biometrics for continuous authentication in the Internet-of-Things era: An artificial intelligence perspective," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9128–9143, Sep. 2020.

[17] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 1, pp. 136–148, Jan. 2013.

[18] M. Sepczuk and Z. Kotulski, "A new risk-based authentication management model oriented on user's experience," *Comput. Secur.*, vol. 73, pp. 17–33, Mar. 2018.

[19] F. H. Al-Naji and R. Zagrouba, "A survey on continuous authentication methods in Internet of Things environment," *Comput. Commun.*, vol. 163, pp. 109–133, Nov. 2020.

[20] M. Abuhamad, T. Abuhmed, D. Mohaisen, and D. Nyang, "AUToSen: Deep-learning-based implicit continuous authentication using smartphone sensors," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5008–5020, Jun. 2020.

[21] Y. Li, H. Hu, Z. Zhu, and G. Zhou, "SCANet: Sensor-based continuous authentication with two-stream convolutional neural networks," *ACM Trans. Sensor Netw.*, vol. 16, no. 3, pp. 1–27, Aug. 2020.

[22] T. Zhu, Z. Weng, G. Chen, and L. Fu, "A hybrid deep learning system for real-world mobile user authentication using motion sensors," *Sensors*, vol. 20, no. 14, p. 3876, Jul. 2020.

[23] S. N. Alotaibi, S. Furnell, and N. Clarke, "A novel transparent user authentication approach for mobile applications," *Inf. Secur. J., A Global Perspective*, vol. 27, nos. 5–6, pp. 292–305, Nov. 2018.

[24] D. Hintze, M. Füller, S. Scholz, R. D. Findling, M. Muaaz, P. Kapfer, E. Koch, and R. Mayrhofer, "CORMORANT: Ubiquitous risk-aware multi-modal biometric authentication across mobile devices," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 3, no. 3, pp. 1–23, Sep. 2019.

[25] Y. Ashibani, D. Kauling, and Q. Mahmoud, "Design and implementation of a contextual-based continuous authentication framework for smart homes," *Appl. Syst. Innov.*, vol. 2, no. 1, p. 4, Jan. 2019.

[26] P. Nespoli, M. Zago, A. H. Celdrán, M. Gil Pérez, F. G. Mármol, and F. J. G. Clemente, "PALOT: Profiling and authenticating users leveraging Internet of Things," *Sensors*, vol. 19, no. 12, p. 2832, Jun. 2019.

[27] R. Security. *RSA? Adaptive Authentication—Solution Brief*. Accessed: 2020. [Online]. Available: https://www.rsa.com/content/dam/en/solution-brief/rsa-adaptive-authentication.pdf

[28] Google. *Choose When Your Android Phone Can Stay Unlocked*. Accessed: 2020. [Online]. Available: https://support.google.com/android/answer/9075927

[29] D. M. Shila and K. Srivastava, "CASTRA: Seamless and unobtrusive authentication of users to diverse mobile services," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 4042–4057, Oct. 2018.

[30] C. Wu, K. He, J. Chen, R. Du, and Y. Xiang, "CaIAuth: Context-aware implicit authentication when the screen is awake," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11420–11430, Dec. 2020.

[31] S. Wiefling, P. R. Jørgensen, S. Thunem, and L. L. Iacono, "Pump up password security! Evaluating and enhancing risk-based authentication on a real-world large-scale online service," *ACM Trans. Privacy Secur.*, vol. 26, no. 1, pp. 1–36, Feb. 2023.

[32] S. Wiefling, L. Lo Iacono, and M. Dürmuth, "Is this really you? An empirical study on risk-based authentication applied in the wild," in *Proc. ICT Syst. Secur. Privacy Protection, 34th IFIP TC 11 Int. Conf.* Lisbon, Portugal: Springer, Jun. 2019, pp. 134–148.

[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[34] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.

[35] G. Caminero, M. Lopez-Martin, and B. Carro, "Adversarial environment reinforcement learning algorithm for intrusion detection," *Comput. Netw.*, vol. 159, pp. 96–109, Aug. 2019.

[36] X. Ma and W. Shi, "AESMOTE: Adversarial reinforcement learning with SMOTE for anomaly detection," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 943–956, Apr. 2021.

[37] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Exp. Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112963.

[38] S. Mohamed and R. Ejbali, "Deep SARSA-based reinforcement learning approach for anomaly network intrusion detection system," *Int. J. Inf. Secur.*, vol. 22, no. 1, pp. 235–247, Feb. 2023.
[39] R. Cai, H. Li, S. Wang, C. Chen, and A. C. Kot, "DRL-FAS: A novel framework based on deep reinforcement learning for face anti-spoofing," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 937–951, 2021.
[40] Z. Li, Y. Kong, and C. Jiang, "A transfer double deep Q network based DDoS detection method for Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 72, no. 4, pp. 5317–5331, Apr. 2023.
[41] A. Ferdowsi and W. Saad, "Deep learning for signal authentication and security in massive Internet-of-Things systems," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1371–1387, Feb. 2019.
[42] X. Lu, L. Xiao, T. Xu, Y. Zhao, Y. Tang, and W. Zhuang, "Reinforcement learning based PHY authentication for VANETs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3068–3079, Mar. 2020.
[43] L. Lin, "Reinforcement learning for robots using neural networks," Carnegie Mellon Univ., Pittsburgh, PA, USA, 1992.
[44] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[45] D. Zhao, H. Wang, K. Shao, and Y. Zhu, "Deep reinforcement learning with experience replay based on SARSA," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–6.
[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Jan. 2015.
[47] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[48] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," 2015, *arXiv:1509.06461*.
[49] M. Ganger, E. Duryea, and W. Hu, "Double SARSA and double expected SARSA with shallow and deep learning," *J. Data Anal. Inf. Process.*, vol. 4, no. 4, pp. 159–176, 2016.
[50] H. van Seijen, H. van Hasselt, S. Whiteson, and M. Wiering, "A theoretical and empirical analysis of expected sarsa," in *Proc. IEEE Symp. Adapt. Dyn. Program. Reinforcement Learn.*, Apr. 2009, pp. 177–184.
[51] P. A. Laplante, *Dictionary of Computer Science, Engineering, and Technology*. Boca Raton, FL, USA: CRC Press, 2017.
[52] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 293–321, May 1992.
[53] E. Lin, Q. Chen, and X. Qi, "Deep reinforcement learning for imbalanced classification," *Appl. Intell.*, vol. 50, pp. 1–15, Aug. 2020.
[54] H. He and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*. Hoboken, NJ, USA: Wiley, 2013.
[55] M. Pielot. (2019). *The Telefonica/Mobilephoneuse Dataset (v. 2019-04-29)* [Online]. Available: https://crawdad.org/telefonica/mobilephoneuse/20190429/
[56] Q. Gu, L. Zhu, and Z. Cai, "Evaluation measures of the classification performance of imbalanced data sets," in *Proc. Int. Symp. Intell. Comput. Appl.* Cham, Switzerland: Springer, 2009, pp. 461–471.

CLAUDY PICARD received the B.Ing. degree in computer engineering from Polytechnique Montréal, in 2018, where she is currently pursuing the Ph.D. degree in computer engineering with the Mobile Computing and Networking Research Laboratory (LARIM). Her research interests include mobile user authentication, biometric authentication, and applied machine learning.

SAMUEL PIERRE (Senior Member, IEEE) received the Ph.D. degree (Hons.) from Université du Quebec à Trois-Rivieres (UQTR), in May 2014, and the Ph.D. degree (Hons.) from Université du Quebec en Outaouais (UQO), in November 2016.

He is currently a Full Professor with the Department of Computer and Software Engineering, Polytechnique Montréal, and the Director of the Mobile Computing and Networking Research Laboratory (LARIM). He has authored or coauthored more than 550 technical publications, including papers in refereed archival journals, textbooks, patents, and book chapters. His research interests include wired and wireless communications, mobile computing and networking, cloud computing, and e-learning. He is a fellow of the Engineering Institute of Canada, in 2003, and the Canadian Academy of Engineering, in 2008. In December 2011, he was appointed as a member of the Order of Canada. He has received several awards, including the Prix Poly 1873 for Excellence in Teaching and Training, in 2001 and 2005, and the Knight of the National Order of Quebec, in 2009. In 2017, he received the El Fasi Prize from Agence Universitaire de la Francophonie (AUF) to highlight the action of a person who has exerted a significant influence through the quality of his expertise and the innovative nature of his achievements at the international level in the fields of research, training, development and international cooperation, governance and/or transfer of knowledge or skills. In 2020, he received the Grand Prize for Professional Excellence from the Order of Engineers of Quebec (OIQ).