



ELSEVIER

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

ScienceDirect

journal homepage: [www.elsevier.com/locate/cose](http://www.elsevier.com/locate/cose)


---



---

**Computers  
&  
Security**


---



---



CrossMark

## A novel methodology towards a trusted environment in mashup web applications

Ahmed Patel <sup>a,b</sup>, Samaher Al-Janabi <sup>c,\*</sup>, Ibrahim AlShourbaji <sup>d</sup>,  
Jens Pedersen <sup>e</sup>

<sup>a</sup> School of Computer Science, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor Darul Ehsan, Malaysia

<sup>b</sup> School of Computing and Information Systems, Faculty of Science, Engineering and Computing, Kingston University, Kingston upon Thames KT1 2EE, United Kingdom

<sup>c</sup> Department of Information Networks, Faculty of Information Technology, University of Babylon, Babylon 00964, Iraq

<sup>d</sup> Computer Network Department, Computer Science and Information System College, Jazan University, Jazan 82822-6649, Saudi Arabia

<sup>e</sup> Department of Electronic Systems, Aalborg University, Aalborg, Denmark

---

### ARTICLE INFO

#### Article history:

Received 28 April 2014

Received in revised form

27 September 2014

Accepted 17 October 2014

Available online 28 October 2014

#### Keywords:

Application programming interfaces (API)

Mashup applications

Risk analysis

Security

Threats

Risk filtering data mining algorithm (RFDM)

Confusion matrix

---

### ABSTRACT

A mashup is a web-based application developed through aggregation of data from different public external or internal sources (including trusted and untrusted). Mashup introduces an open environment that is exposed to many security vulnerabilities, threats and risks. These weaknesses will bring security to the forefront when developing mashup applications and will require new ways of identifying and managing said risks. The primary goal of this paper is to present a client side mashup security framework to ensure that the sources for mashup applications are tested and secured against malicious intrusions. This framework is based on risk analysis and mashup source classification that will examine, analyze and evaluate the data transitions between the server-side and the client-side. Risk filtering using data mining suggests a new data mining technique also be utilized to enhance the quality of the risk analysis by removing most of the false risks. This approach is called the Risk Filtering Data Mining algorithm (RFDM). The RFDM framework deals with three types of clusters (trusted, untrusted and hesitation or unknown) to handle the hesitation clusters. Our proposal is to employ Atanassov's Intuitionistic Fuzzy Sets (A-IFs) as it improves the results of an URL. Finally, the results would be evaluated based on five experimental measures generated by a confusion matrix, namely: Accuracy (AC), recall or true positive rate (TP), precision (P), F-measure (considers both precision and recall) and  $F_\beta$ .

© 2014 Elsevier Ltd. All rights reserved.

---

\* Corresponding author.

E-mail addresses: [whinchat2010@gmail.com](mailto:whinchat2010@gmail.com) (A. Patel), [samaher@itnet.uobabylon.edu.iq](mailto:samaher@itnet.uobabylon.edu.iq) (S. Al-Janabi), [i\\_shurbaji@yahoo.com](mailto:i_shurbaji@yahoo.com) (I. AlShourbaji), [jens@es.aau.dk](mailto:jens@es.aau.dk) (J. Pedersen).

<http://dx.doi.org/10.1016/j.cose.2014.10.009>

0167-4048/© 2014 Elsevier Ltd. All rights reserved.

## 1. Introduction

Mashup is an exciting interactive web application that draws upon diverse content retrieved from external data sources to create entirely new and innovative meta-application services (Na et al., 2010). Although mashup implies “unstructured” by definition, it is organized, serving as a weaver that systematically aggregates and stitches together third-party data. For example, one could combine online weather data with a virtual web-based map by integrating a geospatially-indexed temperature feed with a Google Maps interface. This leads to creating a new service that is neither supplied by the provider of the geospatially-indexed temperature feed nor by Google Maps service.

The main characteristic of mashup is the aggregation of contents or program codes from various sources into one integrated webpage to be displayed on the client side (user browser); these sources could be external or internal, trusted or untrusted sources (Yu et al., 2008; Merrill, 2009; Bianchini et al., 2010). Mashup is important to optimize the use of existing data, and is also flexible to obtain the maximum benefits from its personal and professional use. In the past few years, more and more web applications' providers have published Application Programming Interfaces (APIs) that enable software developers to easily integrate data and functions instead of building applications by themselves (Na et al., 2010).

In most cases, mashups lack the ability to gather and integrate different services in a secure way as the services may have completely diverse security requirements in terms of authentication and authorization (Meng and Chen, 2009). As a consequence, the tools that are used to construct mashups have been focused only on integrating security-free data sources and omitting other sources that do not have proper certification for publishing APIs (Yu et al., 2008).

The degree to which the Internet users feel that a web site protects their privacy may also have an impact on their trust of the site. Although perceptions of security and privacy protection are likely to differ according to each user's, traditions, culture, social networking and other factors, web site design and content elements can exchange privacy credentials to ensure user information is protected. With regard to design, site complexity and layout for presence and number of hyperlinks, obtrusiveness of advertising, ease of navigation, and general professionalism have been shown to affect perceptions about the authority of the site (Na et al., 2010). Perceptions on the site's authority may impact the user's privacy perception of the site. Web site content also influences perceptions of a site's privacy protection through privacy credentials, certificates and seals provided by trusted third parties, such as TRUSTe (TRUSTe, 2014), BBB online (BBB, 2014), or VeriSign (Verisign, 2013).

Clearly this can be a problem in enterprise environments because users are very reluctant to reveal their authentication information to third parties. Moreover, many mashups are built by non-technical users with no absolute guarantees that they will not accidentally leak important private or confidential data (Zou and Pavlovski, 2007; Rosenberg et al., 2009). As more new APIs and data sources become available to make

mashup, system, vulnerabilities and security risks also increases. In order to continue such open integration it is essential and highly recommended that the integrated data are trusted and secured against malicious activities (Jackson and Wang, 2007).

The main goal of this work is to propose a mashup security framework that examines, analyzes and evaluates data transition between the server and the client-side (in client-server architecture), to ensure that the data sources are secure against security threats and malicious activities. This framework will implement a multilevel protection mechanism, which classifies the data source into trusted and untrusted source origins. The classification is based on an offline risk analysis process and an online monitoring of the data exchanges between the API providers and the client browser by measuring the residual risks and the sensitivity of the areas and assets that are required to be accessed. The proposed security framework will block/disallow the execution of mashup programs that display high risks to the client side during runtime operation.

## 2. Client-side mashup architecture

Mashups are web applications that integrate multiple data sources or API's into one integrated interface (Yu et al., 2008). Mashups typically allow the end user to discover and integrate a third party Ajax-powered mashup component onto a web-based application or website. Mashups can be considered to have an active role in the evolution of social software, Web 2.0 and Web 3.0 technologies (Bianchini et al., 2010). In a client-side mashup, the service or content integration takes place on the client side, which is typically a web browser. This is in contrast to a server-side mashup, where the service or content integration takes place in the server. A server-side mashup is also called a *proxy-style mashup* because a component in the server acts as a proxy to the service (Hilton, 2009). As illustrated in Fig. 1, a typical mashup framework is comprised primarily of:

- a) End user Browser: The end user or client browser, where the data and application program code will be mashed-up and displayed. It is the space where the integrated application programs and scripts will be executed and processed at runtime.
- b) Mashup Provider: The website where the mashups and required JavaScript libraries will be hosted. The products functionality can be accessed using the API services.
- c) Data: The core element of any mashup is the data being aggregated and presented to the user; the data can strictly come from web services where data is serialized to Extensible Markup Language (XML) or JavaScript Object Notation (Barth and Li, 2011).

The primary reason for using the proxy style is to contend with the basic security protection that the browser security sandbox provides. In a proxy-style mashup, a server-side proxy allows access to the service without the view of the browser security sandbox, thus permitting connection to a site other than the server of origin to access a service.

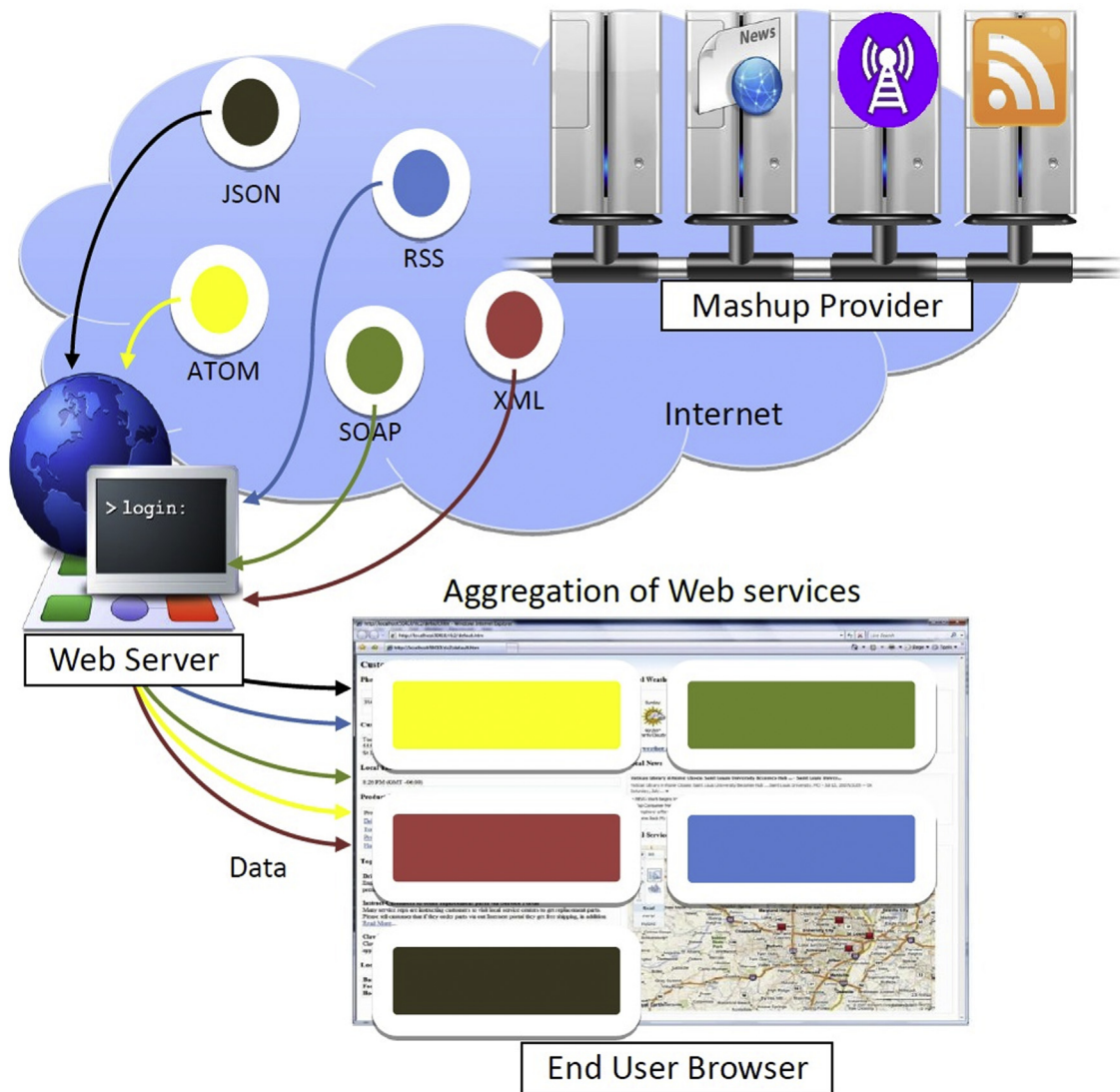


Fig. 1 – Typical Mashup framework aggregation of web services.

However, a client-side mashup also avoids the constraints of the browser security sandbox because the service call is made from a dynamically created `<script>` tag, which can communicate with any domain.

Mashup applications emerging from Web 2.0, Web 3.0 and Web 4.0 technology flawlessly combine contents from multiple public data sources. Mashups have their own natural attributes and common characteristics, which can be categorized as essential features (Alur, 2007; Daniel et al., 2011), including:

- Combines content from two or more resources into a single integrated application.
- Web technology based on World Wide Web Consortium (W3C)'s Web 2.0, Web 3.0 and Web 4.0 as a platform leads to information sharing, collaboration and social networking.
- Situational demands based on end-users' immediate short term needs.
- End-users, as co-creators, decide which features they want and become owners of the data that they generate.
- Rich, interactive, user-friendly, and easy-to-use interfaces to access easy to start/run meta-applications.
- Lightweight programming model in which languages and simple data format make development easy and cost effective.
- Observe the result of the meta-application instantly before showing or distributing it online.

Furthermore, mashups have extensive applications across most fields from businesses to healthcare, and departments from marketing to research and development. We have identified several universally applicable mashup applications. For example, Fig. 2 shows a customer service mashup that could be used by a representative when receiving a call to handle a call out. By combining the data from public services such as weather and news with internal sources of data such

## Customer Service Mashup

**Phone Number**

314-555-1212

**Customer Information**

Tom Drake  
555 Mockingbird Avenue  
St Louis, MO 63201

**Local Time**

8:26 PM (GMT -0600)

**Product Information**

| Product                            | Date Purchased |
|------------------------------------|----------------|
| <a href="#">Drillmaster 5000</a>   | 2007-Jan-05    |
| <a href="#">Torque Wrench 6000</a> | 2006-Dec-02    |
| <a href="#">Pressbench 2000</a>    | 2004-Mar-17    |
| <a href="#">Hammer 4000</a>        | 1992-May-05    |

**Top Service Issues**

**Drillmaster Batteries Dying Early**  
Engineering has confirmed that the batteries on the new Drillmaster line have started to build memory prematurely. [Read More...](#)

**Instruct Customers to order replacement parts via Service Portal**  
Many service reps are instructing customers to visit local service centers to get replacement parts. Please tell customer that if they order parts via our Internet portal they get free shipping. in addition [Read More...](#)

**Claw Hammers experience issues**  
Claw Hammers (all makes and models) have been reported to have shearing issues when pressure is applied to them. [Read More...](#)

**Local Talking Points**

Baseball Team: St. Louis Cardinals  
Football Team: St. Louis Rams  
Hockey Team: St. Louis Blues

**Local Weather**

Sunday  
  
90°/69°  
Partly Cloudy

Monday  
  
93°/73°  
Mostly Sunny

Tuesday  
  
92°/74°  
Scattered T-Storms

Wednesday  
  
92°/74°  
Isolated T-Storms

from [weather.com](#)

**Local News**

Vatican Library in Rome Closes: Saint Louis University Becomes Hub ... - Saint Louis Univer...  
Vatican Library in Rome Closes: Saint Louis University Becomes Hub ... Saint Louis University, MO - Jul 12, 2007.LOUIS - On Saturday, July... »

- NEW: Work begins to eliminate red lights on Hwy. 40 - Suburban Journals
- Top Consumer Magazine Says Saint Louis University is Among the ... - Saint Louis University
- Symphony offers deals for 40th anniversary - St. Louis Business Journal
- Gimme Back My Mask! - Riverfront Times

**Local Service Centers**




Fig. 2 – An example of a customer service Mashup.

as a database of service locations and service status, this mashup application provides the most optimum representative to reach the customer.

### 3. Risks and security challenges of mashups

Security, privacy and confidentiality of electronic data are major concerns in Information and Communication Technology (ICT) and are the leading matter in informatics (Meng and Chen., 2009; Tanaka et al., 2011). The main intention of security is the protection of personal data and information against danger, damage, theft, corruption, loss or natural disaster as well as criminal activities (Liu et al., 2009), while allowing data to remain accessible and productive to its intended users. Confidentiality is the degree of protection against the disclosure of personal and sensitive data to untrusted and/or unauthorized parties (Patel et al., 2013). Its main concern is regarding the transactions of personal data via the Internet for payment or profiling. The term privacy is referring to the user's desire or intent of protecting his/her

personal and sensitive data from being accessed by others without permission (Meng and Chen., 2009; Ahmad, 2008).

Mashups introduce an ambiguous data exchange in open environment, which exposes many new security risks and vulnerabilities. This open environment with the absence of standardization and flexibility leads to diverse security threats that can multiply very quickly. Therefore, it is important to highlight the security measures and address their requirements and implications when developing mashup applications.

Secured mashup applications require new methods of identifying and managing threats and security risks exposed to the systems where the mashup operates. Mashups aggregate data and JavaScript programs within the client browser from many different sources, but most of them are not been recognized; users are forced to have a complete trust of these sources (ASP Alliance, 2007; Tanaka et al., 2011).

Mashup applications, by their nature, involve interaction between various page data. Often the data are loaded from different sources. Access controls in today's browsers are governed by what is known as the Same Origin Policy (SOP)

(Jackson and Wang, 2007). The origins of the data are identified by the Internet domain, protocol, and a port number. This provides total isolation by preventing application programs and scripts loaded from one origin to access document properties from another.

Documents from the same origin may freely access each other's content, while such access is disallowed for documents of different origins. Unfortunately, the SOP mechanism turns out to be problematic for mashup security. The origin tracking in SOP is only partial and allows content from different sources to coexist under the same place of origination. When an HTML script-tag is used to load a JavaScript program from a different origin, the loaded script is integrated and combined into the requested document, and thereby can freely interact with it and be executed in the client-side. For the same reasons, interaction between different data sources loaded in this way is unsecured and unrestricted.

The problem of script-tag inclusion for mashup applications is that the user must trust the third parties (mashup providers) to protect their information. Effectively, the security of the user no longer depends only upon the client-side, but also on the security of the third parties whose scripts are included. For example, the HTML script-tag is used to load content from some other origin and to integrate it within the requested document. Once integrated, such content is considered to be of the same origin and to pass the origin tracking in SOP as it is an integrated and combined document. This means that the content is accessible to scripts in other documents from the same origin. As an example, Cross-Site Scripting (XSS) is a common attack in which an attacker injects a malicious piece of program code into an otherwise benign site. As illustrated in Fig. 3, the HTML script-tag is used to load content from an *evil.com* domain (malicious) which is completely different from a *trusted.com* domain (origin) and integrates it within the requested document. An XSS attack exploits vulnerable web applications that display input parameters back to the browser without checking for the presence of active content in them (Meng and Chen., 2009).

Unfortunately, malware program codes are not easily detectable inside the mashup and may infect Internet users' computers and expose confidential and private information when they simply browse infected mashup applications. Therefore, there is an urgency to provide a secure mashup environment. This work proposes a solution for the implications of the security issues in the mashup platform.

#### 4. Mashup security requirements

Security controls in today's browsers are governed by what is known as the Same Origin Policy. Most of the mashup applications bypass this policy by evading it. As a result, the current security model does not satisfy the user's needs and the security protection requirements that make the information systems running the mashup application vulnerable. This means that a new security framework must be built from scratch in order to have a unique and effective form of protection.

The first step in setting up a new security framework is to identify and state the parties involved in a typical mashup application (Patel et al., 2010b). We have already described the architecture of a typical mashup which is composed of three components: The user interface (client), the mashup provider and the data. The data is transmitted from the client to the server and vice versa. The central party is, of course, the user of the web browser in which the data and application program will be aggregated and executed. Therefore, most security requirements are user-oriented requirements on the client side. As shown in Table 1, the mashup safety requirements fall into common categories.

In general, safety measures in the broadest sense are difficult to provide and are not always important for some applications in enterprise information systems. Many, if not most applications, do not need these problems solved. For example, many simple applications, like surfing the Internet by normal users for leisure holiday packages which do not impeach on the user's rights or the system's behavior, do not need any major safety measures other than simple malware and spam control. Applications such as mashup need to protect the private data of the user that should not be observed by other parties. If a client-side dataflow monitor scheme with trusted and untrusted classification lists is implemented, the user should be able to expect that private and confidential information cannot be read by less trustworthy mashup providers.

Patel et al., 2013, defined eight critical security concerns and requirements of secure and trustworthy mobile agent-based e-marketplaces run over the Internet. These automated agents play the role of real users, service providers, brokers, auctioneers, fund transfer managers, financial institutions and trusted third parties. Together they form a community that must deliver on their obligations and promises.

```
http://trusted.com/search?keyword=<script>
document.images[0].src="http://evil.com/steal?cookie="+
document.cookie; </script>
```

Fig. 3 – Cross-site scripting example.

**Table 1 – Safety measure requirements for mashup applications.**

| SR# | Requirement                     | Description  | Specify  |
|-----|---------------------------------|--|--|
| SR1 | Data Security (Integrity)       | The data exchanged between the user and the provider should not be intercepted and corrupted by untrusted third parties before it is received by the service provider. | Imposed in the client (user) side to ensure data integrity.  |
| SR2 | Data Security (Availability)    | User's contents should not be modified, disclosed and/or disrupted by a third party or service provider.   | The service provider should ensure data availability<br>Safe data execution within the service provider. |
| SR3 | Data Security (Confidentiality) | The user should expect a safe and secure execution without being concerned about the disclosure of his/her private and sensitive data.                                 | The service provider should ensure user privacy.   |
| SR4 | Privacy                         | User's input is expected to be available only to a particular website and not to be intercepted by other parties.  | Imposed in the client (user) side to ensure data, transaction and communications are genuine.            |
| SR5 | Authenticity                    | The user may want to have a guarantee about the identity of the parties with which he/she is communicating.  | Imposed in the service provider side.  |
| SR6 | Auditing and Digital Forensics  | The systematic measurable technical assessment of policy employment and to trace back investigations of offensive incident.  |  |
| SR7 | ID Management                   | The service provider should keep track of each and every user who uses its web based applications aligned with the current cyber legislations.                         | Implemented and imposed in the service provider side.  |

Another important security and privacy measure is to create awareness in the general public about mashup's security and privacy. This is because some end-users do not have an understanding of this type of technology and its use; therefore they are not in a position to make balanced judgments concerning the extent to which it may have a negative impact on their own perceived standards of privacy.

There are also many other security related ethical and technical issues that also need to be resolved. For instance, any breaches and intrusions in the running, administration and management of cloud computing services in any enterprise domain have to be detected and, subsequently, prevented through novel ways using machine learning and predictive analysis techniques (Patel et al., 2013).

Finally, it will also require actual and non-actual regulatory and standards bodies, governments, industries and service providers to address the safety measures' issues to synthesize legislations, directives and guidelines for mashup applications as part of a comprehensive deployment strategy.

## 5. The mashup security solution

Mashups are an increasingly popular approach to developing new kinds of situational web applications by combining content from various sources. These types of developments expose extensive security threats to the client-side where the mashup application is to be executed. Current security measures and protection mechanisms are unsatisfactory; therefore, there is an urgent need for developing a security framework. Security needs to be addressed very well to protect the client from malicious program codes and criminal activities.

In this work, we propose a novel security framework to ensure that applications are trusted and secured against malicious and harmful application programs. This can be achieved by applying a new data mining technique called the Risk Filtering Data Mining algorithm (RFDM) in order to enhance the quality of the risk analysis and to remove a major part of the false risks at runtime in the client-side mashup application program. This method allows quarantining/hibernating by prevention the exposure of valuable and sensitive information whether personal or otherwise into non-executable restrictive defined areas within the client-side program.

The RFDM becomes as part of the URL Listener. In this scheme, we treat the hesitation class result by the classifier (validation vector) to co-work with the A-IFs. After handling the hesitation class (i.e. unknown class), all the URLs are labeled as trusted or untrusted. Finally, the results are evaluated for this novel methodology by computing the confusion matrix and using four measures: Accuracy (AC), recall or true positive rate (TP), precision (P) and F-measure, which includes both precision and recall.

In general, this work deals with two problems related to imbalanced and overlapping classes that are a real classifier's challenge for constructing standard classifiers:

- The problem of imbalanced classes is extremely common in practice and can be observed in several fields such as anomaly detection and medical diagnosis; it occurs when

the total number of the training set of a classifier has contained some legal examples related to (i.e., majority class or “trusted”) and far less than the other examples related to (minority class or “untrusted”). Usually, up sampling and down sampling are used in order to deal with the imbalance problems.

- While the problem of overlapping classes occurs when the set of training examples presented to the classifier have many different samples they may seem to be valid examples and have very similar features in both classes (trusted or untrusted).

Our approach seeks to increase the security, thus the data protection is composed into *two levels of protection*; the top level handles JavaScript source validation and authentication, while the bottom level handles the monitoring of data exchange between the source and the client and vice-versa. These monitor the access rights of each request made by the server or remote client to a system asset. Most importantly, the conceptual framework of our proposed mashup security tool

is designed based on the requirements analysis and current state of the art. The proposed conceptual framework of the system includes the general modules and processes as shown in Fig. 4.

The general functional operations of the proposed system are:

- URL Listener & RFDM:** The URL listener filters the incoming HTML document before the integration with the inbound JavaScript program code; it performs extracting and collecting of the URLs of the HTML-script tag into RFDM. The first step in the RFDM is to assess the risk analysis of URLs by removing most of their false risks and abnormal patterns of behavior. Generally, three criteria are considered to determine if an activity is classified as an attack attempt:
  - a) Matching source IP address.
  - b) Matching target IP address.
  - c) Alarm time stamp in the time window in which the attack occurred.

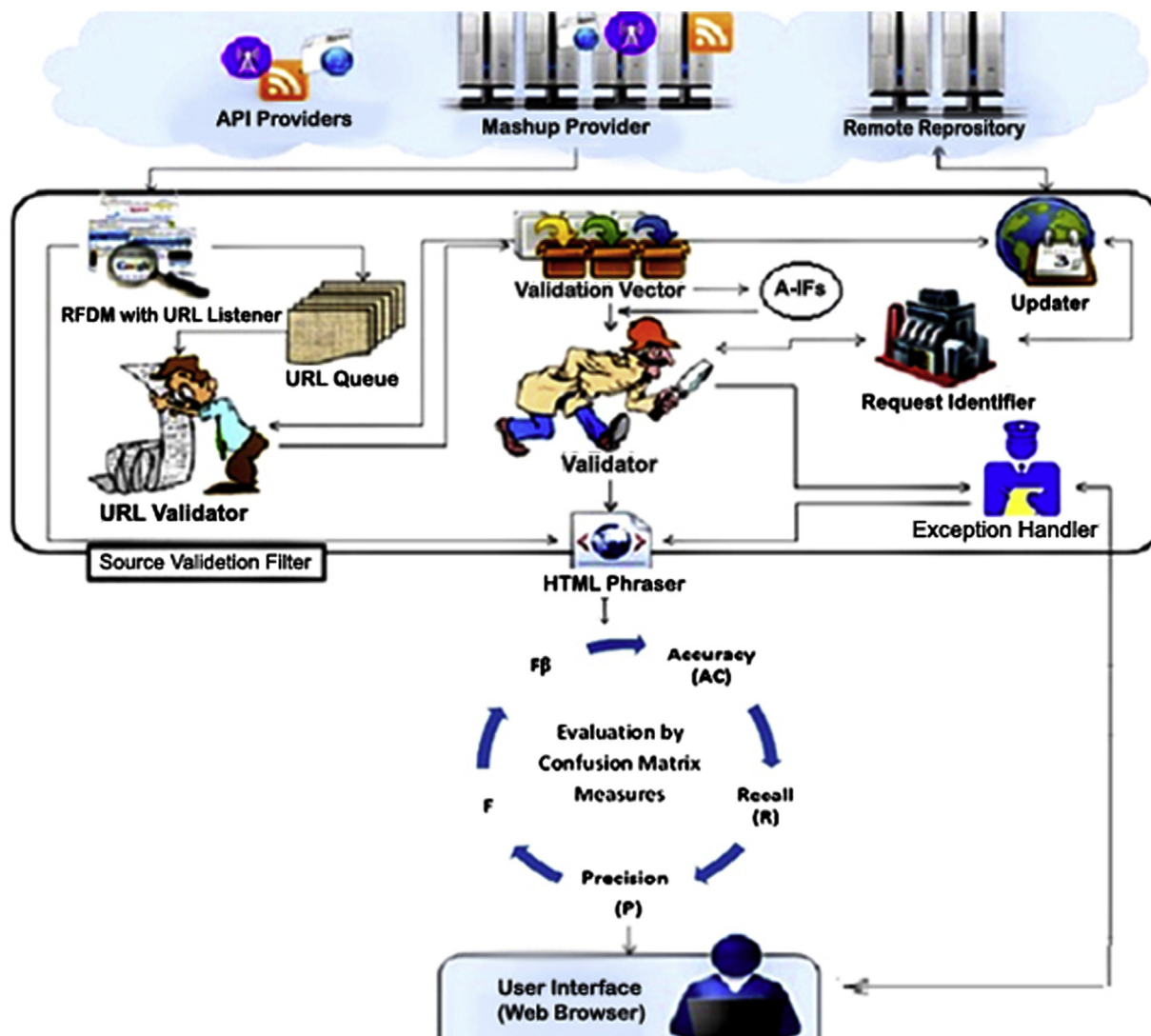


Fig. 4 – Conceptual framework of the system.

The main idea of the RFDm is to find a set of URLs in which their distances among risks are less than or equal to the neighbor's Original Threshold (i.e. NOT is a control parameter that controls the distances between URL's risk) then, to retrieve a generalized risk (GR) for each URL.

### 5.1. How to determine the “NOT” parameter?

Cluster validity process is used to evaluate the results of a clustering algorithm. Moreover, the indices related to these approaches aim at measuring the degree to which a dataset confirms an a-priori specified scheme. On the other hand, the relative criteria aim at finding the best clustering scheme that a clustering algorithm can define considering certain assumptions and parameters (Halkidi et al., 2001). The more suitable criteria for the NOT value estimation is the relative criteria. There are two criteria proposed for clustering evaluation and selection of an optimal clustering scheme (Berry et al., 1996): Compactness and separation. Compactness means that the members of each cluster should be as close to each other as possible while separation means that the clusters themselves should be widely spaced. There are many validity indices for the relative criteria, among them are: The Dunn, the Jaccard, the Davies-Bouldin (AL-Janabi, 2013) and the SD (Halkidi et al., 2001) indices. We have selected the SD index because its time complexity is  $O(n)$  (Halkidi et al., 2000). In the sequel of this subsection, we will state this index. The SD validity index is defined based on the concepts of the average scattering for clusters and total separation between clusters. The average scattering for clusters (Halkidi et al., 2000) is defined by the following equation:

$$\text{Scat}(c) = \frac{1}{c} \sum_{i=1}^c \frac{\|\sigma(v_i)\|}{\|\sigma(X)\|} \quad (1)$$

Where,  $c$  represents the number of clusters,  $v_i$  refers to the center of cluster  $i$ ,  $\sigma(v_i)$  refers to the variance of cluster  $i$ , and  $\sigma(X)$  is the variance of a dataset. The definition of total separation between clusters is given by the following equation:

$$\text{Dis}(c) = \frac{D_{\max}}{D_{\min}} \sum_{k=1}^c \left( \sum_{z=1}^c \|v_k - v_z\| \right)^{-1} \quad (2)$$

Where,  $D_{\max}$  represents the maximum distance between cluster centers and  $D_{\min}$  represents the minimum distance between cluster centers. A validity index SD can be computed as follows:

$$\text{SD}(c) = \beta \text{Scat}(c) + \text{Dis}(c) \quad (3)$$

Where,  $\beta$  is a weighting factor equal to  $\text{Dis}(c_{\max})$  where  $c_{\max}$  is the maximum number of clusters. The NOT parameter value that minimizes the SD index can be considered as the best value.

The RFDm executes according to the following steps:

- i. The set of risks contain several clusters ( $C_1, C_2, \dots, C_m$ ) that represent URLs which are required by the user, noise, and attacks.
- ii. Every cluster  $C_i$ ,  $1 \leq i \leq m$ , contains a set of URLs whose distances between them and the center of cluster  $C_i \leq \text{NOT}$ .

- iii. Extract the generalized risk for every cluster  $C_i$ ,  $1 \leq i \leq m$ , by finding, separately, the nearest common ancestor for each URL and then merge the identical generalized risks.
- iv. Forward these generalized risks to the security analyst to extract the root causes and to write filters for them once the set of a GR starts the job of URL Queue.
  - B. **URL Queue:** The queue is a type of memory structure (container), specifically designed to operate in a first-in first-out (FIFO) context, where the elements are inserted into one end of the container and extracted from the other. The URL Queue holds the extracted URLs from the HTML-script tag processed by the URL Listener component.
  - C. **URL Validator:** This component is responsible for authenticating the origin of the third-party services by validating the URL links of the external sources within the HTML script extracted by the URL Listener before it is integrated and combined with the requested document. The URL validator also has an ongoing access to the local repository.
  - D. **Validation Vector:** This component classifies URLs after they have been processed by the validator. It consists of three labeled queues, each of which accepts a specific type of validated URLs: One for URLs labeled as **trusted**, one for those URLs labeled as **untrusted** and the last for the **unknown (hesitation) URLs**, these latter relate to or are unprecedented sources.

*The main questions are, what are Atanassov's intuitionistic fuzzy sets (A-IFs) and why are they used in this stage of the suggested methodology?* The answers to these questions are as follows:

- i. The Atanassov's intuitionistic fuzzy sets are generated according to an automatic and mathematically justified procedure from the relative frequency distributions representing the URL.
- ii. We use the information about the so-called hesitation URL (which, besides membership and non-membership values characterizes Atanassov's intuitionistic fuzzy sets), making it possible to improve the results of URL classification.
- iii. A-IFs are used for representation of classes.
- iv. We obtain a better recognition of the smaller classes by exploiting the structure of A-IFs.

*How to determine whether that hesitation URL is classified as a trusted or an untrusted URL based on the A-IF's principle?* The answers to this question are as follows:

- i. Compute the relative frequencies connected to the membership values for the trusted and untrusted URLs.
- ii. Compute the value of the membership function for a fuzzy set  $\text{Pos}^+$ .

$$\text{POS}^+(X) = U(X) + \tau(X) \quad (4)$$

where,  $U(x)$  is the value of the membership function A-IFs,  $\tau(X)$  are the values of the hesitation URLs.



- iii. Compute the value of the non-membership function for a fuzzy set Pos:

$$\text{POS}^-(X) = V(X) + \tau(X) \quad (5)$$

where  $V(X)$  is the value of the non-membership function, A-IFs,  $\tau(X)$  are the values of hesitation URLs.

- iv. From equations (4) and (5) and taking into account that  $U(X) + V(X) + \tau(X) = 1$ , we obtain the values  $\tau(X)$

$$\begin{aligned} \text{POS}^+(X) + \text{POS}^-(X) &= U(X) + \tau(X) + \text{POS}^-(X) = V(X) + \tau(X) \\ &= 1 + \tau(X) \end{aligned}$$

$$\tau(X) = \text{POS}^+(X) + \text{POS}^-(X) - 1 \quad (6)$$

- v. Find the values of the membership function where A-IFs are called  $U(X)$  based on equations (4) and (6).

$$U(X) = \text{POS}^+(X) - \tau(X) \quad (7)$$

- vi. Find the values of the non-membership function where A-IFs are called  $V(X)$  based on equations (5) and (6) that, we obtain.

$$V(X) = \text{POS}^-(X) - \tau(X) \quad (8)$$

E. **Validator:** It is another essential component in the system that is responsible for monitoring the data flow and information exchange between the browser and the remote server. The main function of this component is to monitor the access to the predefined restricted data areas in the clients' side and to prevent the exposure to sensitive and restricted system assets. These restricted data areas and assets are defined based on their sensitivity, profitability, data confidentiality and data privacy it holds during the process of risk assessment.

F. **Updater:** This component is responsible for synchronizing the remote repository with local repositories (black and white API provider lists and access rights lists).

G. **Request Identifier:** The main functions of this component include identifying what information is requested by the remote server and the information asset that is required to be accessed as well. This component works concurrently with the *URL validator* in order to provide all the required information to accomplish the correct decision.

H. **Exception Handler:** Its main purpose is to grant the end user the final decision whether to process or block the suspicious and malicious activities. It handles the user's special permission to execute and aggregate the requested data.

I. **HTML Phraser:** Represents the interaction point with the end user browser. It is also responsible for aggregating and integrating the mashup application and its JavaScript libraries.

J. **A confusion matrix:** It is a specific table layout allows visualization of the performance of a novel methodology. Table 2 shows the observations that represent the source of real URLs (i.e., trusted or untrusted URL) against predicted analysis that result from our method after handling the hesitation URL. Each row of the matrix represents the URL that has been automatically predicted by a novel methodology, while each column represents the actual URL that has been seen in the evaluation period.

There are several standard terms that have been defined for the two classes of the confusion matrix as follows:

- i. The accuracy AC is the proportion of the total number of predictions that are correct. It is determined using the following equation:

$$AC = \frac{a + d}{a + b + c + d} \quad (9)$$

- ii. The recall or true positive rate TP is the proportion of positive cases that are correctly identified and calculated using the following equation:

$$TP = \frac{a}{a + c} \quad (10)$$

- iii. Finally, precision P is the proportion of the predicted positive cases that are correct and calculated using the following equation:

$$P = \frac{a}{a + b} \quad (11)$$

- iv. The F measure considers both precision and recall providing a single measurement for a system:

$$F = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (12)$$

The F measure was derived so that  $F_\beta$  measures the effectiveness of analysis (Narayana et al., 2012). The parameter  $\beta$  generally takes the values 0.5, 1 and 2, with the higher values of  $\beta$  placing a greater emphasis on securing the analysis right in the sense of achieving a higher positive analysis value:

**Table 2 – The confusion matrix compares the forecasting URLs with the observations.**

| Predicted                    | Observed              |                         |
|------------------------------|-----------------------|-------------------------|
|                              | Positive: Trusted URL | Negative: Untrusted URL |
| Positive: Trusted analysis   | a                     | b                       |
| Negative: Untrusted analysis | c                     | d                       |

$$F_{\beta} = \frac{(1 + \beta^2) \cdot (\text{Precision} \cdot \text{Recall})}{(\beta^2 \cdot \text{Precision} + \text{Recall})} \tag{13}$$

To explicate on the functionality of the system, the conceptual framework is complemented with the functional model of the proposed security tool. Fig. 5 illustrates the functional model of a client-side mashup security tool (i.e., functional model of the Examining, Filtering and Analysis (EFA) of URL of web location system) which is composed into five main layers:

A. **First Layer:** When an end user browser sends a request to the remote server requesting a web page with mashup application, an HTML the user (client) passes this request to the specified remote mashup provider. The server responds to the requested web site and loads the page onto the client's interface. This page normally

includes links to a JavaScript library from the mashup provider.

B. **Second Layer:** The URL listener in the source monitor module will analyze that page and extract the JavaScript library links. The resulting links will then be forwarded to RFDM in order to cluster the generalized risks before sending them for filtering. The URL stores the resulting links in the queue component within the source monitor module to be processed by the URL validator. The URL validator is complemented with the XML file, which is an ongoing update from the remote repository with trusted and untrusted API Providers' domain names. The main function of the URL validator is to validate and authenticate the origin of the third-party services by verifying the URL links of external sources within the HTML script that is extracted by the URL

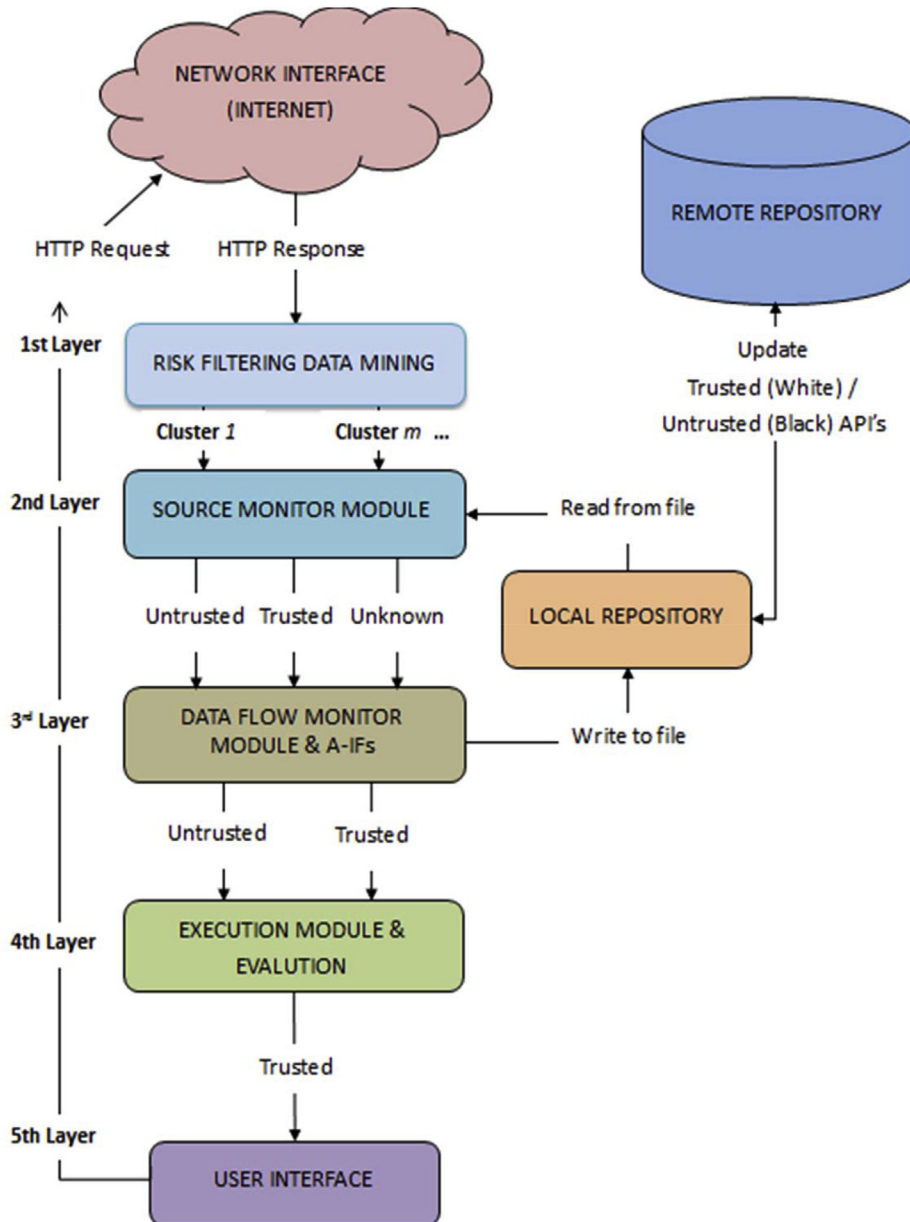


Fig. 5 – Functional model of the examining, filtering and analysis (EFA) of URL of Web location.

listener before it is integrated and combined with the requested document.

- C. **Third Layer:** Upon validation, the extracted URLs will then be classified and labeled as *trusted*, *untrusted* or *unknown*. The unknown or hesitation URL is sent to the second classifier A-IF to determine whether the trusted or untrusted URL based on its membership function. The URL is classified by the first and second classifiers, and then it is directed and stored in the validation vector component in the data flow monitor module in order to be processed by the validator in the next step. The validator component in data flow monitor module is responsible for validating the requested information and system assets that attempt to access to ensure that

the trusted API provider will be unable to access any sensitive or private resources.

- D. **Fourth Layer:** The validator works concurrently with the Request Identifier Module. The Validator will now process the requests queued in the validation vector with the aid of the Request Identifier component and provide the validation results to the HTML phrase. All the results of the previous layers enter into the evaluator, which analysis and reports about the robustness and effectiveness of the suggested novel methodology based on the five measures consisting of *Accuracy*, *Recall*, *Precision*,  $F$ ,  $F_{\beta}$ .
- E. **Fifth Layer:** The goal of the validator is to identify and detect activities that are unusual and try to access

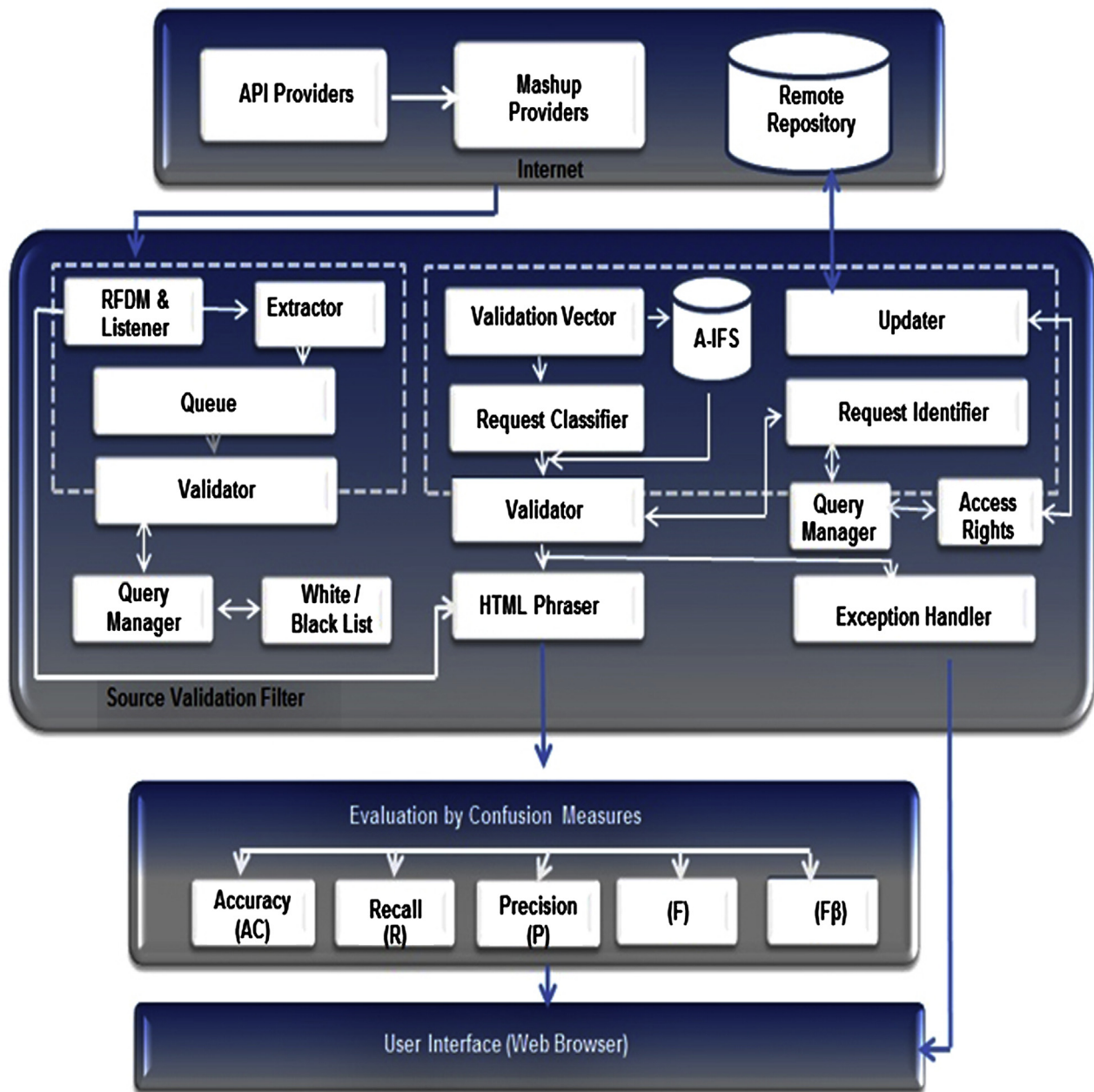


Fig. 6 – System architecture of our client side mashup security tool.

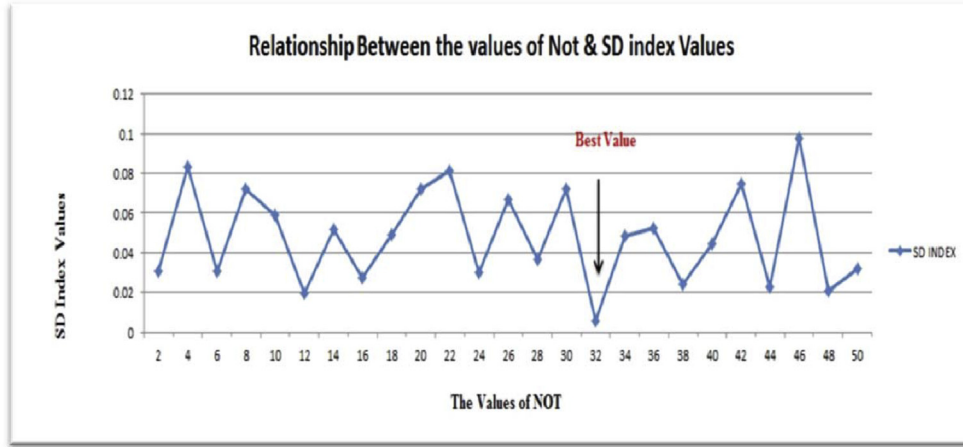


Fig. 7 – The relationship between NOT & SD index values.

sensitive data by temporarily blocking the mashup application and displaying a warning message on the user interface/browser in order to prevent them from accessing or damaging data and to help mitigate risks.

The updater component is responsible for synchronizing the local and remote repositories to update the black and white API provider lists and access rights lists as well. In conjunction with the exception handler component, the user will be able to better determine and also has the decisive rule of whether to execute or permanently block the suspicious activities and then update the local black-lists with the Java library domain name.

The system architecture of the client-side Mashup security tool is typically shown in Fig. 6. The system architectural model also shows the principal sub-systems introduced earlier. Fig. 7 illustrates the pseudo code of the proposed system and the data flow between each of its modules.

The research follows a rational, logical approach and processes in a top-down manner as shown in the flow chart below in this paper. Every key step has been logically devised to ensure that the research work is well-defined and can proceed in a systematic fashion, thus minimizing any unforeseen dependencies or failures. The request will be validated to check what information and/or system resource tries to access even if the URL of the API provider is in the trusted

list, in order to ensure that the sensitive and private information is kept safe and will not be exposed by a third party as well, and also to align our solution with the proposed system's requirements. Further, a sample of a trusted API provider is shown in Table 3 as listed in [www.programmableweb.com](http://www.programmableweb.com), where the evaluation of each source is based on the authentication services and SSL certification provided by VeriSign (Programmable web, 2014). VeriSign authentication services provide solutions which allows companies and consumers to engage in online communications and commerce with a very high level of confidence (VeriSign, 2013).

VeriSign is one of the most widely used authentication services and can prove to be an invaluable standard; its goal is to offer a wide range of online security services and trust. In the beginning, the black-list will be empty and then will be automatically updated and modified by the validator component based on the requests made by the specified domain, while the white-list will be initialized with the verified lists of safe, secure and authenticated API providers.

## 6. Discussion

A mashup is a web application that integrates content from various different providers to create new services, originally not offered by the content providers. Mashup applications are

Table 3 – Sample of trusted API providers obtained from the Programmableweb site (Programmable web, 2014).

| API provider            | Description                | URL  |
|-------------------------|----------------------------|--|
| Google Maps             | Mapping services           | <a href="http://code.google.com">code.google.com</a>                           |
| Flickr                  | Photo sharing service      | <a href="http://code.flickr.com">code.flickr.com</a>                           |
| YouTube                 | Video sharing              | <a href="http://code.google.com">code.google.com</a>                           |
| Twitter                 | Microblogging service      | <a href="http://dev.twitter.com">dev.twitter.com</a>                           |
| Amazon eCommerce        | Online retailer            | <a href="http://affiliate-program.amazon.com">affiliate-program.amazon.com</a> |
| Facebook                | Social networking service  | <a href="http://developers.facebook.com">developers.facebook.com</a>           |
| eBay                    | Online auction marketplace | <a href="http://developer.ebay.com">developer.ebay.com</a>                     |
| Last.fm                 | Online radio service       | <a href="http://last.fm">last.fm</a>   |
| Microsoft Virtual Earth | Mapping services           | <a href="http://microsoft.com">microsoft.com</a>                               |
| Google Search           | Search services            | <a href="http://code.google.com">code.google.com</a>                           |
| del.icio.us             | Social bookmarking         | <a href="http://delicious.com">delicious.com</a>                               |

up and coming trends that allow users and web developers to integrate contents and program codes from other content providers. This open model of integration leads to security vulnerabilities that can multiply very quickly.

With each new data source added to a mashup, security risks increase. Malware program codes are not easily detectable inside the mashup and may infect Internet users' computers and expose confidential and private information when they simply browse infected mashup applications. This work is intended to propose a solution for security issues and against threats in the Mashup platform. With Mashups' growing popularity, the problem of securing information flow between Mashup components becomes critically pertinent.

This section, describes the experiments conducted to evaluate RFDm, and to show how RFDm enhances the quality of analyzing the risk by reducing false risks. This is validated by the following four statements:

- (i) Few root causes are responsible for generating a huge number of risks.
- (ii) The clustering by our algorithm helps the security analyst to discover the root causes of false risks.
- (iii) If the real root causes are discovered and good filtering rules are written for them, then the filtering is safe.
- (iv) Converting these clusters to filtering rules helps in reducing future risks' load.

We can determine the best number of NOT based on computing the minimum values of SD index as given in Table 4 which shows both the NOT and SD INDEX values.

In the above table, we can observe that the best value of NOT is 32, it is also determined by bold font because that value is correlated with the minimum number of SD index. In general, in our experiments 25 different values were used to represent NOT values and to compute the 25 different values of SD index in order to better determine the best value. Fig. 7 typically shows the relationship between NOT and SD index values.

The following are the main reasons for using A-IFs as classifiers of the hesitation URLs:

| Table 4 – Find best NOT based on the values of SD index. |             |            |                    |
|--|-------------|------------|--------------------|
| NOT Values   | SD INDEX    | NOT Values | SD INDEX           |
| 2  | 0.030367799 | 28         | 0.036393156        |
| 4  | 0.083239352 | 30         | 0.07230287         |
| 6  | 0.030423686 | <b>32</b>  | <b>0.005409479</b> |
| 8  | 0.072024792 | 34         | 0.048247743        |
| 10   | 0.059104849 | 36         | 0.052347072        |
| 12   | 0.019752328 | 38         | 0.023885391        |
| 14   | 0.051471607 | 40         | 0.044170298        |
| 16   | 0.027548625 | 42         | 0.075019999        |
| 18   | 0.048904539 | 44         | 0.022850628        |
| 20   | 0.072241044 | 46         | 0.097847532        |
| 22   | 0.081034157 | 48         | 0.020795018        |
| 24   | 0.030188383 | 50         | 0.031841535        |
| 26   | 0.066826998 |            |                    |

Bold value represent the best value of SD index when note value equal to 32.

**Table 5 – Three sets of evaluation of a novel methodology based on the five measures.**

| Input                               | Evaluation measures | Intuitionistic Fuzzy Classifier |               |
|-------------------------------------|---------------------|---------------------------------|---------------|
|                                     |                     | Training                        | Testing       |
| # Training & Testing Dataset = 720  | AC                  | <b>0.94</b>                     | 0.89          |
| $\beta = 0.5$                       | Recall (TP)         | 0.81                            | <b>0.86</b>   |
| $\mathfrak{B} = 0.7$                | Precision P         | 0.17                            | <b>0.65</b>   |
|                                     | F                   | 0.281                           | <b>0.7404</b> |
|                                     | $F_{\beta}$         | <b>1.040</b>                    | 0.683         |
| # Training & Testing Dataset = 1275 | AC                  | <b>0.95</b>                     | 0.91          |
| $\beta = 0.5$                       | Recall (TP)         | 0.94                            | <b>0.95</b>   |
| $\mathfrak{B} = 0.6$                | Precision P         | 0.78                            | <b>0.88</b>   |
|                                     | F                   | 0.8526                          | <b>0.9137</b> |
|                                     | $F_{\beta}$         | 0.807                           | <b>0.893</b>  |
| # Training & Testing Dataset = 2000 | AC                  | 0.96                            | <b>0.98</b>   |
| $\beta = 0.5$                       | Recall (TP)         | <b>0.99</b>                     | 0.92          |
| $\mathfrak{B} = 0.9$                | Precision P         | 0.78                            | <b>0.89</b>   |
|                                     | F                   | 0.8725                          | <b>0.9048</b> |
|                                     | $F_{\beta}$         | 0.815                           | <b>0.896</b>  |

Bold values indicates the best values of (accuracy, Recall, Precision, F and  $F_{\beta}$ ) of Intuitionistic Fuzzy Classifier in training or testing phase.

- (i) [Programmable web, 2014](#) provides the user with the white-list or the trusted and secret URL. Therefore, it can be considered as the standard norm of online security and trust. In this way, it becomes more effective, easy to distinguish and decide which URLs should be considered as trusted or untrusted by matching them with the standard white list. Most importantly, this matching process will not solve the imbalanced and overlapping classes. Therefore, finding another classifier to handle those two problems is a must.
- (ii) The classifier that is based on the mathematical concept gives more accurate results; A-IFs are automatic and mathematically justified procedures from the relative frequency distributions representing the URL.
- (iii) The fuzzy set is very suitable to deal with the state that contains ambiguity (i.e., fuzziness) and the hesitation URL is the best example of this state.

In effect, each examined instance  $e$  was described (due to the definition of A-IFs) by a triplet: membership value to a *trusted* (bigger) class, non-membership value to a *trusted* class (equal to membership value to an *untrusted* – smaller class) and a hesitation margin that is  $e: (\mu_e, V_e, \pi_e)$  to enhance the possibility of a proper classification of the instances belonging to a *trusted* class. While training the intuitionistic fuzzy

**Table 6 – The best behaviors of A-IF classifier for different number of records in databases.**

| Size of DBs | Evaluation measures |             |             |        |             |
|-------------|---------------------|-------------|-------------|--------|-------------|
|             | Accuracy (AC)       | Recall (TP) | Precision P | F      | $F_{\beta}$ |
| 720         | 0.94                | 0.86        | 0.65        | 0.7404 | 1.040       |
| 1275        | 0.95                | 0.95        | 0.88        | 0.9137 | 0.893       |
| 2000        | 0.98                | 0.99        | 0.89        | 0.9048 | 0.896       |

classifier, the values of the hesitation margins were divided in order to determine which class is the best of the trusted class  $e: (\mu_e, \delta\pi_e, V_{e,+ (1-\delta)\pi_e})$  where  $\delta \in (0.5, 1)$ . As explained in the previous section “How to determine whether that hesitation URL is classified as a trusted or an untrusted URL based on the A-IF’s principle?” In order to evaluate a novel methodology, three experiments were conducted on the databases URL that are different in the number of samples. Table 5 shows the evaluation results of the methodology based on five different evaluation measures (AC, TP, P, F and  $F_{\beta}$ ) for each of the training and testing datasets.

The best value of each evaluation measure generation by methodology for a given dataset is bolded. Table 6 shows the best results were obtained for the different databases by using those five evaluation measures (Fig. 8).

**Algorithm 1.** Pseudo code of the novel methodology in mashup web application

**Input:** Multi requests submitted by the user through the user’s browser on the specific field.

**Output:** Aggregation of trusted URLs related to the user’s request to display them.

**Step 1:** Collection of the URL requests from the Internet (i.e., API Providers, Mashup Providers, and Remote Repository).

**Step 2:** Pass the collection of URLs to the risk filtering data mining (RFDM) algorithm to provide the set of generalized risks and to enhance the quality of the analysis risk by removing a big part of the false risks.

**Step3:** Passes the set of clusters that contain all URLs extraction by step 1 and generation from step 2 to the first classifier to label

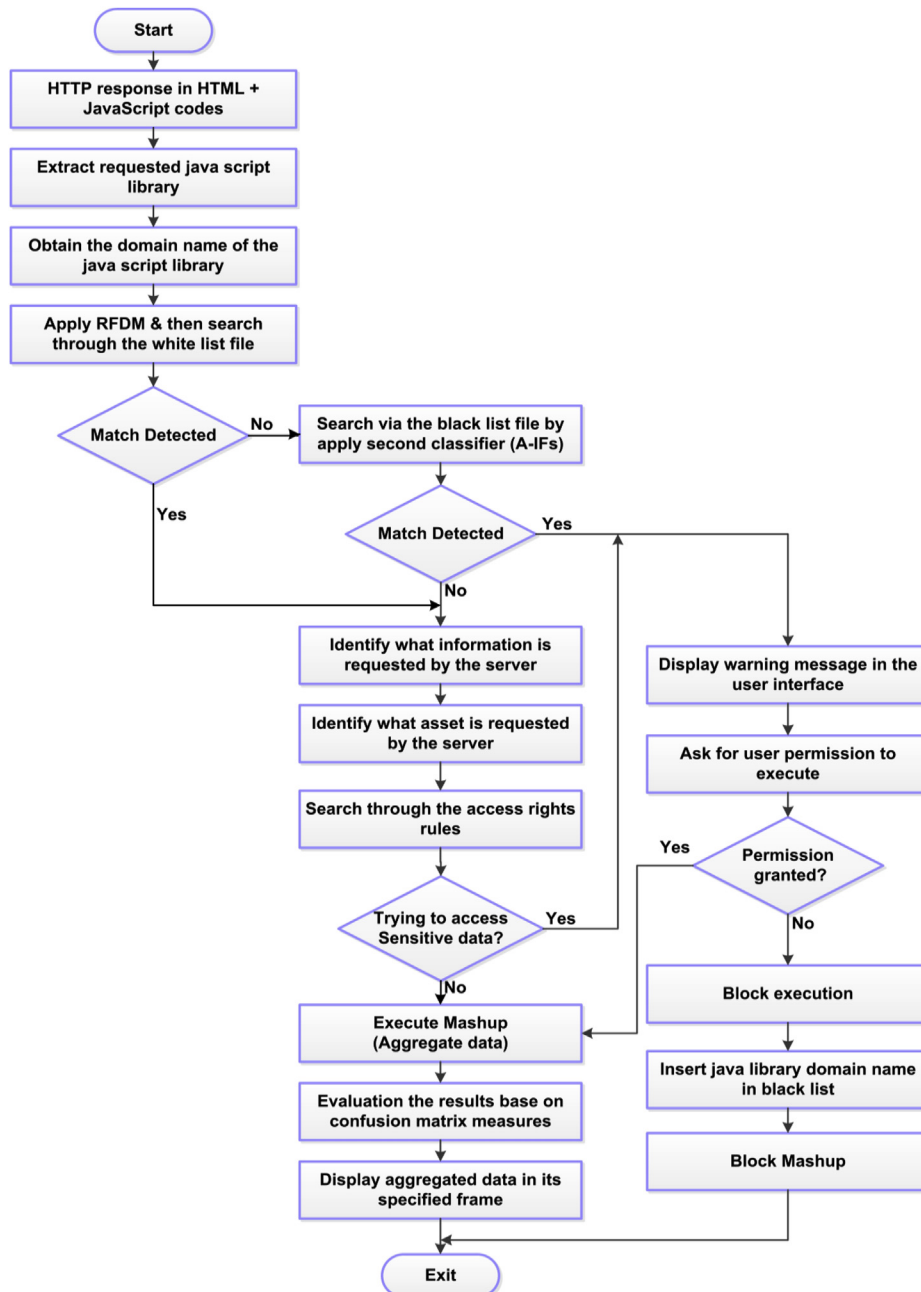


Fig. 8 – Flowchart of the proposed client side mashup security system.

that cluster (i.e., trusted, untrusted and hesitation) then pass only the trusted and untrusted classes to step 5.

**Step 4:** Pass the entire URLs label as hesitation class into the second classifier (A-IFs). These classifiers are based on the values of the membership function classifying the hesitation URL into trusted or untrusted classes then passed as results to step 5.

**Step 5:** Search through the white list file (that is explained in Table 3):

```

*IF domain name is trust
  10 * Identify the information & asset requested by the server
    * Search through the access right rules
    *IF cannot access sensitive data
      20* Aggregation data by executing Mashup
        *Display these URLs for the user
    Else
      30* Display warning message to user
        *Ask the user's permission to execute
        *IF the user's answer is yes, then
          * GO TO 20
        Else
          * Block the Mashup Application
          *Inset Java Library Domain name into black list
        *End if
    *End if
Else
  * Search through the Black List File
  * IF Domain Name is Untrusted
    * GO TO 30
  Else
    * GO TO 10
  *End if
*End if

```

**Step 6:** Evaluate the results based on the five measures computed by the confusion matrix and pass the trusted URL onto the user's monitor.

**Step 7:** End a Novel Methodology that provided the user of a trusted' environment in Mashup web applications.

## 7. Conclusion and future works

This work introduced and proposed a novel approach of mashup security based on risk analysis and mashup application program code source validation. The proposed security framework was composed of three main modules:

- 1) **The Source Monitor Module** which was responsible for validating and authenticating the source origin of the third-party service provider by monitoring the URL links of external sources embedded in the page's HTML script before it was integrated and combined with the rest of the requester document.
- 2) The second module was the **Dataflow Monitor** which was responsible for monitoring the data flow and exchange between the browser and the remote server. It was also responsible for monitoring access to predefined restricted areas in the clients' machines and for preventing the exposure of sensitive, private and protected data.
- 3) The **Execution Monitor** which was responsible for aggregating data and executing the requested JavaScript

methods only was labeled as trusted from the Dataflow Monitor. The sum total of these effectively protected against malicious intrusions when developing mashup meta-applications from a diverse set of primary application sources.

For our future work, we plan to extend and implement this approach by developing an extension to Mozilla Firefox web browser to secure the client side from any malicious activity exploitation through the mashups using JavaScript. This extension will have the authorization to access external XML files. The white-list and black-list will be hosted and updated. The XML file will be updated at runtime execution. The shared XML file will help to expand the trusted and untrusted mashup sources. In this way it will provide a safe and secure environment for the mashups while allowing for open sharing.

## REFERENCES

- Ahmad D. The confused deputy and the domain hijacker. *IEEE Secur Priv Mag* 2008;16(1):74–7.
- Al-Janabi S. Design software for classify objects for air photos & satellite images. LAP Lambert Academic Publishing; 2013. ISBN: 978-3-659-44919-2, [http://www.amazon.com/Design-Software-Classify-Objects-Satellite/dp/3659449199/ref=la\\_B00MBJC2CM\\_1\\_1?s=books&ie=UTF8&qid=1411110264&sr=1-1](http://www.amazon.com/Design-Software-Classify-Objects-Satellite/dp/3659449199/ref=la_B00MBJC2CM_1_1?s=books&ie=UTF8&qid=1411110264&sr=1-1).
- Alur D. The Enterprise web 2.0 Blog. 2007. Retrieved from: <http://blogs.jackbe.com/2007/07/defining-mashups.html>.
- ASP Alliance. Introducing JSON. 2007. Retrieved from, [http://aspalliance.com/1168\\_introducing\\_json](http://aspalliance.com/1168_introducing_json).
- Barth A, Li W. Attacks on JavaScript Mashup Communication. 2011. Retrieved from, <http://www.adambarth.com/papers/2009/barth-jackson-li.pdf>.
- BBB. Retrieved from: <http://www.bbb.org/>; 2014.
- Berry M-J, Linoff G. Data mining techniques: for marketing, sales, and customer support. John Wiley & Sons, Inc; 1996.
- Bianchini D, De Antonellis V, Melchiori M. A recommendation system for semantic mashup design. In: 2010 Workshops on Database and Expert Systems Applications; 2010. p. 159–63. <http://dx.doi.org/10.1109/DEXA.2010.48>.
- Daniel F, Matera M, Weiss M. Next in mashup development: user-created apps on the web. *IT Prof* 2011;13(5):22–9. <http://dx.doi.org/10.1109/MITP.2011.85>.
- Halkidi M, Vazirgiannis M, Batistakis Y. Quality scheme assessment in the clustering process. In: Proceeding of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00), London, UK; 2000. p. 265–76.
- Halkidi M, Batistakis Y, Vazirgiannis M. On clustering validation techniques. *J Intell Inf Syst* 2001;17(2–3):107–45.
- Hilton R. Web application integration using mashups. 2009. Master thesis of University of Texas at Arlington.
- Jackson C, Wang H-J. Subspace. In: Sixth international conference on World Wide Web – WWW'07. New York, United states: ACM Press; 2007. p. 611–20. <http://dx.doi.org/10.1145/1242572.1242655>.
- Liu X-L, Song M, Lianru L. A solution for Mashup Platform based on SOA. In: international conferences on pervasive computing (JCPC); 2009. p. 483–8. <http://dx.doi.org/10.1109/JCPC.2009.5420138>.
- Meng J, Chen J. A mashup model for distributed data integration. In: International conference on management of e-Commerce

and e-Government; 2009. p. 168–71. <http://dx.doi.org/10.1109/ICMeCG.2009.46>.

- Merrill D. Mashups: the new breed of Web app. IBM Developer Works; 2009. Retrieved from, <http://www.ibm.com/developerworks/xml/library/x-mashups/index.html>. 2011.
- Na L, Patel A, Latih R, Wills C, Shukur Z, Mulla R. A study of mashup as a software application development technique with examples from an end-user programming perspective. *J Comput Sci* 2010;6(11):1406–15. <http://dx.doi.org/10.3844/jcssp.2010.1406.1415>.
- Narayana V-A, Premchand P, Govardhan A. Performance and comparative analysis of the two contrary approaches for detecting near duplicate web documents in web crawling. *Int J Electr Comput Eng (IJECE)* 2012;2(6):819–30.
- Patel A, Qassim Q, Wills C. A survey of intrusion detection and prevention systems. *Inf Manag Comput Secur* 2010b;18(4):277–90.
- Patel A, Taghavi M, Bakhtiyari K, Júnior J-C. An intrusion detection and prevention system in cloud computing: a systematic review. *J Netw Comput Appl* 2013;36(1):25–41. <http://dx.doi.org/10.1016/j.jnca.2012.08.007>.
- Programmable web. API Dashboard. 2014. Retrieved from, <http://www.programmableweb.com/apis>.
- Rosenberg F, Khalaf R, Duftler M, Curbera F, Austel P. End-to-End security for Enterprise Mashups. In: Proceedings of the 7th International Joint Conference on Service Oriented Computing (ICSOC'09), Sweden; 2009. p. 389–403. <http://dx.doi.org/10.1007/978-3-642-10383-4>.
- Tanaka M, Kume T, Matsuo A. Web API Creation for Enterprise mashup. In: 2011 IEEE World Congress on Services; 2011. p. 319–26. <http://dx.doi.org/10.1109/SERVICES.2011.83>.
- TRUSTe. Retrieved from: <http://www.truste.com/products-and-services/enterprise-privacy/TRUSTed-websites>: 2014.
- VeriSign. Retrieved from: <http://www.verisign.com/products-services/index.html?tid=gnps>: 2013.
- Yu J, Benatallah B, Casati F, Daniel F. Understanding mashup development. *Internet Comput* 2008;12(5):44–52. <http://dx.doi.org/10.1109/MIC.2008.114>.
- Zou J, Pavlovski C-J. Towards accountable enterprise mashup services. In: IEEE International Conference on e-Business Engineering (ICEBE'07); 2007. p. 205–12. <http://dx.doi.org/10.1109/ICEBE.2007.12>.



**Ahmed Patel** received his MSc and PhD degrees in Computer Science from Trinity College Dublin (TCD), specializing packet switched networks. He is Professor at the Universiti Kebangsaan Malaysia and a Visiting Professor at Kingston University in the UK. His research interests are Computer Networking and Security, Forensic Computing, MANET & VANET, Autonomic Computing, Distributed Systems, Search Engines and Mashup Applications for the Web. He has published over 240 technical and scientific papers and co-authored and edited several book on Computer Network Security, Group Communications and Distributed Search Systems for the Internet.

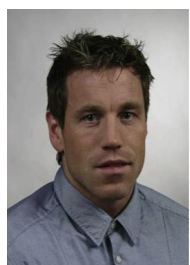
He is a member of the Editorial Board of several International Journals and participated in many Irish, Malaysian and European research projects.



**Samaher Al-Janabi** received her BSc, MSc and PhD degrees in Computer Science from University of Babylon, Iraq, specializing in the development, performance measurement and intelligent analysis of hug/big databases. She is Lecturer in the Departments of Information Networks, Software and Computer Science. Her research interests are Intelligent Data Analysis, Knowledge Discovery, Soft Computing Techniques, Artificial Intelligence, Data Mining, Prediction Techniques, Mobil Services and Network Security. She has published over 30 scientific papers and authored one book on new trends of KDD towards IDA and one book on soft computing techniques. She is a reviewer of several local and international journals.



**Ibrahim AlShourbaji** received his BSc degree in Computer Science from Al-Zaytoonah University, Amman, Jordan, in 2004 and his MSc degree in Information, Network, and Computer Security from New York Institute of technology, USA, in 2007. He is a Lecturer at Jazan University in Saudi Arabia. His research interests are Computer Security, Cryptography, Identification and Authentication, Wireless and Mobile Network Security, Mashup Web Applications and Software Engineering. He has published several papers.



**Jens Myrup Pedersen** is an Associate Professor and Head of the Networking and Security Section, Department of Electronic Systems, Aalborg University. His current research interests include network planning, traffic monitoring, and network security. He obtained his MSc in Mathematics and Computer Science from Aalborg University in 2002, and his PhD in Electrical Engineering also from Aalborg University in 2005. He authored/co-authored over 70 publications for international conferences and journals, and has participated in Danish, Nordic and European funded research projects. He is also a board member of a number of companies involved in technology and innovation.