Kuwait J.Sci., Vol.49, No.(1),January.2022,pp(1-22)

# An adaptive authentication and authorization model for service oriented enterprise computing

Mohamed Ibrahim, Beer Mohamed*, Mohd Fadzil Hassan

*Dept. of Computer & Information Sciences*
*Universiti Teknologi Petronas, 32610 Seri Iskandar, Malaysia*

**Corresponding author: bmdibrahim@gmail.com*

**Abstract**

Service oriented enterprise computing is an integration architectural style aimed to expose and consume coarse grained and fine grained modularization of business functionalities as services that are being deployed in the loosely coupled organizational environment. The web service is the implementation technology of service oriented architecture (SOA) where it is built on the existing networking and web interfacing standards as it has to use the web as a medium of communication and does not have any specialized in-built layer for security. The majority of the vendor security products in the market need specialized hardware/software components, eventually, they break the standards and principles of service oriented architecture. The traditional way of problem solving is not effective for developing security solutions for service oriented computing, as its boundaries keep expanding beyond a single organiza-tional environment due to the advent of communication and business technologies such as the Internet of Things (IoT), hyper-personalization, and edge computing. Hence, it is a mandatory entity in this digital age of enterprise computing to have a specialized authentication and authorization solution exclusively for addressing the existing security gaps in SOA in an adaptive way forward approach. In this paper, the security gaps in the existing Identity and Access Management (IDAM) solutions for service oriented enterprise computing are analyzed, and a novel intelligent security engine which is packed with extended authentication and authorization solution model for service consumption is presented. The authentication and authorization security requirements are considered as cross cutting concerns of SOA implementation and the solution is constructed as Aspect-Oriented Programming (AOP) advices, which enables the solution can be attached as a 'plug & play' component without changing the underlying source code of the service implementation. For Proof-of-Concept (PoC), the proposed authentication and authorization security model is tested in a large scale service oriented enterprise computing environment and the results have been analyzed statistically. It is evident from the results that the proposed security model addresses security issues comparatively better than existing security solutions.

**Keywords:** Adaptive security model; authentication; authorization; microservices security; SOA security.

## 1. Introduction

Enterprise computing is an information technology driven business operational model, which encompasses a myriad of related software and hardware associated with enterprise applications. The computing design solutions generally fall into two software architectural models namely, tightly coupled and loosely coupled models. The tightly coupled model promotes standalone application deployment, where the loosely coupled model brings the advantages of distributed computing. Indeed, enterprise computing is intended to be integrating such loosely coupled software components for better scalability committed availability, and efficient business operations (Lam, 2020), (Rabelo *et al.*, 2015).

The enterprise application integration (EAI) is an operational business domain driven architectural principle, which is targeted to integrate the set of inter related software applications in an enterprise. However, EAI is challenged by many different factors such as software, hardware, platform, business modeling, and very importantly security as the business data are shared and transmitted over the network (Kumar *et al.*, 2015).

Though the EAI has been achieved in many application interfacing technologies, the service oriented architecture (SOA) is a paradigm shift for application integration which decomposes the business functionalities into smaller and independent software routines, called services and these services can be consumed by the client entities as needed basis (Dhara *et al.*, 2015); (DiFrancesco, 2017); (Balalaie *et al.*, 2016). The web services are the implementation technology of SOA, and these web services can be accessed over the web using standard Internet protocols such as HTTP (HyperText Transfer Protocol) and HTTPS (Hypertext Transfer Protocol Secure). The security for service oriented architecture is defaulted to transport and network layer security, which is inadequate and obsolete for this current trend of web and communication technologies (Kulesza *et al.*, 2018); (Sfar *et al.*, 2018); (Zheng *et al.*, 2019); (Rudman & Bruwer, 2016); (Newman *et al.*, 2016); (Sirohi *et al.*, 2016). The vendor products are still evolving; however, many products breach the security standards for customization or demand to use as the whole product end-to-end, or both. Certainly, it is a must to have a comprehensive and tailorable security solu-tion for service oriented enterprise computing without breaching any existing security standards and in a cost effective manner.

As the architecture of web services is complex and loosely coupled, security considerations need to be addressed extensively at the designing stage itself, as web services need a high degree of protection. Security measures should be enforced in the web services environment to ensure that data can only be accessed by designated legitimate users, and to provide some degree of certainty on the identity of service processes when a client is about to transfer sensitive information (Masood & Java, 2015).

As per the conducted literature review, there is a lack of security solutions which are proposed for service oriented enterprise computing in adaptive way-forward methodology (Katsikogiannis *et al.*, 2016);(Li *et al.*, 2018); (Abdelrazek *et al.*, 2017); (Mohamed *et al.*, 2019) and yet to have a comprehensive authentication and authorization solution in tailorable means for preventing security attack occurrences and reducing the security risk factor on service oriented enterprise computing (Wadhwa, 2016), (Yuan *et al.*, 2016).

In this paper, a novel authentication and authorization model which is specifically constructed for securing service oriented enterprise application integration in an adaptive way forward specification is presented. For that, the security gaps on the existing authentication/authorization solutions for service oriented enterprise computing environment are analyzed in the view of available literature, where these gaps are targeted to be fulfilled as part of the proposed authentication/authorization model. A proposed solution model is developed and presented in an adaptive way forward implementation that applies a ”predict-prevent-learn” pattern for enforcing security. An end-to-end authentication and authorization flow is detailed out with adequate illustrations and sample data. Finally, the proposed solution is provided with real-time implementation of Proof-of-Concept (PoC) and the obtained data are validated through mathematical models on their correlation relationship.

The remaining sections of this research paper are organized as the motivation and context of the chosen research topic is briefed in section 2, the proposed security model is detailed out in section 3, the results from proof-of-concept (PoC) are analyzed and discussed in section 4, and section 5 concludes the paper.

## 2. Motivation and context

The service oriented enterprise computing in the architectural principles of SOA was achieved in a few implementations which include DCOM (Distributed Component Object Model), ORB (Object Request Broker), and RMI (Remote Method Invocation) (Lam, 2020), (Zimmermann *et al.*, 2018). However, these architectures work on their own defined protocols, and these protocols are specific to certain programming languages and compilers that do not support the development of distributed systems over heterogeneous platforms (Dhara *et al.*, 2015). Currently, SOA provides remedies for those issues using

common Internet protocols with the sort of web services (Rafique *et al.*, 2015), (Abuosba, 2015).

A service is a well defined, self-contained software element that does not rely on the context or state of other services (Pirnau & Botezatu, 2017). A web service achieves its purpose in an exceptionally technology-neutral manner; it offers well defined interfaces for distributed functionalities, independent of computer architecture, operating systems, and programming languages as well. In this way, webservices had emerged as a dominant model for creating and writing distributed network based business partnerships. Security is one of the key concerns when designing distributed business applications and this motivates security mandates on web-services.

The major security gaps in the existing security authentication/ authorization solutions which are being applied for service oriented architecture as per the literature review (Mourad *et al.*, 2008); (Tout *et al.*, 2015); (Azzam *et al.*, 2016); (Ayoubi *et al.*, 2012) are outlined below.

(A) Missing adaptive approach: Most of these available authentication/ authorization solutions are reactive-based, but the required one is a proactive approach that can predict and prevent security attacks. It is due to the state of these solutions that are targeted for authentication/ authorization for monolithic applications, not intended for complex service oriented enterprise computing.

(B) Breaching security standards: There are potential security models from both vendors and research proposals, however, they either result in breaching service oriented computing standards or result in another security product with their own tightly-coupled security measurements.

(C) Expensiveness on customization: The existing security solutions are yet to result in addressing the major security authentication/ authorization risks of inter-organizational service penetration, and they should be used as a 'whole' where the customization on addressing only specific security risks is expensive. The expectation of the authentication solution for service oriented enterprise computing is to have it tailorable according to the organizational domain-specific security needs and policies.

(D) Demands for additional infrastructure: Some promising vendor security products for authentication/ authorization of enterprise computing require specialized hardware/ software components which result in additional capital expenditure (CAPEX) and operating expenditure (OPEX) to the organizations. However, these solutions are focused on resolving only the specific portion of the attack vector.

(E) Product instead of the specification: There are few security solutions available as reviewed in the literature which uses the existing security standards and specifications; however, they are released to the market as products. Hence, they are not open to further modification and implementation by third parties/ researchers/ open-source communities.

(F) Restricted to the closed organizational environment: Some other available security solutions are providing good results; however, they work only in a closed-organizational environment. Hence, they are not suitable for applying this latest emerged era of application integration which spans beyond the organizational boundaries such as the Internet of Things (IoT) and edge computing.

(G) Missing cloud-native deployment: Most of the research proposals as reviewed in the literature on security for enterprise computing are developed based on the on-premise deployment architecture. It requires the underlying infrastructure, platform, and supportive software components that should be present in the organizational environment itself. As cloud computing is another era with the advent of high-speed network communication technologies which is evolving to offer anything as a service, the expected security should be capable to be offered as a 'software-as-a-service (SaaS)' pattern where the client organizations can consume these security services as-needed basis by subscription.

(H) Changes in source code: The existing security proposals for securing service oriented enterprise computing require code change on the service implementation part to embed the security-handling

code. Also, it requires additional effort to change the code to decouple the security implementation if decided. The expected security proposal should be constructed as a plug-and-play model with the minimum configuration at the underlying service implementation.

The proposed adaptive authentication/ authorization model targets to address these security gaps and verify the implementation of proposed security in a real-time SOA environment as Proof-of-Concept (PoC). Many of the security proposal as in literature either follow their predefined architectural approaches or demand for additional infrastructure, both violate the principles of service oriented architecture. The expected security solution can be added as a security layer to the current service oriented architecture and microservice architecture which should not violate the industrial security standards even the application integration is expanded to cross-organizational and cloud computing levels.

## 3. Proposed security solution model

As the number of potential threats on the cyber world is ever increasing and the attack surface has been widening, it became a business mandate to protect the resources from these security threats. The traditional way of identifying security threats and following-up perimeter defense strategies is obsolete (Dhara *et al.*, 2015), (Rudman & Bruwer, 2016), (Bherde & Pund, 2016). It should be the paradigm shift to a continuous response, which is shifting from a reactive to a proactive approach to handling and preventing security attacks.

The core elements of the proposed authentication and authorization security solution defending against authentication/ authorization based attacks of service oriented enterprise application integration are described in the followings sub-sections: the core feature of the proposed adaptive security model is highlighted in section 3.1 where the solution approach is significantly advanced from the traditional way of security implementation, and the federated authentication and authorization trust relationship among the SOA entities for accessing the exposed services in the enterprise computing environment is mathematically expressed in section 3.2, and the novel authentication and authorization security model based on the defined adaptiveness and trust relationship for consuming services are explained in section 3.3.

### 3.1 Adaptive security model

An adaptive security approach uses a combination of integration strategies and tactics to protect data and application system resources of dynamically changing enterprise computing environment to stay ahead of any potential threats with the ability to adhere to suit the organizational security conditions in preventing enablement of security attacks as agile as possible. This adaptive security architecture is focused to sense, detect, and respond to the prevention of threats before the attackers make use of threat vulnerabilities on performing attacks.

The proposed adaptive security model for authentication and authorization applies the Predict-Prevent-Learn pattern, as depicted in Figure 1. The ‘predict’ stage assesses the security risks, anticipates security threats, and refines security postures for the enterprise computing systems. The ‘prevent’ stage hard-ens the enterprise computing systems defending against security attacks by implementing the developed security algorithms without jeopardizing the underlying security standards. The knowledge base of the threat prediction is sourced by the security analytic processes, which detect anomalous patterns of user activities with an application system.

One of the core tenets of software development methodology is to enforce segregation of concerns. This tenet states that the concerns as the functional behaviors should be separated into self-matured modules so that any changes to these concerns will be performed in one place resulting a single source of truth (SSOT) for the enterprise applications and their underlying data (Shashwat *et al.*, 2018), (Liu *et al.*, 2019). The object oriented Programming (OOP) principles have been used widely for segregating most of the concerns in the form of classes and objects, implicitly along with inheritance and polymorphism. However, there are some concerns, which are cross-cutting in nature, that are very difficult to model using OOP.

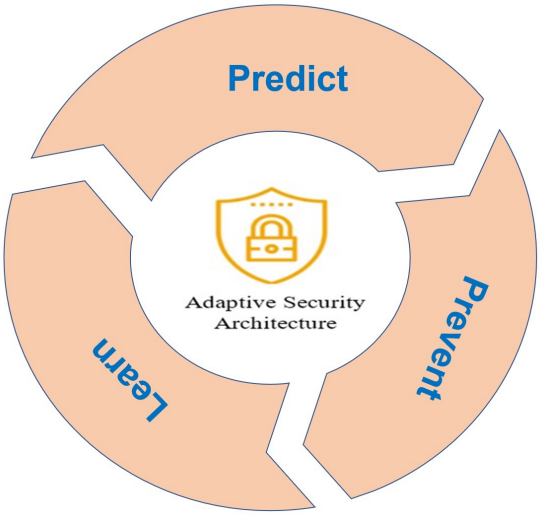



**Fig. 1.** Adaptive security architecture.

These cross-cutting concerns are spread across several code modules of the SOA applications and require the defined functionalities should be applied to the 4concerned parts. Authentication/ authorization is one of such concerns that should be applied across the code modules.

As a novelty, Aspect-Oriented Programming (AOP) deployment strategies had been followed up for the implementation of authentication/ authorization security model instead of regular object oriented programming, which separates the non-functional security implementation from the underlying functional business logic of the services. Hence, the proposed security model will work on the 'plug & play' concept in AOP mode without any code alternation.

### 3.2 Establishing the trust relationship among SOA entities

In service oriented architecture (SOA) terms, web services are the implementation part of SOA where SOAP Web services are coarse-grained service model that works on Simple Object Access Protocol (SOAP) and Microservices are the fine grained service model that works on REST (REpresentational State Transfer) principle (Halili & Ramadani, 2018). The service provider (server) is an entity in SOA which exposes the web services and the service requester (client) is another entity in SOA which consumes the exposed services based on its business computational need. Thus, the SOA forms a complex network of these client and server entities which can span beyond the organizational boundaries on service consumption (Yahyaoui *et al.*, 2020), (Aljazzaf, 2015).

Today, enterprises have many choices when it comes to application deployment and data storage, the traditional on-premise where the enterprises own the infrastructure and purchase software licenses for local installations, and modern cloud computing where enterprises subscribe to what they need and pay as they use basis. In this fast-growing era of advanced networking and cloud computing, anything-as-a-service (XaaS) over the Internet is possible.

In addition to the Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) cloud models, the proposed security solution can be provisioned as a service, Security as a Service (SECaaS), where the entire security solution will be deployed and maintained in clouds and the client enterprises can consume these security services by subscriptions without any additional investment on infrastructure, platform, and maintenance for security. The proposed security solution on this SECaaS-based model will benefit the cloud computing features such as scalability, robustness, demanded performance, cost effectiveness, and assured service availability. At the same time, the openness of cloud computing should not bring any security vulnerabilities for the deployed security solution that will not be the case of an on-premise deployment. The main concern of the cloud computing environment for application integration is trust establishment, which has been implemented in the proposed security with the below mathematical relationship among the SOA entities.

The relationship between two entities in SOA named $x_1$ and $x_2$ from the set of entities named S is stated as a binary relation $R \subseteq S \times S$. The trust relation $x_1 R x_2$, $(x_1, x_2) \in R$ represents a link between the

entities $x_1$ and $x_2$ in common ground for the defined trust graph $G(S, R)$ where the following properties are set to true.

1. $R$ is reflexive: An entity trusts to itself

   $\forall x \in S,\ xRx$

2. $R$ is not symmetric: When an entity trusts another entity within a specific context, then the vice versa cannot become true only because of the forward trust establishment

   $\exists x_1,\ x_2\ \in S,\ \ x_1 R x_2 \nRightarrow x_2 R x_1$

3. $R$ is not transitive: When an entity trusts another entity which in turn trusts another entity, then the trust relationship between the first and third entity cannot become true only because of the continued chain of trust establishment among entities

   $\exists x_1,\ x_2,\ x_2 \in S,\ \ x_1 R x_2 \wedge x_2 R x_3 \nRightarrow x_1 R x_3$

### 3.3 Proposed authentication and authorization security model

The core process of proposed authentication and authorization is based on open authorization standard (OAuth v2.0) (Fett *et al.*, 2016), and the API led connectivity design (Sun *et al.*, 2016) is chosen for solution development and implementation as it is the very latest architectural design methodology for exposing enterprise computing services as APIs through API Gateway (Hu *et al.*, 2014).

The complete authentication and authorization part for accessing the exposed business services are provisioned by a newly introduced intellectual software component, named Intelligent Security Engine (ISE), in the proposed security architecture. The high-level flow of authentication and authorization on accessing microservices is given in Figure 2.

In a nutshell, the client (service consumer) establishes the initial communication with API Gateway with its own client credentials. Then, API Gateway forwards the client's request to the proposed security engine, ISE. The API Gateway will act as an enterprise broker for every client who wants to interact with ISE and API respectively for a trusted security and business services. The ISE is responsible for generating and handling Access Token (AT) for each of the clients, where the clients start consuming the APIs directly without reauthentication. The interactions among the microservices are provisioned by a special access token, named Internal Access Token (IAT). Every microservice in the deployed boundary trusts the IAT and lets other microservices invoke it for consumption without any additional authentication.

As only approved standards and specifications are applied as and when required in the construction of the proposed solution, API-led connectivity design is selected which commonly uses microservices for exposing APIs. Hence, the design flow is portrayed as it uses microservices however, the ISE solution component is open and capable of handling coarse-grained services (SOAP web services) also in addition to the fine grained services (Microservices) (Beer & Hassan, 2018), (Ibrahim & Hassan, 2015).

The elements of the proposed authentication and authorization security solution are explained in the following sub-sections.

#### 3.3.1 API-led connectivity design

As per the API-led connectivity, APIs are classified into three categories as below.

- System APIs: These APIs are more towards accessing internal backend systems of the enterprise, includes legacy systems, mainframe systems, coarse-grained web services, databases, mainframe systems, and other EAI applications through their native protocols of communication such as RMI, SOAP, TCP/IP, and CORBA; and offers a uniform API for its consumers, usually in the form of REST endpoint.

- Process APIs: These APIs take the intermediator role between System APIs and Experience APIs by providing primarily orchestration services. They do not interface backend systems directly but

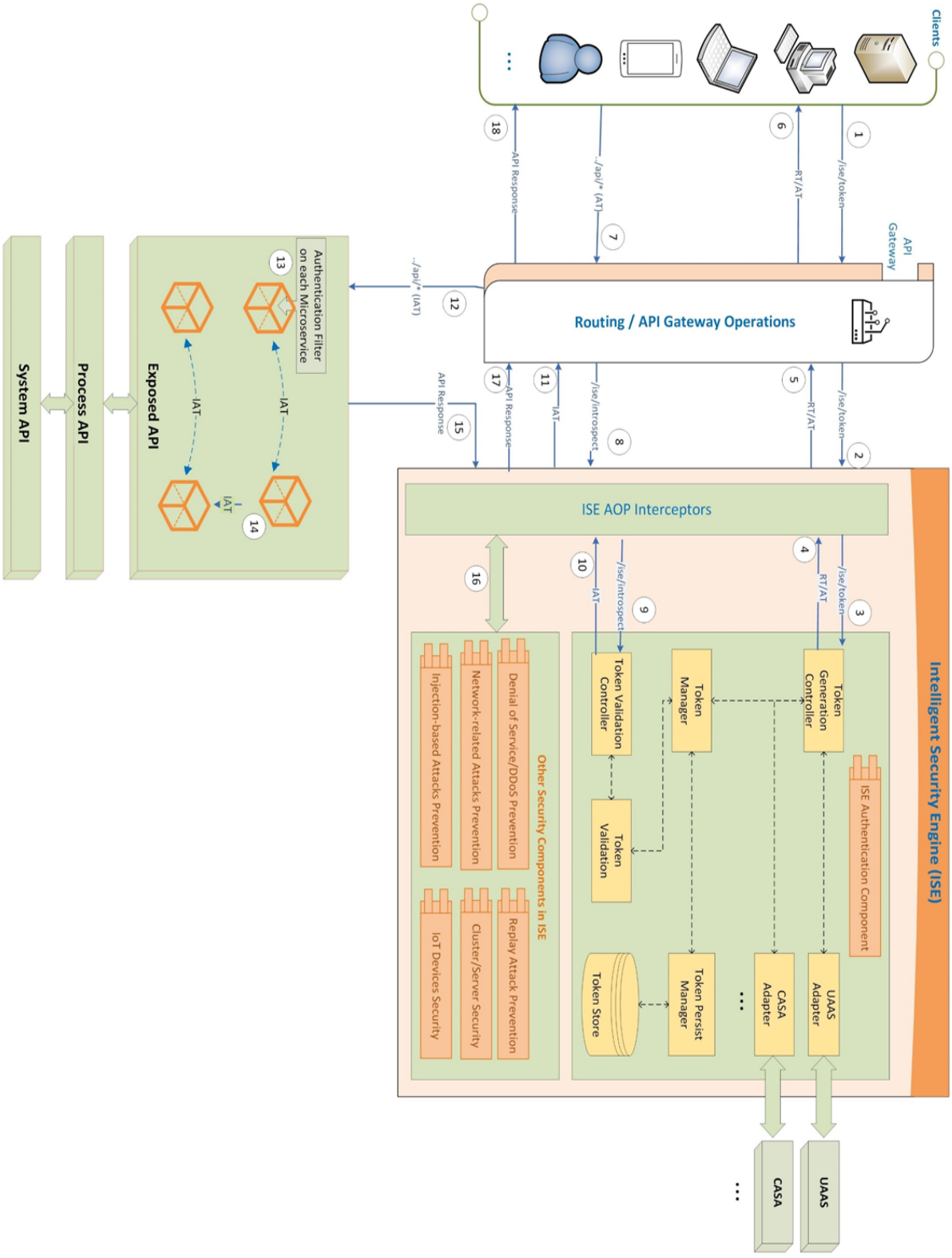



**Fig. 2.** The proposed authentication and authorization security model.

through system APIs. The process APIs will handle only one data format for interfacing, usually REST, as both system APIs and experience APIs can understand.

- Experience APIs: These APIs are more towards exposing services to the outside world through API Gateway. Instead of having separate point-to-point integrations between service providers and clients for each channel, it is an obvious choice of interfacing through a common gateway, which additionally does URL rewriting, request routing, and protocol translation.

#### 3.3.2 Actors in the proposed model

The actors and their roles in the proposed authentication and authorization security model are listed in Table 1.

**Table 1.** Actors in the proposed model.

| Actor | Role / Description |
|---|---|
| Resource Owner | The owner of the services that are deployed in the resource servers. |
| Resource Server | The server where the services are being deployed. |
| Client/Service Consumer | Applications or the "things" (computing devices) in the Internet of Things (IoT) that consume the services. |
| Intelligent Security Engine (ISE) | The core part of the proposed security solution that extends authentication/ authorization functionalities based on intelligent threat prevention functionalities. |
| API Gateway | An entry point for all the clients for accessing the deployed services. |

#### 3.3.3 Types of tokens and their functionalities

Three types of tokens are being applied in the proposed authentication/ authorization security model. The applied tokens and their purpose of implementation are given in Table 2.

#### 3.3.4 Types of authentication levels

Each of the defined access tokens as given in Table 2 has a pre-defined authentication level that should be satisfied before obtaining the corresponding tokens. The permitted list of access levels for this proposed security solution is given in Table 3.

#### 3.3.5 Event flow on the proposed model

Event streaming solution became the de-facto approach for service-to-service communications, as it brings several advantages over direct communication especially on non-blocking, vertical scaling, and self-resiliency expectations. With this context, the series of events as steps on the proposed authentication and authorization process flow is given in Table 4.

#### 3.3.6 Authentication and authorization process

The complete authentication and authorization flow on accessing the microservices from the outside clients and inter process communication among the microservices are governed by the novel proposed component, named Intelligent Security Engine (ISE). The ISE adheres to the OAuth v2.0 (Fett *et al.*, 2016) standard for authorization and equipped with custom-built security components such as User Authentication and Authorization Service (UAAS), Context-Aware Scalable Authentication (CASA), token

**Table 2.** Types of tokens and their functionalities.

| Token | Purpose |
|---|---|
| Access Token (AT) | • The clients use these tokens for accessing microservices.<br>• These tokens are short-lived, typically for a few minutes; however, can live up to a few hours.<br>• Once these tokens are issued, they cannot be revoked, must wait for them to be expired themselves.<br>• An authentication level is associated with each token, such as 1FA (one factor authentication -1FA, two-factor authentication -2FA, etc.) |
| Refresh Token (RT) | • These tokens are granted along with Access Tokens.<br>• These tokens are long-lived.<br>• Typically used for obtaining a fresh Access Token if the previously issued one is expired.<br>• These tokens can be revoked at a point in time. |
| Internal Access Token (IAT) | • These tokens are a specialized form of Access Tokens, which are used for microservices inter-process communications, an invoking microservice trusts the calling microservice in a cluster with an IAT.<br>• These tokens are valid only for a single request-response cycle. |

management entities, and constructed security threat prevention entities, where these components are seamlessly plugged in to the SOA implementations through AOP interceptors (Hassan, 2016).

In the proposed design as depicted in Figure 2, the ISE is separated-out for dedicated deployment aimed to serve as a centralized component for all the domain-specific SOA entities within an organization. However, the ISE can be deployed within the same server in another instance where API Gateway is running, or even it can be plugged in to the API Gateway itself as a security component if the vendor of API Gateway supports it to do.

Any external client who wants to access Experience APIs must be first authenticated with ISE by fulfilling the defined authentication level based on the severity of underlying services being accessed, for obtaining the Access/Refresh Token pair. Any internal client (deployed at the same cluster of System API type) can obtain the Access/Refresh Token pair by directly passing its client-id and its pre-defined client secret to ISE. Then, either the external client or the internal client can invoke the services bypassing the Access Token without any further authentication/ authorization checks. As the Access Tokens are short-lived, the clients can get new Access Tokens from ISE by submitting the corresponding Refresh Tokens, where the ISE will validate the client profiling before providing new Access Tokens to avoid any token theft and misuse attempts by malicious clients.

This proposed security solution uses the JSON Web Token (JWT) standard for generating required tokens, to have the consistency of using only well defined standards instead of other custom ways of generating tokens. JWT is an open standard for data transformation between two parties securely with digitally signed JSON (JavaScript Object Notation) object. At present, the JWT is being signed using either hash-based algorithms or public-key infrastructure (PKI).

A JWT token consists of three parts, where are separated by dots as below.

```
HEADER.PAYLOAD.SIGNATURE
```

where the HEADER part consists of BASE64URL encoded version of two values: (i) the code of the signing algorithm, which is being used for digital signatures, such as RSA, SHA256, and HMAC,

**Table 3.** Types of authentication levels.

| Auth Type | Auth Level | Description |
|---|---|---|
| No Authentication | -1 | This is not a recommended option that no authentication is required to access the services. |
| Registered Device | 0 | Only pre-registered devices can access resources. This type of authentication level is usually applied for meta-data/ pre-login access, for example, deep linking and soft-token approval requests on registered mobile devices for approving any banking transaction. |
| 1FA | 1 | ONE Factor Authentication (1FA) which is applied to accessing any normal resources, but with authentication. The 1FA can be achieved in many ways, usually with the following.<br>(a) Username & Password<br>(b) Authentication passcode, i.e., client secret<br>(c) Any biometric authentication such as fingerprint scanning, facial recognition, and voice recognition. |
| 2FA | 2 | Two Factor Authentication (2FA), also known as Step-up Authentication, where the clients are requested to do one more authentication in addition to the 1FA. This authentication level is applied for accessing highly sensitive resources or performing highly impacting transactions. For example, the user may be promoted for 1FA while performing fund transfer requests in the banking use case, and the same user will be prompted for 2FA if the transferring fund amount is very high which is not considered normal and is a high-risk transaction. The medium of 2FA authentication should have differed from 1FA, for example, the same SMS OTP cannot be applied for both 1FA and 2FA. The 2FA can be achieved in many ways, including the following.<br>(a) One Time Password (OTP)<br>(b) Hard Token via Token Card<br>(c) Soft Token via Smartphone<br>(d) Any biometric authentication such as fingerprint scanning, facial recognition, and voice recognition. |

Step 1: Call ISE via API Gateway to get Access and Refresh Tokens by passing with the required grant (username-password/ OTP/ client secret token)

$$\text{Client } C_i \xrightarrow[client\ validation]{credentials} \text{API Gateway APIg}$$

Step 2: API Gateway forwards the token call to ISE

$$\text{APIg} \xrightarrow[filtration]{credentials} \text{ISE}$$

Step 3: The configured ISE interceptor captures the request and passes it to ISE Authentication component, after successful validation of threat scoring/ risk assessment

$$\text{ISE.Interceptor} \xrightarrow[threat\ scoring]{risk\ assessment} \text{ISE.TokenGen}$$

Step 4: The Token Generation Controller communicates with the required external systems as required for validating the client's authentication/ authorization, generates the Access/ Refresh Tokens (AT/RT) and returns them to the ISE Interceptor for post-processing. Scope of the access on the requested request is added to access token in this step.

$$\text{ISE.Interceptor} \xleftarrow[token\ generation]{RT/AT} \text{ISE.TokenGen}$$

Step 5: ISE interceptor will communicate with other security components (implementation of pre-written security algorithms) for security screening and passes the tokens to API Gateway.

$$\text{APIg} \xleftarrow[token\ screening]{RT/AT} \text{ISE.Interceptor}$$

Step 6: API Gateway forwards the RT/AT to the originated client. Whenever the Access Token is expired, the client can use Refresh Token to get fresh RT/AT pair.

$$C_i \xleftarrow[tokens]{RT/AT} \text{APIg}$$

Step 7: Client calls the required Microservice API by passing obtained Access Token in the authorization header.

$$C_i \xrightarrow[request\ header]{AT} \text{APIg}$$

Step 8: API Gateway inspects the validity of AT with ISE, where ISE acts as the authorization checkpoint.

$$\text{APIg} \xrightarrow[intropect]{AT} \text{ISE.Interceptor}$$

Step 9: The ISE's internal component Token Validation Controller validates the given AT by communicating with internal and external authentication/ authorization components.

$$ISE.Interceptor \xrightarrow[token\ validation]{AT} ISE.TokenManager$$

Step 10: Upon the validation is successful, ISE returns Internal Access Token (IAT) which contains standard claims along with user-defined data (additional data required for a microservice for its business operation) which are specific for each service call. Upon the validation is failed, ISE returns an error code and error description.

$$\text{ISE.Interceptor} \xleftarrow[internal\ access\ token]{IAT} \text{ISE.TokenManager}$$

Step 11: In the case of validation is successful, ISE forwards the IAT to API Gateway by appending security controls; returns error code and error description otherwise.

$$\text{APIg} \xleftarrow[security\ controls]{IAT} \text{ISE.Interceptor}$$

Step 12: Once the API Gateway receives the IAT, then it routes the incoming client request to the corresponding microservice API (deployed on clusters) by passing IAT in the authentication header and removing AT from the header.

$$\text{APIg} \xrightarrow[routing]{IAT} \text{Microservice}(\mu\text{s})$$

Step 13: Each request is intercepted by the Authentication Filter embedded with the microservices. This authentication filter does self-validation of IAT using the ISE Authentication Server's obtained public key if PKI is used, or self-validated through secret key if hashing method is applied. As IAT's scope is restricted to only internal cluster and will not be exposed to outside clients, it is fair enough to do self-validation.

$$\text{Microservice.AuthFilter} \xrightarrow[self\ validation]{IAT} \text{Authentication Server}$$

Step 14: Each microservice can consume any other microservices directly, which are deployed in the same cluster zone using the same IAT.

$$[\mu\text{s}]_{\text{i}} \xrightarrow[invokes]{IAT} [\mu\text{s}]_{\text{j}} ,$$

where $\{\text{i},\text{j}\} \in \text{n}$; n represents the total number of exposed microservies

Step 15: The API response will be intercepted by ISE for security screening.

$$\mu\text{s} \xrightarrow[security\ screening]{API\ Response} \text{ISE.Interceptor}$$

**Table 4.** A series of steps on the proposed authentication and authorization event flow.

| | |
|---|---|
| Step 16: | ISE will check for all the configured security vulnerabilities (as screened on request processing session) and appends security-related data on the response.<br>ISE.Interceptor $\underset{vulnerabilities\ check}{\overset{API\ Response}{\longleftrightarrow}}$ ISE.SecurityComponents |
| Step 17: | The interpreted API response is passed to API Gateway.<br>APIg $\underset{verification}{\overset{API\ Response}{\longleftarrow}}$ ISE.Interceptor |
| Step 18: | API Gateway routes the received API response to the respective client. Client redirection will be performed by API Gateway as and when needed. In case of any error code is returned by ISE for any of these given communication cycles, API Gateway will send failure notification to the client.<br>$C_i$ $\underset{routing}{\overset{API\ Response}{\longleftarrow}}$ APIg |

and (ii) type of the token, which is always 'JWT' in this case. A sample header part, before encoding, is given below.

```
{
"alg":  "RS256",
"typ":  "JWT"
}
```

The PAYLOAD part consists of a BASE64URL encoded version of the below-listed claims and the required application data to share between the services.

- jti: Represents the unique token identifier for the token i.e., JWT_ID
- client_id: Client's ID who invokes the service
- iat: Issued At Time, UNIX timestamp when the token was issued
- exp: Expiration time, UNIX timestamp when the token will expire
- sub: Subject, purpose and for whom the token is being created
- iss: Issuer, server ID who is issuing the token
- scope: List of scopes for which the token is being issued
- aud: Audience, IDs of the client/server for whom token is being issued
- grant_type: Access grant type used for generating the token

A sample payload, before encoding, is given below.

```
{
"jti":  "47316mnik660f9243er",
"client_id":  "ONLINE_BANKING",
"iat":  1564566181,
"exp":  1564566847,
"sub":  "USER1",
"iss":  "ISE",
"scope":  ["refreshToken"],
"aud":  "ONLINE_BANKING ",
"grant_type":
"client_credentials"
}
```

The SIGNATURE is the BASE64URL encoded string of the following.

```
Algorithm (
base64UrlEncode(HEADER) + "."
+ base64UrlEncode(PAYLOAD),
credential_details)
```

The signature part is generated by signing the encoded string of header, the encoded string of payload, and the credential details, using the algorithm specified in the header (header.alg) and then encoding the resultant string with BASE64.

For example, if the PKI-based RSA algorithm is chosen, then the signature part is the encoded string of the following.

```
RSASHA256 (
base64UrlEncode(header) + "."
+ base64UrlEncode(payload),
public_key, private_key)
```

If the hash-based HMAC (Hash-based Message Authentication Code) algorithm is chosen, then the signature part is the encoded string of the following.

```
HMACSHA256 (
base64UrlEncode(header) + "."
+ base64UrlEncode(payload),
secret)
```

In the case of PKI is chosen, the ISE will maintain a private/ public key pair, where the private key is kept confidential and stored in the ISE authentication server, and the public key will be shared to the SOA environment via REST endpoint or other secured channels. The ISE uses the private key for signing a generated token, and this token will be validated by the authentication filer on each microservice at the MSA environment. In the case of hashing is chosen, both the server and client agree on a 'secret string', known as a client secret, which will be used for authorization purposes.

## 4. Results and validation

To experiment with the proposed authentication and authorization solution, a real-time service oriented computing environment of the financial domain is chosen. Mostly real-time banking data are used for the experiment as these data are closely tied-up with the underlying service oriented enterprise computing environment. For testing negative cases, data virtualization tools such as MATLAB and SOAP UI are used, otherwise, it will severely harm the real-time environment.

The test environment for the Proof-of-Concept (PoC) implementation of the proposed security solution is given in Table 5.

**Table 5.** Proof-of-Concept (PoC) test environment.

| Entity | Environment |
|---|---|
| PoC Environment | Banking domain |
| SOA Implementation | SOAP web services<br>RESTful web services |
| Data | Real-time banking data<br>Simulated test data [MATLAB, SOAP UI] |
| Vendor | Security Solutions as procured and used by the organization |
| Processors | Intel(R) Xeon(R) @ 2.30GHz; 16 CPUs; 64 bit |
| Deployment Environment | Physical Machine<br>Virtual Machine (VM)<br>Public Cloud (AWS) |
| Operating System | Red Hat Enterprise Linux (RHEL) v7.x |
| Source code implementation | Java 8; Spring Framework |
| Database | Object Relational Mapping (ORM)<br>Oracle RDBMS; MongoDB NoSQL |
| Programming Model | object oriented Programming (OOP)<br>Aspect-Oriented Programming (AOP) |
| Reference Architecture Standards | World Wide Web Consortium (W3C)<br>Organization for the Advancement of Structured Information Standards (OASIS)<br>Internet Engineering Task Force (IETF)<br>Payment Card Industry Data Security Standard (PCI DSS)<br>National Institute of Standards and Technology (NIST) |

As obtained from the conducted proof of concept (PoC), the statistical data on authentication/ authorization based security attack and prevention on service oriented enterprise computing environment by (i) defaulted security solutions, (ii) different vendor security products as implemented by the organization in which the PoC was conducted, and (iii) the proposed security solution as implemented in the PoC enterprise computing environment is given in Table 6.

The collected data have the core characteristics and features of real-time SOA implementation, include (a) Accuracy and Precision: the exactness of the data is achieved with a real-time test environment,(b) Legitimacy and Validity: data boundaries are preset to SOAP and RESTful web services, (c) Reliability and Consistency: almost the same data source is used for the entire analysis without any contradiction or variance,(d) Timeliness and Relevance: data were collected at the right moment in time, which includes peak, medium, and low transactions, (e) Completeness and Comprehensiveness: incomplete and inaccurate data were completely avoided by excluding them from result analysis, (f) Availability and Accessibility: the primary data cannot be disclosed as they are very sensitive and real-time financial data, however, those data are accessed within the organizational environment for PoC purposes by masking the most sensitive information such as credit card numbers, customer account numbers, and account balances, and (g) Granularity and Uniqueness: The obtained experimental data were never manipulated even the degree of security will vary by one organization to another organization depending upon the severity of data and business domain.

The ratio values in columns A, B, and C represents the ratio between exposure of attacks and prevention of attacks respectively on the defaulted security, vendor security, and proposed security. For example, 0.60 : 0.40 represents the failure of protecting the underlying assets from the potential attacks is 60% whereas the protection from the security attacks is 40%. Only for securing from broken access control, the proposed security solution achieves protection near to vendor security solution; other than that, the proposed security solution provides 37.36% higher security as an average as compared to ven-dor security solutions. It is very clear from the analysis of the result that the proposed security superiors in providing security for the service oriented enterprise computing environment when compared to the vendor and defaulted security solutions.

Besides, the proposed security solution is completely software-based and does not demand any specialized hardware or vendor software. The proposed solution can be customizable and released as tailored components on an as-needed basis. Being the solution is proposed as a generic specification, it is open for any third party such as open-source communities, researchers, consortiums, and vendors who can implement and release these components as software packages in any modern programming language. Hence, it is certainly a cost effective solution too when compared to vendor products where the vendors either demand to consume their products as a whole or charge for the higher cost for customizations. The graphical representations of the interpreted data as observed in Table 6 is portrayed in Figures 3-5.

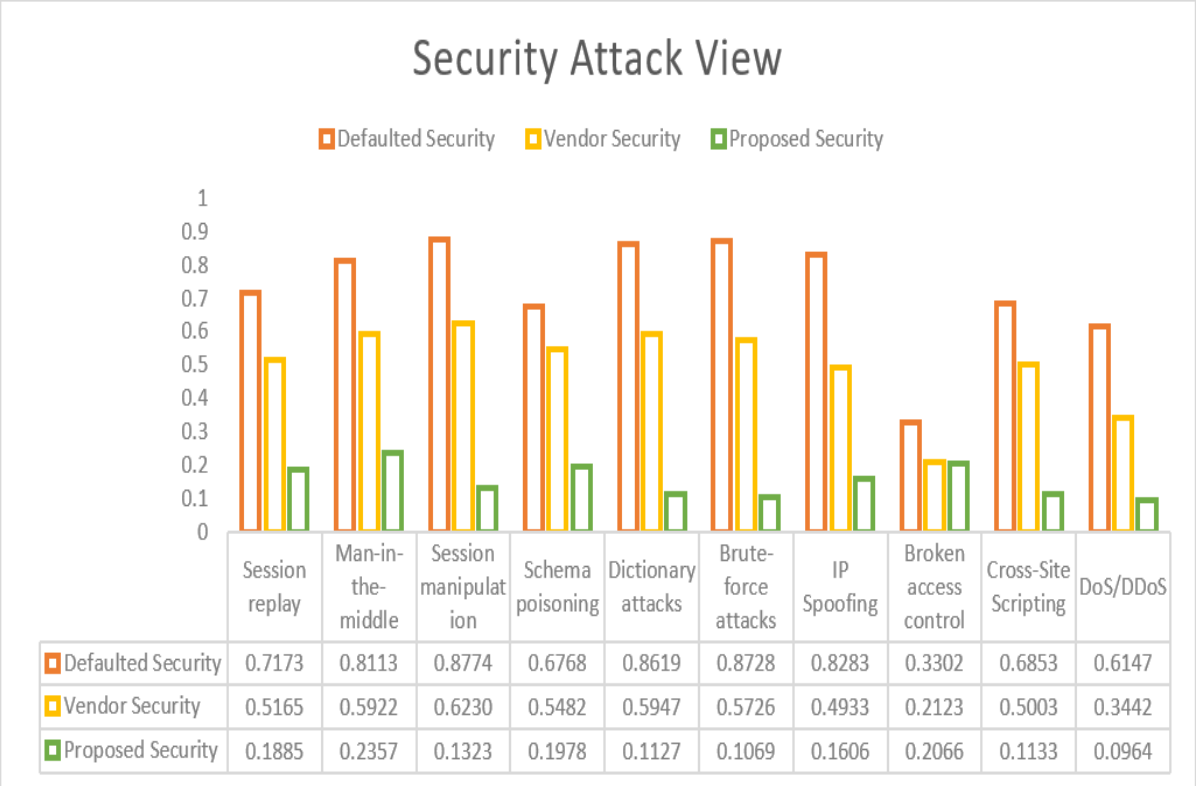
The attack requests, the requests for accessing the test services in the chosen service oriented computing environment which are intended to certainly perform authentication/authorization based attacks, are mixed with the legitimate service-requests to identify the sensitivity and false-positive rate (FPR) of the test result. The sensitivity (S) is derived as below, where AR is the number of detected attack requests and TAR is the total number of attack requests triggered to the testing services over a period of time.

$$S = \frac{\text{AR}}{T_{AR}} \tag{1}$$

The false-positive rate (FPR) is measured by how many attack requests were treated as legitimate requests over the total number of attack requests.

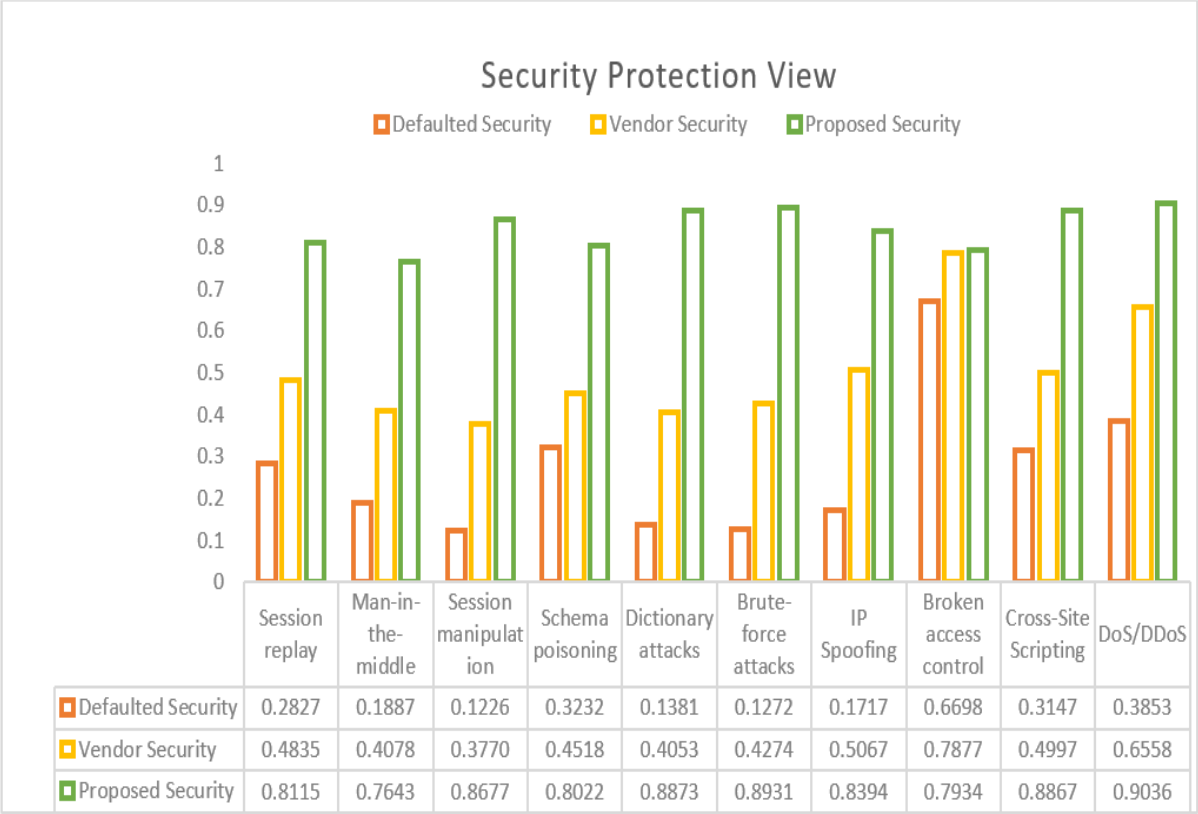
$$FPR = \frac{R}{T_{AR}} \tag{2}$$

In the conducted PoC, the observed sensitivity rate is above 85% and the false positive rate is less than 15%. The sensitivity and FPR are having an indirectly proportional relationship; and these rates can




| | Session replay | Man-in-the-middle | Session manipulation | Schema poisoning | Dictionary attacks | Brute-force attacks | IP Spoofing | Broken access control | Cross-Site Scripting | DoS/DDoS |
|---|---|---|---|---|---|---|---|---|---|---|
| Defaulted Security | 0.7173 | 0.8113 | 0.8774 | 0.6768 | 0.8619 | 0.8728 | 0.8283 | 0.3302 | 0.6853 | 0.6147 |
| Vendor Security | 0.5165 | 0.5922 | 0.6230 | 0.5482 | 0.5947 | 0.5726 | 0.4933 | 0.2123 | 0.5003 | 0.3442 |
| Proposed Security | 0.1885 | 0.2357 | 0.1323 | 0.1978 | 0.1127 | 0.1069 | 0.1606 | 0.2066 | 0.1133 | 0.0964 |

**Fig. 3.** Security attack view on defaulted, vendor, and proposed security.




| | Session replay | Man-in-the-middle | Session manipulation | Schema poisoning | Dictionary attacks | Brute-force attacks | IP Spoofing | Broken access control | Cross-Site Scripting | DoS/DDoS |
|---|---|---|---|---|---|---|---|---|---|---|
| Defaulted Security | 0.2827 | 0.1887 | 0.1226 | 0.3232 | 0.1381 | 0.1272 | 0.1717 | 0.6698 | 0.3147 | 0.3853 |
| Vendor Security | 0.4835 | 0.4078 | 0.3770 | 0.4518 | 0.4053 | 0.4274 | 0.5067 | 0.7877 | 0.4997 | 0.6558 |
| Proposed Security | 0.8115 | 0.7643 | 0.8677 | 0.8022 | 0.8873 | 0.8931 | 0.8394 | 0.7934 | 0.8867 | 0.9036 |

**Fig. 4.** Security protection view on defaulted, vendor, and proposed security.

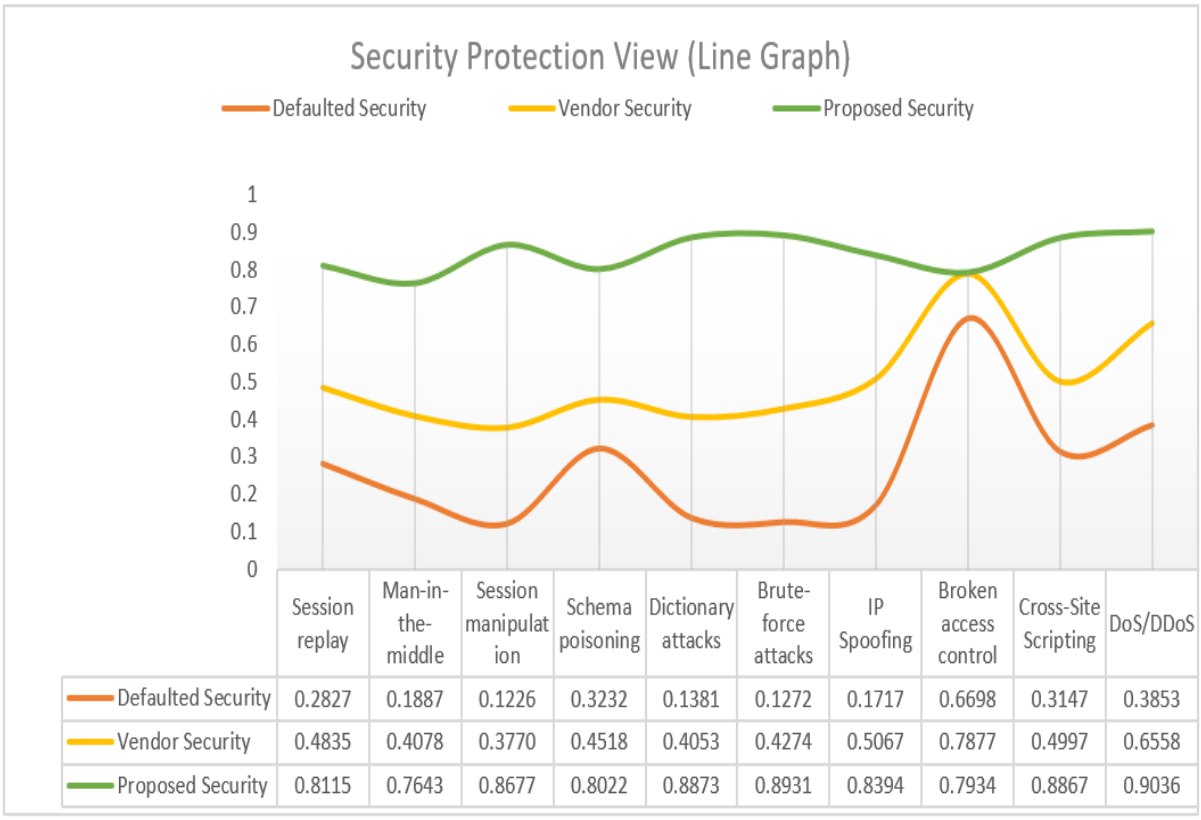



**Fig. 5.** Security protection view on defaulted, vendor, and proposed security in the line graph.

**Table 6.** Statistical analysis on PoC result.

| S.No. | Threat Ti | A Defaulted Security | B Vendor Products | C Proposed Security |
|---|---|---|---|---|
| 1 | Session replay | 0.7173 : 0.2827 | 0.5165 : 0.4835 | 0.1885 : **0.8115** |
| 2 | Man-in-the-middle | 0.8113 : 0.1887 | 0.5922 : 0.4078 | 0.2357 : **0.7643** |
| 3 | Session manipulation | 0.8774 : 0.1226 | 0.6230 : 0.3770 | 0.1323 : **0.8677** |
| 4 | Schema poisoning | 0.6768 : 0.3232 | 0.5482 : 0.4518 | 0.1978 : **0.8022** |
| 5 | Dictionary attacks | 0.8619 : 0.1381 | 0.5947 : 0.4053 | 0.1127 : **0.8873** |
| 6 | Brute-force attacks | 0.8728 : 0.1272 | 0.5726 : 0.4274 | 0.1069 : **0.8931** |
| 7 | IP Spoofing | 0.8283 : 0.1717 | 0.4933 : 0.5067 | 0.1606 : **0.8394** |
| 8 | Broken access control | 0.3302 : 0.6698 | 0.2123 : 0.7877 | 0.2066 : **0.7934** |
| 9 | Cross-Site Scripting (XSS) | 0.6853 : 0.3147 | 0.5003 : 0.4997 | 0.1133 : **0.8867** |
| 10 | Denial of Service (DoS)/DDoS | 0.6147 : 0.3853 | 0.3442 : 0.6558 | 0.0964 : **0.9036** |
| **Mean value in ratio** | | 0.7161 : 0.2839 | 0.4972 : 0.5028 | 0.1481 : **0.8519** |

be further enhanced with deep learning algorithms on the prediction of security attacks, however, this approach will demand additional computational time which will certainly result in a breach of service oriented integration architectural standards. Considering these facts, the research is ended with these higher levels of sensitivity and FPR without jeopardizing the service consumption experience.

The correlation coefficient (r) identifies how two variables change together on the unbounded magnitude of data. The Pearson correlation is a preferred and most used technique for measuring the relationship strength of relative movements on two correlating variables. The value of the correlation coefficient always ranges between -1.0 and 1.0, where +1 denotes a strong positive relationship, -1 denotes a strong negative relationship, and the value at zero represents no linear relationship. The same Pearson correla-tion statistical method is applied to the obtained PoC data to measure the degree of relationship between attack requests and attack-protection with each type of the chosen security threats, as given in Table 7. The applied Pearson correlation coefficient formula is,

$$r = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \tag{3}$$

**Table 7.** Relating the obtained result with the correlation coefficient.

| S.No. | Threat (Ti) | Correlation Coefficient (r) |
|---|---|---|
| 1 | Session replay | 0.9613 |
| 2 | Man-in-the-middle | 0.8902 |
| 3 | Session manipulation | 0.9706 |
| 4 | Schema poisoning | 0.9807 |
| 5 | Dictionary attacks | 0.9698 |
| 6 | Brute-force attacks | 0.9603 |
| 7 | IP Spoofing | 0.9755 |
| 8 | Broken access control | 0.9012 |
| 9 | Cross-Site Scripting (XSS) | 0.9260 |
| 10 | Denial of Service (DoS)/DDoS | 0.9170 |

As an industrial standard, the correlation coefficient with a value less than +0.8 is considered as nonsignificant for positive relationship cases. In the conducted PoC, none of the attack surfaces obtained a correlation coefficient value less than +0.8, which means the obtained PoC data in each threat type are strongly correlated.

## 5. Conclusion

Enterprise computing enables integration on business facilitation services by linking the application modules that are running on the different platforms. These platforms have several authentication and authorization limitations in providing service integration as these services are orchestrated through standard internet protocols such as HTTP and HTTPS. The current state of the art on service oriented architecture (SOA) does not achieve a comprehensive identity and access management (IDAM) security. The existing security solutions, as part of in-built systems security, security products from vendors, and research proposals on authentication/authorization are not able to result in the expected level of application security for SOA communications. The current security solutions have only limited insight into the particular se-curity concerns of service oriented computing architecture. However, a comprehensive tailored security solution is required without breaching the security standards of service oriented computing.

In this paper, the existing security gaps related to the authentication and authorization part of service oriented enterprise application integration are discussed along with their impact on identity disclosure. A novel authentication and authorization security component named “Intelligent Security Engine (ISE)” is proposed in an adaptive way forward specification that can be easily pluggable into the existing implementation of service oriented architecture, either SOAP based or RESTful web services. The proposed

security solution was tested in a large-scale banking environment with real-time data and the interpreted result shows that the proposed security solution outperforms the default and vendors' products. Correspondingly, the solution is proposed as a specification, not as a product, so that it opens room for customization and enhancement which is a novel feature of the proposed model.

## References

**Abdelrazek M, Grundy J,** & **Ibrahim A (2017).** Adaptive Security for Software Systems. In *Managing Trade-Offs in Adaptable Software Architectures*, (pp. 99–127). Morgan Kaufmann, United States.

**Abuosba KA (2015).** Multiagent trust management of web services the asynchronous computing environment profile unification methodology. In *Proceedings of the IEEE World Congress on Internet Security (WorldCIS)*, (pp. 55–59).

**Aljazzaf ZM (2015).** Modelling and measuring the quality of online services. In *Kuwait Journal of Science, vol. **42**(3)*, (pp. 55–59).

**Ayoubi S, Mourad A, Yahyaoui H, & Otrok H (2012).** A novel aspect-oriented BPEL framework for the dynamic enforcement of web services security. In International Journal of Web and Grid Services, (pp. 361-385).

**Azzam M, Tout H, Talhi C, Otrok H, & Yahyaoui H (2016).** From model-driven specification to design-level set-based analysis of XACML policies. In Journal of Computers and Electrical Engineering, (pp. 65-79).

**Balalaie A, Heydarnoori A,** & **Jamshidi P (2016).** Microservices architecture enables DevOps: Migration to a cloud-native architecture. In *IEEE Software, vol. **33**(3)*, (pp. 42-52).

**Beer MI,** & **Hassan MF (2016).** Adaptive security architecture for protecting RESTful web services in enterprise computing environment. In *Service-Oriented Computing and Applications, vol. **12**(2)*, (pp. 111-121).

**Bherde GP,** & **Pund MA (2016).** Recent attack prevention techniques in web service applications. In *proceedings of the IEEE International Conference on Automatic Control and Dynamic Optimization Techniques*, (pp. 1174-1180).

**Dhara KM, Dharmala M,** & **Sharma CK (2015).** A survey paper on service-oriented architecture approach and modern web services. In *All Capstone Projects -157, http://opus.govst.edu/capstones/157*.

**Di Francesco P (2017).** Architecting microservices. In *proceedings of the IEEE International Conference on Software Architecture Workshop*, (pp. 224-229).

**Fett D, Küsters R,** & **Schmitz G (2016).** A comprehensive formal security analysis of OAuth 2.0. In *proceedings of the ACM Conference on Computer and Communications Security*, (pp. 1204-1215).

**Halili F,** & **Ramadani E (2018).** Web services: a comparison of soap and rest services. In *Modern Applied Science, vol. **12**(3)*, (pp. 175-183).

**Hassan MF (2016).** Construction of customizable SOA security framework using artificial neural networks. In *Jurnal Teknologi, vol. **78**(12)*, (pp. 69-75).

**Hu B, Bao M,** & **Dong (2014).** Improvement of user authentication protocol with anonymity for wireless communications. In *Kuwait Journal of Science, vol. **41**(1)*.

**Ibrahim BM,** & **Hassan MF (2015).** A new customizable security framework for preventing WSDL attacks. In *proceedings of the IEEE International Symposium on Mathematical Sciences and Computing Research*, (pp. 24-29).

**Katsikogiannis G, Mitropoulos S, & Douligeris C (2016).** An Identity and Access Management approach for SOA. In proceedings of the IEEE International Symposium on Signal Processing and Infor-mation Technology, (pp. 126-131).

**Kulesza R, Lima M, Araujo C, De Sousa MF, & Filho AM (2018).** Evolution of Software Architectures: from Web 1.0 to Web 3.0 systems. In proceedings of the 24th Brazilian Symposium on Multimedia and the Web, (pp. 11-13).

**Kumar D, Gupta V,** & **Kapur PK (2015).** Assessment of quality factors in enterprise application integration. In *proceedings of the 4th IEEE International Conference on Reliability, Infocom Technologies and Optimization*, (pp. 1-6).

**Lam Son (2020).** Diagramming Multi-Level service oriented Enterprise Architecture. In *Springer SN Computer Science*, DOI: https://doi.org/10.1007/s42979-019-0014-z.

**Li C, Xue Y, Wang J, Zhang W,** & **Li T (2018).** Edge-oriented computing paradigms: A survey on architecture design and system management. In *ACM Computing Surveys (CSUR), vol. **51**(2)*, (pp. 1-34).

**Liu D, Yan Z, Ding W,** & **Atiquzzaman M (2019).** A survey on secure data analytics in edge computing. In *IEEE Internet of Things Journal, vol. **6**(3)*, (pp. 4946-4967).

**Masood A, & Java J (2015).** Static analysis for web service security-Tools & techniques for a secure development life cycle. In *proceedings of the IEEE International Symposium on Technologies for Homeland Security*, (pp. 1-6).

**Mohamed M.I.B and Hassan M.F, Safdar S,** & **Saleem M.Q (2019).** Adaptive security architectural model for protecting identity federation in service oriented computing. In *Journal of King Saud University-Computer and Information Sciences*.

**Mourad A, Laverdiere MA, & Debbabi M (2008).** A high-level aspect-oriented-based framework for software security hardening. In Information Security Journal -A Global Perspective, (pp. 56-74).

**Newman R, Chang V, Walters RJ,** & **Wills GB (2016).** Web 2.0 - The past and the future. In *International Journal of Information Management, vol. **36**(4)*, (pp. 591-598).

**Pirnau C,** & **Botezatu MA (2017).** service oriented Architecture (SOA) and Web Services. In *Database Systems Journal, vol. **7**(4)*, (pp. 32-39).

**Rabelo RJ, Noran O,** & **Bernus P (2015).** Towards the next generation service-oriented enterprise architecture. In *proceedings of the 19th IEEE International Enterprise Distributed Object Computing Workshop*, (pp. 91-100).

**Rafique S, Humayun M, Hamid B, Abbas A, Akhtar M,** & **Iqbal K (2015).** Web application security vulnerabilities detection approaches: A systematic mapping study. In *proceedings of the 16th IEEE International Conference on Software Engineering, Artificial Intelligence, Networking and Paral-lel/ Distributed Computing*, (pp. 1-6).

**Rudman R,** & **Bruwer R (2016).** Defining Web 3.0: opportunities and challenges. In *The Electronic Library*.

**Sfar AR, Natalizio E, Challal Y,** & **Chtourou Z (2018).** A roadmap for security challenges in the Internet of Things. In *Digital Communications and Networks, vol. **4**(2)*, (pp. 118-137).

**Shashwat A, Kumar D,** & **Chanana L (2018).** Message level security enhancement for service oriented architecture. In *proceedings of the IEEE 4th International Conference on Computational Intelligence & Communication Technology*, (pp. 1-6).

**Sirohi P, Agarwal A, & Tyagi S (2016).** A comprehensive study on security attacks on SSL/TLS protocol. In proceedings of the 2nd IEEE International Conference on Next Generation Computing Technologies, (pp. 893-898).

**Sun L, Tan C, Robertson S, Liu K, Cook M, & Collins C (2016).** Open digital business ecosystems: A pathway for value Co-creation. In proceedings of the Springer International conference on informatics and semiotics in organisations, (pp. 85-94).

**Tout H, Mourad A, Talhi C, & Otrok H (2015).** AOMD approach for context-adaptable and conflictfree web services composition. In Journal of Computers and Electrical Engineering, (pp. 200-217).

**Wadhwa A (2016).** Comprehensive Analysis of Security Issues and Solutions While Migrating to Cloud Environment. In International Journal of New Innovations in Engineering and Technology, vol. **4(4),** (pp. 127-130).

**Yahyaoui H, El-Qurna J,** & **Almulla M (2020).** Specification and recognition of service trust behaviors. In *Kuwait Journal of Science, vol. **47**(1).*

**Yuan Y, Li B,** & **Kreger H (2016).** SOA reference architecture： standards and analysis. In *Springer International Conference on Smart Computing and Communication*, (pp. 469-476).

**Zheng L, Yuan H, Peng X, Zhu G, Guo Y,** & **Deng G (2019).** Research and implementation of Web application system vulnerability location technology. In *proceedings of the Springer International Conference on Cyber Security Intelligence and Analytics*, (pp. 937-944).

**Zimmermann A, Schmidt R, Sandkuhl K, Jugel D, Bogner J,** & **Mohring M (2018).** Evolution of enterprise architecture for digital transformation. In *proceedings of the 22nd IEEE International Enterprise Distributed Object Computing Workshop*, (pp. 87-96).

**Submitted:** 19/10/2020
**Revised:** 21/03/2021
**Accepted:** 21/03/2021
**DOI:** 10.48129/kjs.v49i1.10745