Received: 13 September 2021 | Revised: 14 May 2022 | Accepted: 16 May 2022

DOI: 10.1002/spe.3113

RESEARCH ARTICLE

WILEY

Market-inspired framework for securing assets in cloud computing environments

Giannis Tziakouris[1] | Carlos Mera-Gómez[2] | Francisco Ramírez[1] | Rami Bahsoon[1] | Rajkumar Buyya[3]

[1]School of Computer Science, University of Birmingham, Birmingham, Edgbaston, UK

[2]Facultad de Ingeniería en Electricidad y Computación, ESPOL Polytechnic University, Escuela Superior Politécnica del Litoral, ESPOL, Guayaquil, Ecuador

[3]Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, The University of Melbourne, Parkville, Victoria, Australia

Correspondence
Carlos Mera-Gómez, Facultad de Ingeniería en Electricidad y Computación, ESPOL Polytechnic University, Escuela Superior Politécnica del Litoral, ESPOL, Campus Gustavo Galindo Km 30.5 Vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador.
Email: cjmera@espol.edu.ec

Funding information
None

Abstract

Self-adaptive security methods have been extensively leveraged for securing software systems and users from runtime threats in online and elastic environments, such as the cloud. The existing solutions treat security as an aggregated quality by enforcing “one service for all” without considering the explicit security requirements of each asset or the costs associated with security. Dealing with the security of assets in ultra-large environments calls for rethinking the way we select and compose services—considering not only the services but the underlying supporting computational resources in the process. We motivate the need for an asset-centric, self-adaptive security framework that selects and allocates services and underlying resources in the cloud. The solution leverages learning algorithms and market-inspired approaches to dynamically manage changes in the runtime security goals/requirements of assets with the provision of suitable services and resources, while catering for monetary and computational constraints. The proposed framework aims to inform the self-adaptive security efforts of security researchers and practitioners operating in dynamic large-scale environments, such as the Cloud. To illustrate the utility of the proposed framework it is evaluated using simulation on an application based scenario, involving cloud-based storage and security services.

KEYWORDS

assets, cloud computing, market-based, security, self-adaptation

1 | INTRODUCTION

The elastic and multi-tenant nature of the cloud alongside the need for enforcing security through shared, on-demand access has changed the way we view and secure assets. By assets we refer to a “commodity” in the cloud that can be of value for a user/organization such as files, privacy settings, location, digital identities, and accounts among others that support the operation of cloud-based applications.

During the last decade significant progress has been witnessed on self-adaptive security systems for the cloud. In spite of the progress, the existing solutions tend to be limited in the way they treat security and its constraints

Abbreviations: EDR, emergency detection response; IOC, indicators of compromise; IT, information technology; OT, operation technology; SLA, service level agreement; SPs, service providers.

(i.e., computational and monetary). Though that some of the existing cloud-based self-adaptive solutions allow for the customization of their security, they treat security as an "aggregated quality" by enforcing the deployment of "one service for all," while overlooking the security requirements and constraints of individual assets. This practice results into unnecessary high costs due to the usage of obsolete services and resources, as well as unmet security requirements per asset. Moreover, current solutions do not facilitate effective mechanisms for the discovery and allocation of required services and resources to users. Instead, they conduct exhaustive searches in vast search spaces for identifying candidate solutions which can be proven ineffective for ultra-large environments, such as the cloud. Current solutions need to consider the varying nature of different assets and the need for customized, ad-hoc security provision, along with the demand for effective mechanisms for the discovery and allocation of appropriate services and resources. This is a challenging to undertake as self-adaptive systems need not only to continuously meet the changing security needs, priorities, and constraints of multiple assets from different users. They also need to deploy lightweight, efficient mechanisms to enable users (even in the presence of scarce resources) to shortlist candidate solutions and acquire service in a quick manner without embarking into an exhaustive search.

We extend our previous work[1] as an attempt to address these problems. Our work provides a self-adaptive solution that manages the changing security requirements of individual assets using a decentralized agent-based auction procedure and a supervised learning technique. The use of an auction procedure enables the proposed framework to deal with the scale of the problem and the trade-offs that can arise due to the self-interested and diverse nature of security requests coming from various assets. Whereas the use of a supervised learning technique (i.e., random forest classification algorithm)[2,3] allows our framework to operate in a proactive and automated fashion by detecting runtime anomalies that could be indicators of possible security threats and mitigating them prior to their manifestation. By using the random forest classifier, it is feasible to arrive on more efficient bidding plans, informed by historical data. Instead of entering bidders into an exhaustive bidding for candidate offers, the learning helps to identify optimal security strategies for securing an asset, thus allowing us to effectively identify and shortlist appropriate candidate solutions. The selection of random forest classifier was verified by extensive experimentation (Section 5.4) inspired on existing literature[4] that illustrate its utility and effectiveness. In this direction, our article is making two key contributions:

- Propose a self-adaptive security method that builds on market-inspired approaches and learning algorithms for managing the runtime security requirements/goals of assets along with the services and underlying resources required for their satisfaction. The solution uses the random forest algorithm to inform the auctioning process for the effective identification and selection of ad-hoc adaptation strategies for securing assets, while catering for their monetary and computational constraints. The framework is evaluated as a simulation and tested on an application scenario involving cloud-based storage and security services.
- The market-based approach makes a number of auctions mechanisms as sensible candidate. We evaluate and compare how the choice of an auction can affect the trade-offs between security and cost of a solution. The objective is to inform the choice of the auction and its suitability to runtime scenarios. We have engineered and instantiated our system with two dominant market mechanisms, namely, the English[5] and posted-offer auction[6] models.

The results indicate that market-inspired methodologies in conjunction with learning algorithms can pose as an effective and scalable solution for the discovery and allocation of services and resources supporting the security of multiple assets in the cloud. This can be attributed to the decentralized decision making of markets which allow users to handle their own security requirements/data and promote the efficient concurrent management of multiple varying security requests. Moreover, we illustrate that the posted-offer auction model is more effective in occasions where time is a critical aspect of adaptation due to its computational light and simplistic nature, whereas the English auction should be employed when users are secured and need to acquire cheap or scarce services/resources, due to the bargaining that takes place between users and service providers (SPs). Both auction models showed that can effectively operate in environments with limited resources by prioritizing security requests based on their urgency (reflected in bidding prices); in contrast to conventional non-market, "first come first served" allocation mechanisms.

Although previous works on cloud security have revolved around policy enforcement,[7] intrusion detection,[8] authentication,[9,10] and authorization,[11] among others; to be best of our knowledge, this work is the first to focus on the mitigation of runtime threats or satisfaction of security requirements of individual assets. Moreover, we are the first to adopt market-based mechanisms combined with learning algorithms to propose a self-adaptive security approach for cloud computing environments.

The rest of this article is organized as follows. Section 2 motivates the need for a self-adaptive security system that leverages agent-based, market-inspired methodologies and learning algorithms for the selection and allocation of services and resources in the cloud. Section 3 introduces our architecture. Section 4 analyzes the design and implementation of the proposed framework. Section 5 tests and evaluates the architecture using simulation. Section 6 reviews related work. Section 7 concludes with future work.

2 | SELF-SECURING ASSETS BY COMBINING MARKET AND LEARNING

This section motivates the need for a self-adaptive market-inspired framework that caters for the runtime security goals/requirements of assets. It stimulates the usage of agent-based architectures and market-based methodologies as an effective solution to the problem. We then justify the grounds of asserting the effectiveness of our framework based on an application scenario involving cloud-based storage and computational resources for security in a multinational oil and gas environment.

2.1 | The use of market in the self-securing framework

We perceive cloud as the 5th utility;[12] an ultra-large marketplace with shared, on-demand services, and resources that are traded in the same manner as traditional utilities. These services and resources can serve the changing security goals of assets in dynamic and shared environments. The assets can be secured by selecting appropriate services and underlying resources that can best satisfy their changing security goals and constraints. The link between services and resources is crucial as it captures not only the computational overhead that a service could impose on underlying resources but also the security they imply on the solution. In particular, a goal-oriented methodology can be followed. Each security goal (e.g., text file security) can be further decomposed into a set of security sub-goals, such as anonymity, integrity, and availability, which can then be mapped with services and underlying resources that can best support the goals and sub-goals of an asset, as illustrated in Figure 1.

Given the multi-tenant and shared nature of cloud systems along with their ultra-large scale, any cloud-based security solutions should be grounded on fundamentally scalable architectures to eliminate the single point of processing and coordination. Market inspired methodologies can stand as a potential solution to the problem and an effective online optimization mechanism[13-19] for the continuous satisfaction of the varying security requirements of multiple assets. Market solutions enable both users and providers to make their own decisions for maximizing their utility and regulate the supply and demand of services and resources at market equilibrium. In the presence of scarce resources, auctioning mechanisms

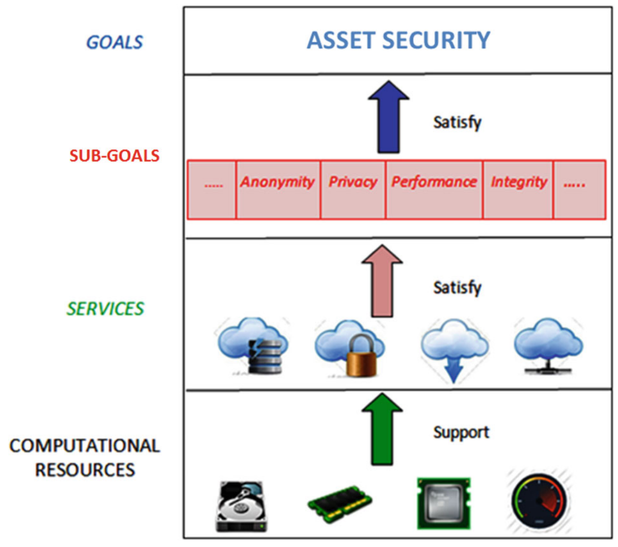

FIGURE 1 Illustrates the link between goals, sub-goals, services, and resources

can promote the effective allocation of services and resources by prioritizing security requests based on their urgency (reflected in bidding prices); Thus, ensuring their provision to users that face imminent threat.

Market models can also promote transparency in the way services and resources are traded and mapped to security goals and sub-goals as their operations are founded on a systematic procedure, henceforth encouraging trust between SPs and users.

Furthermore, the decentralized decision making nature of market methodologies advocate the development of more dependable and elastic self-adaptive systems as they: (i) eliminate the single point of failure; (ii) enable users to handle their own security requirements and data; and (iii) simplify the concurrent management of multiple security requests, as a major part of the computations and decisions are performed in a decentralized manner. Finally, the usage of market principles operate on existing services and resources as an add-on, which can relieve security engineers from introducing structural changes to adaptive solutions, which may cause unintended effects.

2.2 | Motivating example

We consider a cybersecurity incident response driven example in a multinational oil and gas organization. "Company X" operates in 30 countries, utilizing both information technology (IT) and operation technology (OT), which further perplexes security. Multinational oil and gas companies are often targeted by numerous attacks, including state-sponsored groups and organized crime due to them (often) being part of national critical infrastructure that can impact large numbers of users or even entire states and/or countries. Securing assets in such environments is challenging due to the vast difference in cybersecurity requirements and constraints of assets in conjunction with (often) limited resources available for supporting security.

A solution for securing assets in such dynamic and ultra-large environment is the deployment of a market-inspired security solution, enabled by machine learning that will allow users to compete for service, while catering for computational and monetary costs. For the purposes of this example, we operate with the following assumptions:

- The employees of Company X leverage the proposed system to access two main services: (i) *cloud-based storage* for logging security and activity logs that can be used by forensic analysts in the case of a compromise to examine malicious activity; and (ii) *cloud-based computational resources* supporting the use of an emergency detection response (EDR) tool used to defend against runtime threats. The EDR tool maintains different security features that come at different computational and monetary costs. In particular, the EDR provides the following features: antivirus protection, behavioral monitoring, dynamic file analysis, OS advanced querying, treat intelligence analytics, and network intrusion detection.
- The use of storage and computational resources is provided to users by two clouds owned by a subsidiary of Company X. Employees can use any of the two clouds at a cost. The effective use of resources in the environment is required to avoid potential wastage due to limited resources.
- We define assets as any systems/machines such as laptops, desktops, servers, and smart portable devices owned by Company X.
- All assets have software agents deployed, monitoring their contextual data and content. The data monitored for this example is: (i) location of the asset, (ii) security classification (secret, classified, restricted, unclassified) depending on the importance of an asset for Company X, (iii) indicators of compromise (IOC) from the detection of suspicious malicious activity, and (iv) network health, indicating if a device is connected on a trusted network.

For simplicity, this example focuses on the activity of a single user and two assets. The chief financial officer (CFO), John, of Company X uses a laptop and a tablet. The laptop often stores sensitive data and is considered to be a high value asset, where the tablet is often used for meetings with very little or no sensitive data. The initial security configuration for each asset in this example is as follows:

- The laptop operates in a trusted location and network, classified as restricted with no IOCs. In terms of security, an antivirus and advanced behavioral monitoring is used, whereas for logging a cloud storage of 2 GB is used to record access logs and file related activity.
- The tablet is also in a trusted location/network, labeled as unclassified with no IOCs. In terms of security an antivirus is utilized, whereas a cloud storage of 1 GB is used to record access logs.

John needs to travel to country “Y” to conduct business traveling with both assets. He sits in a coffee shop there waiting for a business partner during which he turns on his laptop and tablet. The software agent detects significant changes in the monitored parameters, resulting to: operating from an untrusted location and network, with potential network scanning activity in the network (passive attack behavior detected). The change in the parameters triggers the need for adaptation. Adaptation starts with the random forest classifier analyzing the changes observed for each asset to identify the new security requirements and dynamically shortlist possible solutions based on historical behavior and best security practices. By analyzing historical behavior, the self-adaptive system is able to learn and enforce security in an automated manner, while arriving to a more efficient bidding strategy. Instead of entering bidders into an exhaustive bidding for candidate offers, the learning helps to identify optimal security strategies for securing an asset while avoiding the wasteful allocation of resources to users that do not require their immediate use.

A model security decision that the random forest algorithm could make to address the runtime security requirements of both assets in this scenario is to deploy network intrusion detection along with enhanced threat intelligence analytics. In addition, enforce logging of the network package headers for the laptop. Concerning the tablet, given its low significance for the company, arising from its non-sensitive classification state, deploy network intrusion detection. To meet the new security requirements, the allocation of additional computational resources and storage is required for both assets. As we assume that the two cloud providers maintain scarce resources and there are multiple self-interested users that need to secure their assets, the self-adaptive system uses an auctioning mechanism to allow John and the other users to compete for resources. To do so the software agent of each asset prepares a bid to enter auction. The bid encapsulates a description of the resources required as well as the highest price that John is willing to pay to secure its assets. Once formed, John’s bids are forwarded to a central auctioneer (coordinator) along with bids from rival users for auctioning with the two cloud providers.

There are multiple candidate solutions for auctioning, with each introducing different characteristics, benefits, and limitations to the problem environment. For this example, we consider a variation of the well-known posted-offer auction. The posted-offer auction is conducted in two stages. In the first stage, the sellers (i.e., cloud providers) privately select a price for their storage service for the certain trading period along with the maximum number of storage instances they can offer at that price. Once all sellers make their offers available their prices are revealed to buyers and rival sellers. The trading period follows, where buyers (i.e., Company X employees) are selected in descending order based on their bidding prices (instead of being randomly selected[4]) and given the opportunity to purchase service in a take-it-or-leave-it basis. By introducing user bidding prices in the posted-offer model, it is possible for the self-adaptive system to dynamically determine if a user can afford to pay a seller’s requested price to automate the selection process. As well as, to use bidding prices as a heuristic for ranking/selecting users to identify buyers that face imminent threat and require immediate security provision for their assets. This rises from the sentiment that users that face imminent threat are willing to pay higher prices compared to users that are not threatened. The auctioning round continues until all buyers have acquired service, or until all available resources have been purchased.

For the purposes of this example, we assume that 1000 bids were received from various bidders and sufficient resources exist to only serve of 800 bids/assets. John submitted high bidding prices for the laptop and a low price for the tablet. The prices are automatically calculated according to historical averages of bidding prices of bids with similar security requirements/importance. Based on the submitted bidding prices, John’s laptop acquired the requested resources instantly, whereas the tablet will receive the additional resources with delay to allow other assets with higher bidding prices, necessitating for immediate security to acquire service first. Continuing with the scenario, John is now traveling to another location in country Y and he is using his tablet for a meeting during which he suddenly receives a new document containing sensitive information for an upcoming merger with another company, which is classified as secret. The software agent detects the creation of the new file which causes a change in the asset (tablet) classification changing from unclassified to secret due to the sensitive nature of the new file and redeploys the random forest algorithm to determine the new runtime security requirements. Given the new classification of the asset there is a need for additional security as well as additional logging. In particular, dynamic file analysis, network intrusion detection, and complete logs for all security modifications, file and networking activity are required. As such, a new bid is created to reflect the refined computational and storage requirements supporting the added security features of the tablet along with a higher bidding price to warrant a match with a cloud provider.

3 | CONCEPTUAL MARKET-BASED ARCHITECTURE

This section provides a high-level analysis of the entities and operation phases of the proposed framework (Figure 2).

3.1 | System entities description

- *Agents*: Agents are self-interested, autonomous entities that represent users in the cloud. An agent is responsible for monitoring changes in the runtime security goals and sub-goals of assets and triggering security adaptation. In particular, agents record events that can cause changes in the security of assets along with relevant contextual and behavioral data. By maintaining data records for each user, it is feasible to construct training datasets for informing the random forest algorithm for replicating a user's behavior and enforcing security in an automated and proactive manner. When an unpredictable event occurs that signifies known malicious behavior, the agent examines if the existing services and resources can satisfy the refined security goals and underlying sub-goal(s). If the assessment illustrates that are inadequate, the agent triggers service-resource adaptation. To adapt, the random forest algorithm examines the recorded historical data of a user to discover correlations between previously encountered threats (including their mitigation strategies) and the existing situation. The results from the random forest algorithm are then used to inform the auctioning mechanism for shortlisting candidate solutions by constructing a security specification, called a *bid*, which reflects the refined security goals and sub-goals. The bid consists of the following attributes: (i) the asset that needs to be secured; (ii) the security classification; (iii) the security sub-goals that require support; (iv) the type of service(s) required; (v) the highest price that a user is willing to pay to secure his/her asset; and (vi) the level of security provision required. Once a bid is formed, it is forwarded to the coordinator (auctioneer) for auctioning.
- *Cloud service providers*: Cloud SPs are vendors that trade their services and resources in the market. They are responsible for publicly announcing their offers (called *asks*) to the coordinator. The asks comprise a specification of the traded services and resources along with the price that they want to sell them on. In case of a match between an ask and a bid, the SP allocates the required service(s) and resources to the winner agent by enabling access to the user. The allocated resources fulfill the runtime security goals and sub-goals of an asset. Thus, each user has access to different configurations of services (varying in service type, level of security, security features) and resources (varying in: number of processors, memory, data rate, and storage space) that may be provided by different SPs.
- *Coordinator*: The coordinator is a software system that acts as an auctioneer and market regulator. It sits between agents and Cloud Service Providers and is responsible for implementing trading rounds during which it accepts bids

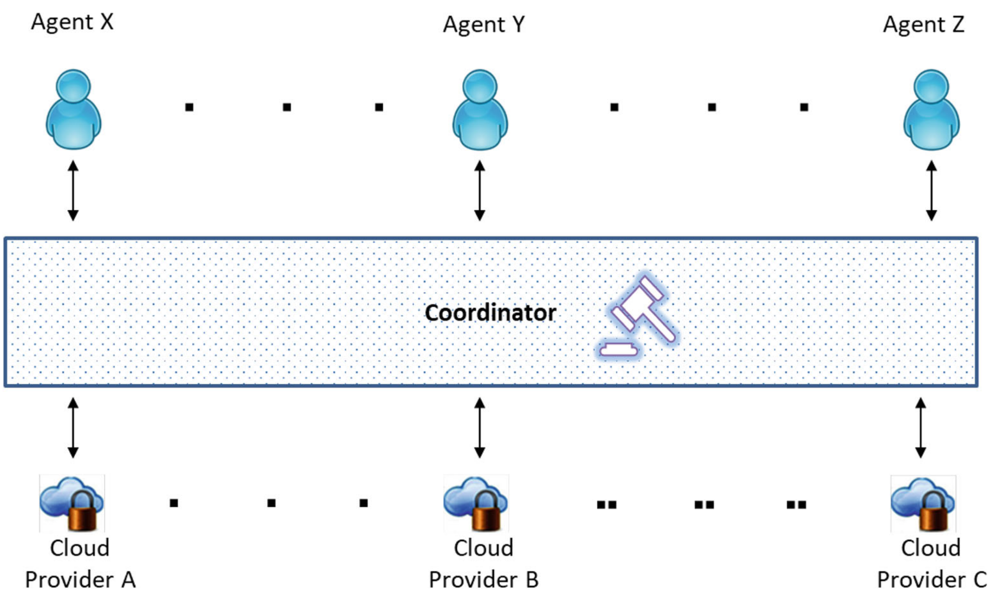

FIGURE 2 Entity-operation diagram

and asks from agents and SPs respectively to match between them. The coordinator also oversees SPs and agents for service level agreement (SLA) compliance as well as warranting the allocation of resource(s) to agents.

3.2 | System operation phases

The proposed framework consists of four phases operating in a continuous cycle, starting with monitoring that passes all relevant contextual and content data to analyze and inform the adaptation process which in turn triggers feedback (allocation of required resources) to meet the changing security requirements of an asset. Once a set of security requirements are met, the monitoring phase starts again to detect new changes in the security requirements of an asset. The operational phases are as follow:

- *Monitor phase*: Sensors operate on the agent side to detect contextual (e.g., user location, available resources, etc.) and content-related (e.g., file modification, etc.) changes. The sensors are realized as lightweight software that can be easily configured to log and process different content and contextual activity that is relevant to the different use cases that the proposed framework is leveraged for. The input of the sensors is fed into a watchdog process that runs as a monitor to depict changes that can affect the security goals and sub-goals of an asset.
- *Analysis phase*: During the analysis phase the data recorded from the sensors are examined to determine if the existing services and resources can satisfy the runtime security goals of an asset. In case where they are deemed insufficient adaptation is triggered. We adopt a goal-oriented approach to map security goals to their sub-goals. For example, the goal "File security" can be further decomposed into authentication, anonymity, and authorization features. Critical changes to the dynamics of one or more interrelated sub-goals can trigger the need for adaptation. As adaption is expensive, the frequency of adaptation can be determined by considering the extent to which the security goals and sub-goals diverge from the tolerance level.
- *Adaptation phase*: Adaptation initiates with the formulation of a bid that inquires the discovery, selection, and allocation of appropriate services and resources from different SPs within a cloud. The bid is constructed with the assistance of the random forest algorithm which identifies the optimal course of action for facing an unpredictable event based on previously encountered threats and mitigation strategies. The bid undergoes a validity test, in which the current state of the user (refers to the existing SLAs that bound a user with SPs) is examined to determine if it violates previous SLAs. User state is critical when dealing with runtime service/resource selection due to conflicts that might arise between preceding ongoing SLAs and newly selected services due to the different policies, objectives, and requirements that bound each service. If a bid is successfully validated, it is forwarded to the coordinator for auctioning, whereas if it maintains conflicting goals, it is refined to remove the conflicting service(s). Once a bid is received by the coordinator, it is entered in an auction with available ask(s) to discover services and resources that can best satisfy the security goals and sub-goals of an asset at the requested price.
- *Feedback phase*: In the event of a match, the agent pays the SP(s) for their services/resources via the coordinator. Following payment, the SP(s) make their services and resources available for allocation to the winner agent. Once a match occurs, the agent forms a SLA with the matched SP(s) and updates the state of the user to contain the latest agreement(s). Similarly, the training dataset of the user is updated to maintain information concerning the latest security threats encountered and their mitigation strategies.

4 | VARIANT DESIGN AND IMPLEMENTATION

This section presents and analyzes the design and implementation of the proposed framework.

4.1 | Adaptation triggering

During the monitor phase, the watchdog service of the agent is deployed on each system monitoring activity to detect and record changes in the security goals and sub-goals of assets (e.g., anonymity, integrity, etc.). Once an unpredictable event is sensed, the recorded data is fed to the analysis phase to determine if adaptation is needed.

Two questions are raised from the monitor and analysis phase: Which system attributes to record? and When should adaptation be stimulated? According to Salehie and Tahvildari,[20] the adaptation methodology varies for each self-adaptive system due to six causes: "(i) the different attributes of adaptable software and the dependency between them; (ii) the temporal aspects of adaptation; (iii) the system attributes that can be altered through adaptation and what needs to be altered in each situation?; (iv) the goals of a system; (v) the level of automation; and (vi) what are the most appropriate actions to take for each given condition?" Since those parameters are unknown to us and determining the optimal attributes for informing adaptation is outside the scope of this work, we do not provide a fine-grained analysis of which system attributes to record and the threshold tolerance for adaptation. Instead, based on our experience with machine learning, we base our adaptation methodology on the following attributes for illustration purposes:

- Location: This dimension is concerned with the location of an asset. The values used to describe this attribute are: secured location, unknown location, and hostile location.
- Indicators of compromise: Illustrates whether suspicious malicious activity is detected against an asset in question.
- Security classification: Illustrates the value of each asset for a company and/or individual. The security classification attributes are as follows.
  - Unclassified: An unclassified asset is of low value for a user and/or an organization, it does not store valuable/sensitive data and does not maintain any significant operational role. It is considered with a low priority when securing multiple assets.
  - Restricted: A restricted asset is of medium value for a user and/or an organization, it stores partially valuable/sensitive data and can maintain a high impact operational role. It is considered with a low to medium priority (depending on the data stored and/or operational role) when securing multiple assets.
  - Confidential: A confidential asset is of high value for a user and/or an organization, it stores valuable/sensitive data and maintains a significant operational role. It is considered with a high priority when securing multiple assets.
  - Secret: A secret asset is of critical value for a user and/or an organization, it stores extremely valuable/sensitive data and maintains a critical operational role. It is considered with urgent priority when securing multiple assets.
- Network health: Describes the security of a network that an asset operates in/is connected to. The values used for describing this attribute are: trusted network, unknown network, and malicious network.

When the "state" of one or more of the aforementioned attributes changes the agents trigger the need for adaptation to proactively secure its assets. To simulate the events causing the changes on the above attributes, we have assigned a timer to each agent which alters their state/values at random time intervals.

4.2 | Forming bids

Once adaptation is triggered, the adaptation phase is employed (algorithmic steps presented in Algorithm 1) to refine and map the security sub-goals of an asset to suitable services and resources. These services/resources are believed to satisfy the runtime security goals and sub-goals of an asset in the most efficient way. In order to perform adaptation, the agents use the random forest algorithm to analyze the historical data of users and accordingly inform the auctioning mechanism via the formulation of a bid. The random forest classifier refers to a supervised learning technique founded on bootstrap aggregation and the random selection of features. Given a dataset D of size $N((x1, y1), \dots, (xN, yN))$, the algorithm constructs B tree classifiers by selecting uniformly at random Z bootstrap samples of size $N \leq N$ from D. At each node of each tree b, where $b \in B$, the algorithm selects a random set of variables m from the variables p ($m \subseteq p$) and splits the nodes in the tree to two children nodes. Once the ensemble trees $T_{b_1}^B$ are constructed, each tree votes for the dominant class with equal weight $\frac{1}{B}\sum_1^B Pb(c|x)$, where c is the probability of each outcome in a single tree and x each test point. The class with the most votes is the algorithm's derived decision (i.e., the constructed bid). The use of ensemble classifiers allow for the mitigation of bias and data overfitting due to their high variance. In-depth experimentation was performed for the selection of the random forest algorithm during which 22 classifiers were examined in terms of effectiveness and accuracy based on two scenarios with 5000 samples provided as a training dataset (Section 5.4). Our findings are further supported by existing literature[4] illustrating the efficiency and effectiveness of the random forest algorithm in analyzing large, complex datasets.

Algorithm 1. Bid formulation

```
Declare: Define $B$ as a set of ensemble trees;
         Define $Z$ as a set of $N$ data samples used to create a tree;
         Define $m$ as variables selected from a total number of $p$ variables in dataset;
         Define $c$ as the probability of each outcome in a tree;
         Define $x$ as each test point/leaf in a tree;
         Define $v$ as a flexible percentile increment value;
1  if Adaptation isNeeded then
       /* Begin RandomForestAlgo                                    */
2      for $b \leftarrow 1$ to $B$ do
3          Select $Z$ samples of $N$ size from Dataset $D$
4          repeat
5              create Tree T$b$;
6              Select $m$ variables from $p$ variables;
7              Pick best split-point among $m$;
8              Split node to two children nodes;
9          until node Nmin is reached in Tree Tb;
10         EnsembleTrees.add (T$b$)
11     end
12     SecDecision ← AVG ( SUM (P$b$ ( c|x ) );
       /* End RandomForestAlgo                                      */
13     historicalData ← retrieveHistoryOfBids();
14     if AuctionType isPostedOffer then
15         fixPrices← ExtractFixBidPrices(historicalData);
16         BidPrice← AVG(fixPrices) $\pm$ AVG(fixPrices) $\times v$;
17         bid← (BidPrice,secDecision);
18     else if AuctionType is EnglishAuction then
19         closPrice ← ExtractClosingPrices(historicalData);
20         BidPrice ← AVG(closPrice) $\pm$ AVG(closPrice) $\times v$;
21         bid ← (BidPrice, secDecision);
22     end
23 end
```

The constructed bids consist of the following attributes: (i) the asset that needs to be secured; (ii) the security classification; (iii) the security sub-goals that require support; (iv) the highest price that a user is willing to pay (the average value of all historical bids with a similar security classification); (v) the type of service (data storage and/or security services) required; and (vi) the level of security provision required. In order for the agents to specify the required level of security provision for each service, they use a predefined list of security descriptors. For security services the following descriptors are used: encryption at transit (encrypts data while they are transferred online, e.g., TLS/SSL); proxy support (usage of proxy servers for anonymity); and media encryption (provides authentication and authorization, e.g., secure real-time transport protocol), whereas storage services are described by: encryption at rest (encryption of stored data); encryption at transit; different encryption keys per file; password protected files; and segmentation of files to different physical machines. Based on the security goals and sub-goals of each asset, the agents use different combinations of security descriptors to request different levels of security provision.

To determine how the choice of an auction can affect the trade-off between security and cost of a solution, we have engineered and instantiated our system with two dominant market mechanisms, namely, the English auction and a variant of the posted-offer auction model. Consequently, we use two different methods to calculate the bidding prices when forming bids. Our modified posted-offer auction model[4] is founded on a take-it-or-leave-it basis. In this model, the SPs publicly announce the services and resources that are trading along with their associated costs for a certain trading period. During the trading period, agents are selected (one at a time) in descending order based on their bidding prices (instead of being randomly selected[4]) and given the opportunity to accept or decline SP offers. By introducing user bidding prices in the posted-offer model it is possible for the self-adaptive system to determine if a user can afford to pay a seller's requested price hence to automate the selection process. As well as to use bidding prices as a heuristic for ranking/selecting users based on the criticality of their requests. The auctioning round continues until all buyers have acquired service, or until all offered service instances have been purchased. To approximate the behavior of a user that is trying to establish the appropriate price for biding, we allow agents to publicly view historical data of bids of interest. Agents determine their bidding prices by calculating the average value of all historical bidding prices with similar security classification and then increase or decrease that value by a percentage $v \left(\frac{\sum_{i=1}^{n} FixedBidPrices}{FixedBidPrices.size} \pm \sum_{i=1}^{n} FixedBidPrices \times v \right)$. v is a flexible parameter which can change according to the needs of the market and its users to achieve market equilibrium. The calculated bidding price is considered to be the highest price that a user is willing to bid/pay in an auction. Once the bidding price is calculated the agent encapsulates the price along with the other required attributes in a bid.

Similarly, the English auction procedure follows almost identical steps to the posted-offer model to calculate bidding prices. In the English auction model [3] the bidding price initiates at a low price (established by the seller/SP) and then raises incrementally such as progressively higher bids are solicited until the auction is closed or no higher bids are received. Therefore, each agent calculates its highest bidding price by considering the closing prices of completed auctions, in contrast to the fixed bidding prices used in the posted-offer model $\left(\frac{\sum_{i=1}^{n} ClosingPrices}{ClosingPrices.size} + \sum_{i=1}^{n} ClosingPrices \times v \right)$.

4.3 | Forming asks

SP's on their side form their offers/asks which they forward to the coordinator for auctioning. Each SP is assigned a timer which at random time intervals triggers the process of constructing and submitting asks. SPs use two different methods for calculating their selling prices, namely, the posted-offer and English auction procedures. In both procedures, the SPs determine the price of their services and resources based on the historical data of submitted asks. SPs estimate their selling prices by calculating the average value of previous submitted ask prices and then subtract or add a percentage s on that value, depending on the profit margin that a SP wants to make. s is a flexible parameter which can change according to the needs of the simulation user. For our experiments, s was randomly generated with normal distribution between the values 0.01 and 0.1. For the posted-offer procedure the values used for calculating the ask prices are the fixed prices requested by SPs ($\frac{\sum_{i=1}^{n} fixedPrices}{NumOfFixedPrices} \pm \sum_{i=1}^{n} fixedPrices \times s$), whereas for the English auction is the closing auction prices of interest ($\frac{\sum_{i=1}^{n} ClosingPrices}{NumOfClosingPrices} \pm \sum_{i=1}^{n} ClosingPrices \times s$). Once the selling price is calculated the agents encapsulate the price

Algorithm 2. Ask formulation

```
Declare: Define $s$ as a parameter with values: 1%-10%;
1  repeat
2      if RandomTimer isTriggered then
3          histData ← retrieveHistoryOfAsks();
4          if AuctionType isPostedOffer then
5              fixPrices ← extractAuctionPrices(histData);
6              cost ← AVG(fixPrices) ± AVG(fixPrices) ×$s$;
7              ask ← (cost, features, resources, securityLevel);
8          else if AuctionType is EnglishAuction then
9              ClosPrice ← extractClosingPrices(histData);
10             cost ← AVG(closPrice) ± AVG(closPrice) ×$s$;
11             ask ← (cost, features, resources, securityLevel);
12         end
13         forwardToCoordinator(ask) ;
14     end
15 until Simulation isTerminated;
```

along with a specification of the offered services and resources in an ask. The algorithmic steps describing the creation of asks are presented in Algorithm 2.

4.4 | Auctioning

Once a bid is received, the coordinator discovers appropriate asks that can satisfy the security goals and sub-goals of an asset and registers the agent in an auction to compete with rival bidders to acquire service. Depending on the method selected for calculating the bid and ask prices (i.e., posted-offer or English auction) there is an analogous procedure for auctioning (algorithmic steps presented in Algorithm 3). In case where the posted-offer methodology is employed, the coordinator discovers SPs that can support the runtime security goals and sub-goals of an asset by comparing the service specification in an ask with the bid specification. In particular, the coordinator compares the: type of service, level of security provision (consisting of the aforementioned security descriptors for each service), computational constraints/requirements (if any) and price to determine the suitability of a service for an agent. In case where an ask violates any of the specified requirements and constraints (e.g., a service offers/utilizes inadequate computational resources) of an asset, the ask is eliminated. Upon elimination of all unsuitable asks, the coordinator sorts agents, that require similar services and resources, in a descending price order to rank them based on the criticality of their bids/requests. Following, the auctioneer selects agents (one at a time) starting from the top of the list to allow them to purchase the needed services, until all agents are served or until all available units are sold.

Algorithm 3. Auctioning procedure

```
Declare: Define p as a bid increment value ;
1  if bid isReceived then
2    if auctionType isPostedOffer then
3      askShortList ← findAsksWhere(bidSecFeature satisfiedBy askSecFeature & bidSecLevelsatisfiedBy
         askSecLevel & bidReqResource satisfiedBy askOfferResource & bidPrice <= askPrice);
4      descendingPriceOrder(bidList);
5      repeat
6        select ($i$th bid from bidList);
7        match ($i$th bid) with (askShortList[Random]);
8        bidList[$i$th + 1]
9      until Bids areServed || Asks areMatched;
10   end
11   else if AuctionType isEnglishAuction then
12     askShortList ← findAuctionsWhere(bidSecFeature satisfiedBy askSecFeature & bidSecLevel satisfiedBy
         askSecLevel & bidReqResource satisfiedBy askOfferResource & bidPrice <= auctionPrice);
13     for $j = 1$st to Nth in askShortList do
14       askShortList[j].setBid(auctionPrice+$p$);
15     end
16     while auctions areNotCompleted do
17       if agent isOutbided & highestBidPrice <= auctionCurrentPrice then
18         reBid(auctionPrice+$p$)
19       end
20     end
21   end
22   if auction isComplete & agent isHighBidder then
23     declareWinner();
24     deleteBidsInSimilarOngoningAuctions();
25   end
26 end
```

TABLE 1 Bid increment values

Current price	Bid increment
\$0.01–\$0.99	\$0.06
\$1.00–\$4.99	\$0.25
\$5.00–\$14.99	\$0.60
\$15.00–\$59.99	\$1.30
\$60.00–\$149.99	\$3.00
\$150.00–\$299.99	\$5.00
\$300.00–\$599.99	\$14.00
\$600.00–\$1499.99	\$25.00

In the occasion where the English auction is used, the coordinator discovers all on-going auctions that satisfy the: type of service, security level, computational constraints (if any) and bidding price and sets a bid on behalf of the agent. The bidding price reflects the current highest price in an auction plus a bid increment value *p* (see Table 1). The bid increment price is the minimum amount by which an agent's bid can be raised to become the highest bidder. The bid increment value can be determined based on the highest bid in an auction. The increment values used by our framework are similar to eBay's proxy auction increment values.[21] These values are case specific and they can be altered by agents according to their runtime needs and the market prices.

In the occasion where a rival agent tries to outbid the winning agent, the out-bid agent automatically increases its biding price to remain the highest bidder, while ensuring that the highest price specified in its bid is not violated. The winning auction, in which a match occurs, is the one that an agent has set a bid and upon completion of the auction round has remained the highest bidder. If a match occurs and the agent has set a bid to more than one ongoing auctions that trade similar services/resources these bids are discarded. Submitting multiple bids to more than one auctions that trade similar services/resources is permitted to increase the likelihood of a match to occur. As the proposed system is concerned with the time critical task of runtime security it needs to efficiently match bids with asks in short time periods to overcome unpredictable events. Failing to do so can result into the compromise of the runtime security of assets.

4.5 | Allocation of services and resources

Once a match occurs the feedback phase is initiated, during which the coordinator notifies the winner SP and agent to commerce the trade. The agent is requested to forward the payment for the won service(s) and resources to the SP. The transaction is recorded by the coordinator to ensure that no party will lie concerning the validity of the payment and allocation. In case where the auctioning was performed based on the English auction, the agent needs to pay the price of the second highest bid plus a defined bid increment (see Table 1), whereas if the posted-offer auction was used the fixed price set by a SP is paid. Once the payment is received, the SP releases the requested services and resources for allocation. Following, the SP creates a virtual machine (VM) to encapsulates the requested service instance(s) and resources and allocates it to the winner agent. The coordinator is paid for its auctioning services by adding a small commission fee for every successful match which is equally split between the winner agent and SP. The algorithmic steps for the allocation procedure is presented in Algorithm 4.

5 | PERFORMANCE EVALUATION

To assert the applicability and usefulness of our method, we have developed a Java prototype simulation system implementing the proposed framework based on the application scenario involving cloud-based storage and computational resources in a multinational oil and gas environment.* We have identified simulation as the dominant methodology for

*Simulation tool can be found at: github.com/GiannisT/UniMarketSimulation.

evaluating our framework as the development of real test-beds limit the experiment to the scale of the test-bed and make the reproduction of results a difficult undertake. Additionally, the creation of real test-beds introduce low-level tasks which are time and money consuming. Furthermore, real life markets do not allow the manipulation of key attributes from developers, which restricts the experiment and its outcomes. Contrary, the use of simulation tools allows the evaluation of hypothesis in an environment where one can reproduce experiments. It allow users to test their services/resources in a repeatable and controllable environment free of cost. At the provider side, simulation environments allow the evaluation of different kinds of resource leasing scenarios under varying loads and pricing distributions. Lastly, simulation tools enable the homogeneous quantification of results as they are generic, architecture imperative solutions, where real life markets have their own composition and deployment requirements.[22]

The engineered solution manages the runtime security goals and sub-goals of the assets of multiple assets (devices owned by the oil and gas company) via the selection and allocation of services and underlying resources. For illustration purposes we restrict the available services in the market to computational resources supporting EDR security features and cloud-based storage services for data security logging. Depending on the security requirements of an asset, different EDR services and logging features can be activated imposing different computational and monetary costs. The EDR provides the following features: antivirus protection, behavioral monitoring, dynamic file analysis, OS advanced querying, treat intelligence analytics, and network intrusion detection. Where storage services support the record of security logging, including network, user authentication requests and file activity. Despite the small number of different types of services and resources supported by our simulation, it can be extended to support a wider variety of services/resources based on the needs of the users and the market itself.

Our simulation system allowed us to answer the following research questions: (i) Can asset-centric security be more cost-effective and efficient for the satisfaction of runtime security goals than aggregated security approaches?; (ii) Can market mechanisms prioritize and satisfy security requests more effectively than current “first come first served” allocation mechanisms?; (iii) How can different auctioning models influence the security, performance, and costs of the proposed framework?; (iv) Can learning algorithms effectively mimic user behavior and secure assets in an automated fashion?; and (v) Can learning algorithms be used to arrive on more efficient bidding plans, instead of entering bidders into an exhaustive bidding for candidate offers?

To conduct our experiment we leveraged a notebook running on Intel Core i5-3210M CPU @ 2.50 GHz, 8 GB RAM and JDK version 17 and configured our simulation in the following way: (i) 20% of service providers operate in a malicious manner and (ii) adverse conditions for 20% of resources selected as damaged. The consideration of malicious activities and adverse conditions in the proposed experiment limit the number of service providers and their resources further illustrating the value of auction-based allocation.

5.1 | Aggregate versus asset-centric security

Existing self-adaptive systems are not concerned with the runtime security of individual assets; instead, they treat security as an aggregated quality, which imposes higher costs and unmet security goals. To demonstrate the above assertion and exemplify the significance of asset-centric security, we have examined if the selection of a single storage service can satisfy the security goals of all of the assets/files maintained by an agent. To conduct this experiment, we have configured our simulation to operate with both asset-centric and aggregated security to draw conclusions concerning the effectiveness

Algorithm 4. Allocation of services and resources

```
1 if AuctionMatch isTrue then
2     notifyWinners(SP, Agent) ;
3     payment ← agent.Pay(auctionPrice + marketFee/2) ;
4     if payment isPerformed then
5         VM ← CreateVM(Services, Resources) ;
6         SP.Allocate(VM)to(Agent) ;
7     end
8 end
```

of each approach. In particular, the purpose of this experiment is to identify the number of assets with satisfied security goals, excessive security provision and insufficient security provision. In the occasion where an asset acquires insufficient security, it can lead to the compromise of its security, where in the presence of excessive security the associated monetary and computational costs can be unnecessarily high.

To conduct this experiment, we have used 1000 assets (company devices). For each asset we have constructed a bid describing its security requirements. All the values used for forming the bids were randomly generated to simulate the diverse nature of the security requirements of different assets. We have then compared the constructed bids with the security specification/ask of a real life, well secured storage provider, that is, pCloud[23] (supports encryption at rest and transit, different encryption keys per file and password protected files), to determine if aggregate security can satisfy the security goals of all assets. Our results demonstrated that only 630 assets met their exact security goals, where 200 obtained excessive security and 170 insufficient security due to absence of data segmentation techniques. In terms of resource consumption, pCloud required the utilization of 2 CPU cores @ 3.16 GHz, 450 MB RAM, and 4 GB of storage to accommodate all 1000 assets. Contrary, the experiment conducted with our asset-centric security framework demonstrated the satisfaction of the security goals of all assets, as well as lower computational costs. To accommodate all 1000 assets our methodology required 1 CPU core, 325 MB RAM, and 2 GB disk space, which is significantly less than the resources reserved in the experiment with aggregate security.

5.2 | Market versus non-market allocation mechanisms

As ultra-large multi-national companies often need to continuously satisfy the changing security requirements of large numbers of users with limited resources, it is essential to use effective allocation mechanisms that can prioritize security requests based on their significance. By doing so, self-adaptive systems will allow users that face imminent threat to acquire service first. This experiment demonstrates that market mechanisms can pose as an effective solution to the above problem as users can reflect the significance of their requests in their bidding prices.

To conduct this experiment, we have simulated a cloud storage provider (i.e., pCloud) and 1000 assets (company devices). The simulation tool was configured to operate with limited resources that only suffice for securing 700 of the assets. For each asset we have randomly generated a bid specification and labeled them according to their security classification: 500 as of "secret," 100 as of "classified," 110 as of "restricted" and 290 as of "unclassified." We have then executed our simulation with both a market (i.e., variant of posted-offer model) and a non-market (first-come-first-served) allocation mechanisms and observed their effectiveness in prioritizing security requests based on the number of requests/assets served from each security classification group. For the experiment with the non-market mechanism, the security requests were submitted in a random order to approximate real-life conditions. The obtained results show that only 352 "secret" requests, 80 "classified" requests, 88 "restricted" requests, and 180 "unclassified" requests were served. The results indicate the wasteful allocation of resources as a large number of "secret" requests were not immediate served, where a big number of "unclassified" requests were accommodated first, leaving high value assets exposed to attacks.

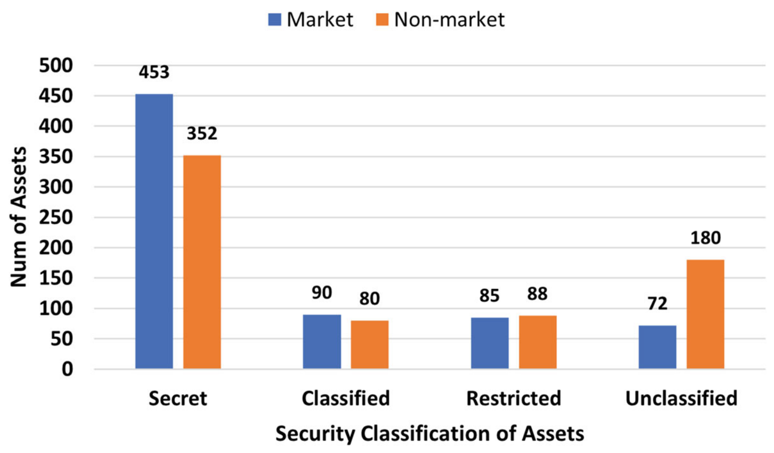

FIGURE 3 Number of assets satisfied from each "security classification group" with market and non-market mechanisms

Following, another experiment was conducted with the proposed market mechanism indicating a more effective allocation of resources, allowing: 453 “secret,” 90 “classified,” 85 “restricted,” and 72 “unclassified” assets to acquire service. The results illustrate a significant increase in served “secrete” classified assets, as well as a massive drop in the number of accommodated “unclassified” requests (as illustrated in Figure 3). This can be attributed to the absence of simplistic allocation mechanisms that are grounded on a “first come first served” basis.

The small number of 47, “secret” assets not accommodated by the posted-offer mechanism can be attributed to the low bidding prices. The bidding prices were semi-randomly generated according to each group of security requests. “Secret” bidding prices were generated with normal distribution between \$30 and \$17, where “classified” bidding prices between \$23 and \$12, “restricted” bids between \$19 and \$9 and bids for “unclassified” between \$15 and \$3. The rational behind selecting these prices is grounded on the sentiment that users that face imminent threat are often willing to pay higher prices to secure their assets compared to users that are secured.

5.3 | Performance

In approximately 40 s our simulation system was able to replicate a cloud environment, construct the system entities described in Section 3, simulate unpredictable events, formulate bids with the assistance of the random forest algorithm, perform market auctions and allocate services and resources to agents. No network latency or noise was simulated in our approach; hence, the time recorded for the simulation is not an accurate representation of the execution time in real life, since it can vary according to the network latency, noise, data rate, collisions, errors, and traffic in a network.

We have observed that our simulation consumed between 2 and 30 MB of memory to operate with the given configuration. More precisely, the proposed framework used 4.087 MB of memory to support all functionalities of the simulation (as illustrated in Figure 4). It was possible to observe three spikes in memory use that are associated to the auctioning and allocation features of the simulation. In terms of CPU usage (as illustrated in Figure 5) the proposed framework when was configured to simulate a cloud environment of 100,000 asks and 100,000 bids required up to 40% of the utilized CPU

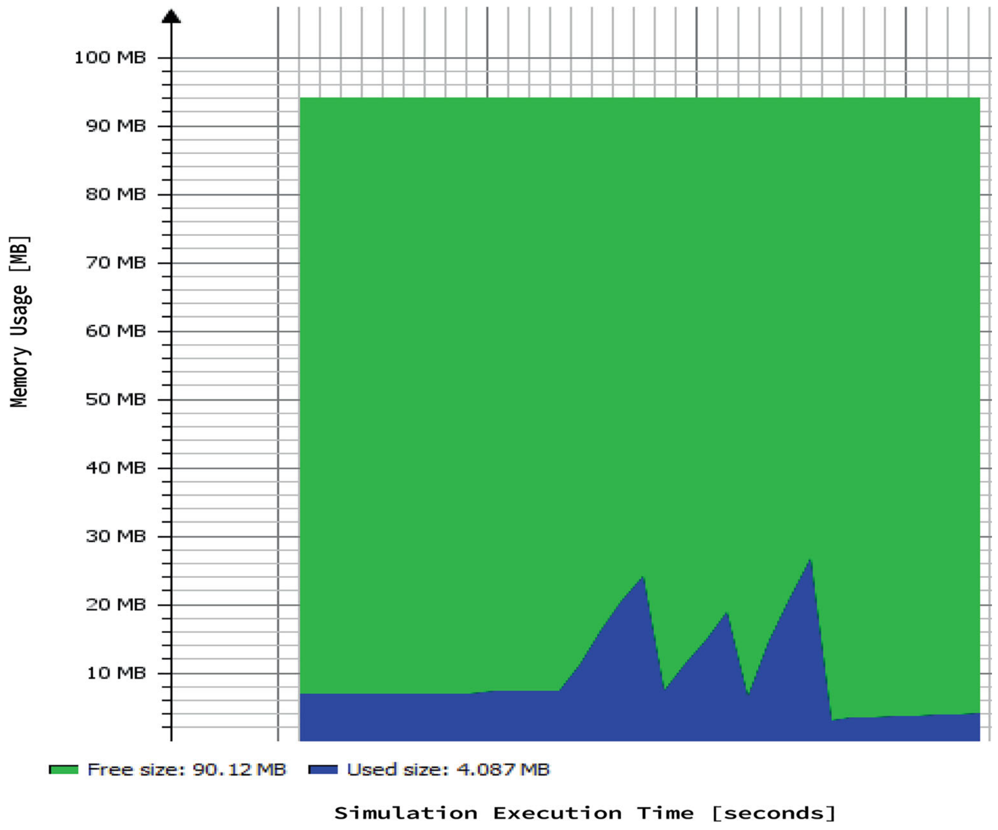

FIGURE 4 Memory usage of simulation

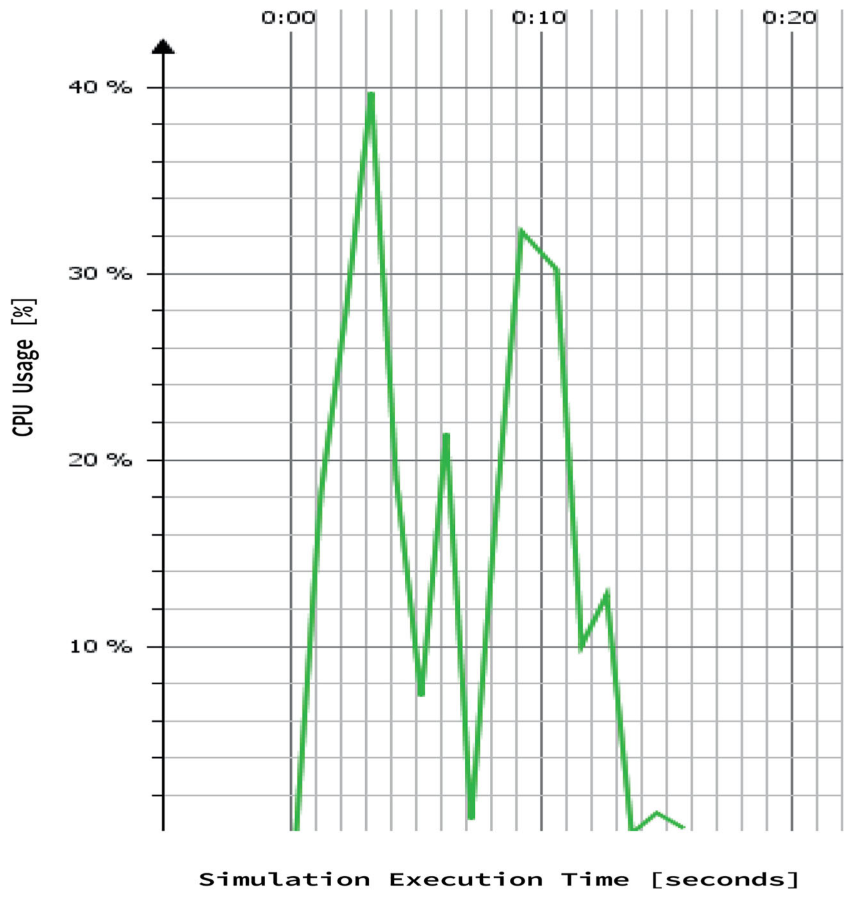

FIGURE 5 Simulation CPU usage

TABLE 2 Simulation execution time

Number of bids	Number of asks	Execution time (s)
10	10	19.03
100	100	19.03
1000	1000	19.03
10,000	10,000	19.03
100,000	100,000	25.05
1,000,000	1,000,000	40.01

resources (Intel Core i5-3210M CPU @ 2.50GHz) to operate. Once more it was possible to observe three spikes of CPU usage that are associated with the auctioning and allocation features of the simulation.

To further examine the performance and scalability of our tool we had executed the simulation six times (using both the posted-offer and English auction models) with a varying number of bids and asks. We have adopted six as our simulation executions as the additional simulation runs haven't provided any insight. The obtained results (see Table 2) illustrate that even when the numbers of asks and bids are significantly increased, the execution time only slightly increases. When the market was configured to auction between 10 and 10,000 bids and asks respectively the execution time remained static to 19.03 s. When the bids and asks were increased to 100,000 the execution time was increased to 25.05 s, whereas when the bids and asks were further increased to 1,000,000 the execution time escalate to 40.01 s.

Throughout this test, we have observed significant differences between the two auctions algorithms. English auction exhibited a more complex, time-consuming procedure that necessitate an average of 789.8 ms to establish an auction

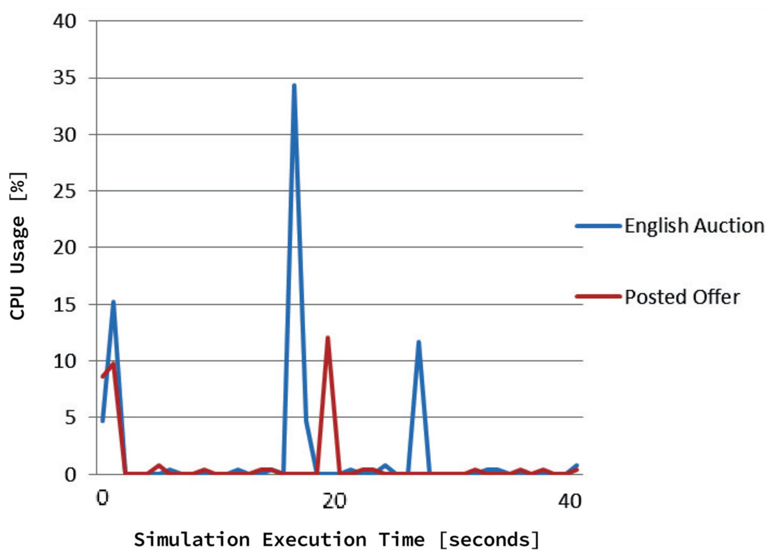

FIGURE 6 CPU usage comparison between English auction and posted-offer algorithm

match in contrast to the 577.8 ms required by the posted-offer algorithm. Additionally, the English auction algorithm displayed signs of high computational overhead, reaching at some occasions 22% more than the CPU overhead generated by the posted-offer model (as illustrated in Figure 6). Contrary, the posted-offer model has proven an efficient and scalable procedure with low computational overhead, due to the lack of complex negotiations between SPs and agents. Therefore, the posted-offer model is more suitable for occasions where agents face imminent threat and time is a critical aspect of adapting security. Contrary, the English auction is more suitable for occasions where agents are not constrained by time, their security is not threaten and are willing to sacrifice time to discover cheap and/or scarce services and resources.

5.4 | Asserting the effectiveness of machine learning algorithms for security

To identify the best machine learning algorithm for our solution and determine whether learning techniques can assist us on reaching more effective bidding plans for security, we have examined the supervised learners offered by WEKA data mining tool[2] and monitored their effectiveness on multiple scenarios involving cloud-based storage and computational resources in an ultra-large environment. In particular, this experiment allowed us to determine: (i) to what extent machine learning algorithms can replace user interaction and secure assets in an automated manner and (ii) if learning algorithms can be used to arrive on more efficient bidding plans.

To evaluate these algorithms, we have generated a training dataset of 5000 samples to inform the supervised learners. Each sample consists of the location, asset security classification and network health of assets in the oil and gas company. We then generated a test dataset of another 5000 samples describing additional scenarios to assert the effectiveness of each algorithm. The values used for constructing the attributes of each sample were semi-randomly generated. The aim of this experiment is for each learning algorithm to classify the test samples according to the level of security provision that they require, namely: secure, fairly secure, and unsecure.

Our results identified (see Table 3) random forest as the dominant learning algorithm for our solution, achieving the correct classification of the 94.86% of the test samples. The acquired results attested the experiments conducted in the work of Caruana and Niculescu-Mizil,[4] which compared a wide number of classification techniques on various datasets and identified random forest as one of the best classification techniques with an extremely high detection accuracy and minimal costs.

In spite of the high classification rate witnessed by the random forest algorithm, its accuracy can vary according to the volume and quality of the data informing its training process. To attest the accuracy of the random forest algorithm in the presence of limited training data, we have used a varying number of training samples and examined the percentage of the correct classified test samples. We have witnessed that even in the presence of 200 training samples, the random forest algorithm was able to correctly classify 88.6% of the 5000 test samples (see Table 4). Showing that the random forest

TABLE 3 Percentage of correctly classified samples per classifier

Classifier	Training	Test	Cross-validation	Average
BayesNet	90.860%	92.000%	90.340%	90.667%
NaiveBayes	91.020%	91.600%	90.300%	90.705%
NaiveBayesMultinomialText	45.320%	46.800%	45.320%	45.390%
NaiveBayesUpdatable	91.020%	91.600%	90.300%	90.705%
Logistic	91.640%	90.000%	90.400%	90.971%
MultilayerPerceptron	92.660%	89.800%	90.380%	91.438%
SimpleLogistic	91.020%	91.400%	90.140%	90.619%
SMO	92.240%	90.600%	90.400%	91.286%
Ibk	100.000%	86.600%	86.760%	93.057%
Kstar	100.000%	14.600%	39.560%	67.152%
DecisionTable	90.920%	90.400%	90.520%	90.705%
Jrip	92.000%	90.000%	89.880%	90.895%
OneR	90.580%	91.600%	90.580%	90.629%
PART	98.020%	90.600%	90.860%	94.257%
ZeroR	45.320%	46.800%	45.320%	45.390%
DecisionStump	53.080%	53.800%	51.060%	52.152%
HoeffdingTree	91.340%	91.200%	90.300%	90.838%
J48	96.500%	89.400%	91.380%	93.724%
LMT	98.720%	90.600%	90.420%	94.381%
RandomForest	100.000%	90.600%	90.140%	94.857%
RandomTree	100.000%	89.600%	85.460%	92.581%
REPTree	93.520%	89.400%	90.900%	92.076%

TABLE 4 Correctly classified samples by random forest

Number of samples	Cross-validation	Test
200	88.05%	88.6%
500	91.61%	90.4%
1000	92.10%	90.6%
2500	91.60%	90.8%
3750	90.21%	91.0%

algorithm can be effective in informing the auctioning mechanism for identifying appropriate services and resources for security, once a small yet significant amount of data is collected for training the algorithm. In the occasion where insufficient and/or inconsistent data is used for informing adaptation it is possible that incorrect conclusions can be drawn, and false security measures are deployed. Therefore, in the existence of insufficient or inconsistent data users need to manually provide input concerning the security countermeasures that they want to enforce.

6 RELATED WORK

The majority of the existing work on cloud security revolves around dimensions such as policy enforcement,[7] intrusion detection,[8] authentication,[9,10] authorization,[11] and encryption frameworks.[24,25] Despite their significance, these solutions are not concerned with the mitigation of runtime threats or the satisfaction of the security requirements of individual assets. Moreover, we are not aware of any systematic attempt in exploring and deploying market-based methodologies for security in cloud. Therefore, we focus on reviewing self-adaptive security methodologies for the cloud that maintain characteristics that can drive the design of our solution.

CyberGuarder[26] is a “visualization security assurance architecture.” It uses three services to enforce security, namely, a “virtual machine security service; a virtual network security service; and a policy based trust management service.” The virtual security service consists of a virtual machine management integrity measurement method for NetApps trusted loading; a NetApp mechanism for the isolation of users from the operating systems; and a mechanism for isolating virtual machines and networks for multiple NetApp’s according to their security requirements and energy constraints. The virtual network service is used for the installation of virtual secure services in a NetApp system, in which the trust management service is used for the implementation of access control policies on network resources as well as the selection of configurations that can maximize the privacy and cost efficiency among different resource pools.

The work of Mazur et al. proposes an agent-modeled self-adaptive security mechanism for the cloud.[27] The framework uses smart agents and network data ontologies to detect and eliminate runtime security threats. The agents are used for the collection of data associated to “network devices, machine language execution, byte code, data streams, and services.” The recorded data are then used to generate data ontologies, which are deployed with game theoretic hazard evaluation by brokering agents for changing their behavior. The usage of ontologies allows the system to perform comparisons between them and determine whether a threat exists, which depends on how close the association between two ontologies is. In case that a malware is detected the framework deploys the predefined security policies to remove it.

CloudSEC[28] is a framework that composes cloud services for the detection and containment of runtime threats. To mitigate threats, CloudSEC deploys three different mechanisms: “the administration group, the collaboration groups, and the peripheral entities.” The administration group is used as the main interface and is maintained by task coordinators. Each coordinator is in control of performing security policy decisions; managing collaboration tasks; and sharing analytical data over different domains. Each collaboration group composes a security service by performing a collaborative process. Collaboration groups comprise of dynamically clustered security agents, with each producing a group of global access points to different security mechanisms. Finally, the peripheral entity aggregates all the SPs and users in the cloud and can deploy security services and consumes resources with the assistance of a push/pull tool.

The work of Squicciarini et al.[29] presents a policy execution methodology for securing users’ sensitive resources (referred as “security-aware objects”) in cloud. According to the context, local laws and service level agreements of a security-aware object the system adapts and deploys different security policies. To secure a security-aware object the framework deploys five security components: “authentication and authorization tools; self-enforcement policy engine; security policies in executable form; secure connections manager; and protected file(s).” Once a security-aware object is formed or its location is changed, the policy composition and translation process is triggered to warrant that only suitable policies are enforced. The policy translation is performed once suitable rules are selected, which is the followed by a static ordering of applicable rules, which result into the dominant security policy.

Ma and Wang[30] promote a dynamic access control method for the cloud. The system comprises of five phases: “monitor, analyze, plan, execute, and knowledge base.” The framework initiates by monitoring the behavior and historical records of user access requests which are fed as an input to the analysis module. The analysis module analyzes the recorded data which ascertains whether it is needed to update the knowledge base by selecting a recorded training

TABLE 5 Summary of related work

	Related work					
Features	**26**	**27**	**28**	**29**	**30**	**Our work**
Asset-centric	×	×	×	✓	✓	✓
Self-adaptive security	✓	✓	✓	✓	×	✓
Market-based adaptation	×	×	×	×	×	✓
Cost-efficient	✓	×	×	×	×	✓

sample. The knowledge base maintains access control data, including the relations between access control attributes. Based on access feedback the relation degree can self-repair and self-improve. Following, the plan module calculates the relation degrees between the recorded samples and accordingly updates the knowledge base for future decisions concerning access control policies. Finally, the execute module obtains the updated knowledge base and enforces the access request decisions. A summary of the comparison between our approach and the closely related works is presented in Table 5.

7 | CONCLUSIONS AND FUTURE WORK

We have developed a method for dynamically managing the security goals of assets and the underlying services and resources required to satisfy them at runtime. The proposed solution allows for the satisfaction of the security goals of individual assets via the identification and selection of appropriate services and resources, while catering for monetary and computational constraints. The method is founded on decentralized agent-based auction models (i.e., the English and posted-offer auction) and the random forest classification technique. The usage of the market allows our solution to deal with the scale of the problem, whereas the random forest classifier enables our system to arrive on more efficient bidding plans and thus identify and shortlist candidate security solutions more effectively.

We have reported on the method simulation and demonstrated its applicability based on an ultra-large, diverse application environment. Our results indicate that asset-centric security is more effective than aggregate security in terms of costs and security goal satisfaction. We have also shown that market models are a dependable and scalable solution for securing multiple assets in environments with limited resources as they allow users to compete for provision in a decentralized manner. Moreover, we have compared the performance of the English and posted-offer auction models and showed that the latter should be employed in occasions where time is a critical aspect of security adaptation due to its trivial nature. Contrary, the English auction should be employed when users are secured and need to acquire cheap and/or scarce services due to the bargaining that takes place between agents and SPs.

As future work, we intend to identify, classify, and mitigate various market specific threats that can disturb the operations of bidders, sellers, and the auction mechanisms in our system.

AUTHOR CONTRIBUTION

All authors contributed to the conceptualization and writing of the paper. Some authors led specific activities as follows. Conceptualization and defining the problem statement: G. Tziakouris, C. Mera-Gómez, and R. Bahsoon. Approach design and evaluation: G. Tziakouris, C. Mera-Gómez, F. Ramírez, and R. Bahsoon. Related work: G. Tziakouris, and C. Mera-Gómez. Study supervision: R. Bahsoon and R. Buyya.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Carlos Mera-Gómez https://orcid.org/0000-0002-7014-1138
Rajkumar Buyya https://orcid.org/0000-0001-9754-6496

REFERENCES

1. Tziakouris G, Mera-Gómez CJ, Bahsoon R. Securing cloud users at runtime via a market mechanism: a case for federated identity. In: 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS) IEEE; 2014. p. 221–228.
2. Holmes G, Donkin A, Witten IH. Weka: a machine learning workbench. Proceedings of ANZIIS'94-Australian New Zealand Intelligent Information Systems Conference. IEEE; 1994:357-361.
3. Liaw A, Wiener M. Classification and regression by random forest. *R News 2*. 2002;3:18-22.
4. Caruana R, Niculescu-Mizil A. An empirical comparison of supervised learning algorithms. Proceddings of the 23rd International Conference on Machine Learning; 2006:161-168.
5. Gul F, Stacchetti E. The English auction with differentiated commodities. *J Econ Theory*. 2000;92(1):66-95.
6. Ketcham J, Smith VL, Williams AW. A comparison of posted-offer and double-auction pricing institutions. *Rev Econ Stud*. 1984;51(4):595-614.

7. Tari Z, Fernandez G. Security enforcement in the DOK federated database system. *Database Security.* Boston, MA: Springer; 1997:23-42.
8. Watson P. A multi-level security model for partitioning workflows over federated clouds. *J Cloud Comput*. 2012;1:1-15.
9. Hatala M, Eap TMT, Shah A. Federated security: lightweight security infrastructure for object repositories and web services. Proceedings of the International Conference on Next Generation Web Services Practices; 2005.
10. Yan L, Rong C, Zhao G. Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography. IEEE International Conference on Cloud Computing; 2009:167-177.
11. Gaedke M, Meinecke J, Nussbaumer M. A modeling approach to federated identity and access management. Proceedings of the Special Interest Tracks and Posters of the 14th International Conference on World Wide Web; 2005:1156-1157.
12. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Gener Comput Syst*. 2009;25(6):599-616.
13. Ruth P, Rhee J, Xu D, Rick K, Goasguen S. Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure. Proceedings of the 2006 IEEE International Conference on Autonomic Computing; 2006:5-14; IEEE.
14. Fu Y, Chase J, Chun B, Schwab S, Vahdat A. SHARP: an architecture for secure resource peering. *ACM SIGOPS Operat Syst Rev*. 2003;37(5):133-148.
15. Lai K, Rasmusson L, Adar E, Zhang L, Huberman BA. Tycoon: an implementation of a distributed, market-based resource allocation system. *Multiagent Grid Syst*. 2005;1(3):169-182.
16. AuYoung A, Chun B, Snoeren A, Vahdat A. Resource allocation in federated distributed computing infrastructures. Proceedings of the 1st Workshop on Operating System and Architectural Support for the on demand IT Infrastructure (OASIS 2004); 2004.
17. Irwin DE, Chase JS, Grit LE, Yumerefendi AR, Becker D, Yocum K. Sharing networked resources with brokered leases. Proceedings of the USENIX Annual Technical Conference (USENIX 2006); 2006.
18. Buyya R, Abramson D, Giddy J, Stockinger H. Economic models for resource management and scheduling in grid computing. *Concurr Comput Pract Exp*. 2002;14(13-15):1507-1542.
19. Vytelingum P, Cliff D, Jennings NR. Strategic bidding in continuous double auctions. *Artif Intell*. 2008;172(14):1700-1729.
20. Salehie M, Tahvildari L. Self-adaptive software: landscape and research challenges. *ACM Trans Auton Adapt Syst*. 2009;4:1-42.
21. Ebay, bid increments; 2021. Accessed June 06, 2021. https://www.ebay.com/pages/help/buy/bid-increments.html
22. Calheiros RN, Ranjan R, Beloglazov A, Rose CAD, Buyya R. CloudSim: a toolkit for the modeling and simulation of cloud resource management and application provisioning techniques. *J Softw Pract Exp*. 2011;41(1):23-50.
23. pCloud AG, pCloud; 2021. Accessed June 06, 2021. https://www.pcloud.com
24. Tchernykh A, Babenko NCM, Miranda-Lopez V. Scalable data storage design for nonstationary IoT environment with adaptive security and reliability. *IEEE IoT J*. 2020;7:10171-10188.
25. Xiong H, Huang MYX, Yu S. Unbounded and efficient revocable attribute-based encryption with adaptive security for cloud-assisted Internet of Things. *IEEE IoT J*. 2022;9(4):3097-3111.
26. Li J, Li B, Wo T, et al. CyberGuarder: a virtualization security assurance architecture for green cloud computing. *Future Gener Comput Syst*. 2012;28(2):379-390.
27. Mazur S, Blasch E, Chen Y, Skormin V. Mitigating cloud computing security risks using a self-monitoring defensive scheme. Proceedings of the IEEE National Conference on Aerospace and Electronics (NAECON); 2011:39-45.
28. Xu J, Yan J, He L, Su P, Feng D. CloudSEC: a cloud architecture for composing collaborative security services. Proceedings of the IEEE 2nd International Conference on Cloud Computing Technology and Science (CloudCom); 2010:703-711.
29. Squicciarini AC, Petracca G, Bertino E. Adaptive data management for self-protecting objects in cloud computing systems. Proceedings of the 8th International Conference on Network and Service Management; 2012:140-144.
30. Ma S, Wang Y. Self-adaptive access control model based on feedback loop. Proceedings of the International Conference on Cloud Computing and Big Data (CloudCom-Asia); 2013:597-602.

How to cite this article: Tziakouris G, Mera-Gómez C, Ramírez F, Bahsoon R, Buyya R. Market-inspired framework for securing assets in cloud computing environments. *Softw Pract Exper*. 2022;52(9):2005-2025. doi: 10.1002/spe.3113