A New Architecture for Intrusion-Tolerant Web Services Based on Design Diversity Techniques

Sadegh Bejani
Department of Information and Communication Technology, Imam Hossein University, Tehran, Iran
sbejani@ihu.ac.ir
Mohammad Abdollahi Azgomi*
Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
azgomi@iust.ac.ir

Received: 28/Sep/2014 Revised: 22/Aug/2015 Accepted: 12/Sep/2015

Abstract

Web services are the realization of service-oriented architecture (SOA). Security is an important challenge of Web services. So far, several security techniques and standards based on traditional security mechanisms (i.e., encryption and digital signature) have been proposed to enhance the security of Web services. The aim of this work has been to propose an approach for securing Web services by employing the concepts and techniques of software fault tolerance (such as design diversity), which is called *intrusion tolerance*. Intrusion tolerance means the continuous delivery of services in presence of security attacks, which can be used as a fundamental approach for enhancing the security of Web services. In this paper, we propose an architecture for intrusion-tolerant Web services (ITWSs) by using both design diversity and composite Web services techniques. The proposed architecture is called *design-diverse intrusion-tolerant Web service* (abbreviated as DDITWS). For Web service composition, BPEL4WS is used. For modeling and verification of the proposed architecture, coloured Petri nets (CPNs) and the "CPN Tools" are used. We have model-checked the behavioral properties of the architecture to ensure its correctness using this tool. The reliability and security evaluation of the architecture is also performed using a stochastic Petri net (SPN) model and the "SHARPE" modeling tool. The results show that the reliability and mean-time-to-security-failure (MTTSF) in the proposed architecture are improved.

Keywords: Software Security; Intrusion Tolerance; Composite Web Service; Reliability; Petri nets.

1. Introduction

The occurrence of faults in a system is a deviation from correctness or accuracy in the system computations. A system failure means a cessation in the execution of the operation that was expected in a due time [1]. The causes of errors in software systems can be deliberate or unintentional (accidental). The occurrence of any faults or defects in software development process causes error in the software system. The cause of fault in system is unintentional and the incidents due to malicious attacks are rooted out of system. Intrusions are aimed to affect the system integrity, confidentiality or availability (CIA). Intrusion effect realizes on different aspects or incorrect system behaviors. [1]

Architectural evolution of software systems development indicates the widespread use of distributed software systems. [2] In process of software development from 2010 onwards, service-oriented architecture (SOA) has replaced the existing architectures. Web services are the main solution for the realization of service-oriented architecture. [2] Web services have specific features such as interoperability, self-description and self-containing. They use UUDI, HTTP, WSDL and SOAP interaction protocols. [3] Wide range of Web services execution environment, unknown users of Web services and challenges of communication security protocols in Web services interactions make Web services more susceptible to intrusion and attack than traditional software. [2]

Since several security techniques and standards based on traditional security mechanisms (i.e., encryption and digital signature, etc.) have been used to enhance the security of the Web services. The approach of these standards is based on "vulnerability avoidance" and "reducing system vulnerability", which are effective for known attacks. In this standards authentication mechanisms, access control, encryption, firewalls, reconfiguration management and data redundancy technique are used.

A second category of mechanisms is the usage of intrusion tolerance techniques. These techniques are effective for increasing system's tolerance against unknown attacks. In these circumstances, intrusion tolerance means the continuous delivery of services in presence of security attacks, which is a fundamental approach for increasing the security of Web services.

A software system is an intrusion-tolerant system (ITS), if after penetration, its basic services continue their performance and the system prevents from the creation of failure in its security features [3].

Web service technology enables the creation of complex services and provides composition services using simple services. There are two type of Web services: (1) based on simple object access protocol (SOAP), and (2)

* Corresponding Author

RESTful. In this research, we concentrate on SOAP-based Web services.

Composite SOAP-based Web services are composed of several Web services, in order to accomplish common work. [4] BPLE4WS is a standard software for Web service composition. Web service composition process in BPEL4WS makes it possible the realization and implementation of Web service composition. [4] The proposed architecture for ITWS is in the form of a composed Web service that can be implemented in BPEL4WS.

In this paper, we propose a new architecture for intrusion-tolerant Web services (ITWSs). The main approach of the proposed architecture is based on using intrusion tolerance concepts, design diversity techniques and composite Web service techniques. Creating efficient mechanisms for Web service intrusion detection, intrusion containment, intrusion recovery, providing data integrity, confidentiality, availability and neutralizing the influence of intrusions are special architectural considerations in the proposed architecture, which is called *design-diverse intrusion-tolerant web service* (DDITWS).

It is expected that by the realization of the proposed architecture, the developed Web service can continue its operation in the presence of intrusions and can provide the continuity of services without security failures.

The remainder of this paper is organized as follows. In Section 2, related works are reviewed. Section 3 gives an overview of the proposed DDITWS architecture. The security behavior of DDITWS is explained and investigated using coloured Petri nets (CPN) model of the architecture by using CPNs Tools is presented in Section 4. The results of the reliability and security evaluation of the proposed architecture are also given in this section. For this purpose, a stochastic Petri net (SPN) model and the SHARPE tool is used. The results show that the reliability and mean-time-to-security-failure (MTTSF) are improved. The paper will be concluded in Section 5.

2. Related Work

In the following, we briefly review the existing standards, techniques and so on for the security of Web services:

- *Web service security standards:* According to [5], various specifications discussed about Web service security. The WWW Consortium has developed various specifications, such as WS-Security (WSS), WS-Federation, WS-Authorization, WS-Policy, WS-Trust, WS-Authentication and WS-Privacy for Web service security. These standards do not protect Web services totally. For example, WS-Security specifies how integrity and confidentiality can be enforced on messages, allows the communication of security tokens and provide end-to-end security.
- *Vulnerability detection techniques:* There are best practices of software testing and a lot of tools, languages and techniques in order to analyze and detect vulnerabilities in software systems. [5] But, an evaluation of several commercial versions of vulnerabilities scanners showed that these tools are primarily limited to low coverage of existing vulnerabilities and high percentage of false positives. Few techniques and tools (such as Netsparker) exist for vulnerability scanning of SOAP-based web services.
- *Intrusion/prevention techniques:* There are intrusion prevention an intrusion detection techniques, which are not effective against new or unknown attacks. [7]
- *Dependable computing techniques:* The existing solutions for dependable Web services are divided into two categories: fault tolerance techniques (such as active and passive replications), and the use of design diversity. A dependable architecture for Web services that uses multi-version techniques is introduced in [7]. In [3], by using design diversity technique and Web services business process execution language (WS-BPEL), the authors have proposed a useful and flexible architecture for dependable Web services.
- *Software fault-tolerance techniques:* There are two types of software fault-tolerance techniques: single-version and multi-version that are used for security improvement. [8] Fault tolerance techniques, including replication, check-pointing and message logging, in addition to reliable messaging and transaction management for which Web services specifications exist. The authors of [8] have discussed how those techniques can be applied to the components of Web services involved in the business activities to make them dependable.
- *Architectures for intrusion-tolerant systems:* There are important architectures such as self-cleansing intrusion tolerance (SCIT), scalable intrusion-tolerant architecture (SITAR) for distributed services and malicious- and accidental-fault tolerance for Internet applications (MAFTIA) for intrusion-tolerant systems [9]. The assumption in the SCIT architecture is that the intrusion detection mechanism is not able to detect unknown attacks and Web services cleansing is necessary. The SITAR architecture is used for the intrusion tolerance of commercial off-the-shelf (COTS) systems. Fault tolerance techniques, such as redundancy and design diversity, are used in the SITAR architecture.
- *Fault tolerance architecture for Web services:* In [10], authors have proposed a new fault-tolerant architecture for Web services named FTWeb. In [12], authors have explained a multi-layer architecture for ITWSs. The specific goal of the architecture is to use single-version software fault-tolerance concepts in the case of malicious failures.

3. The Proposed Architecture

In this section we introduce an architecture for intrusion-tolerant Web services, which is called *design-diverse intrusion-tolerant web service* (DDITWS). The aim of the proposed architecture is to construct and strengthen the capabilities of Web services against both known and unknown security attacks. The proposed architecture is based on the following concepts and techniques:

I. Theoretical concepts of intrusion tolerance approach: In these concepts, the main indicator of intrusion tolerance and their requirements are expressed. In [12], the main indicators of an intrusion-tolerant system are defined as follows:

- Maintaining the integrity of the system's operational environment,
- Detecting intrusions,
- No failure in the security features of system, such as confidentiality, integrity and availability, and
- Stability in the system's operations.

II. Composite Web service technology: In the composite Web service technology used in the proposed architecture, the aggregation and composition of the main Web service with the supplementary Web services that provide the abilities of intrusion tolerance is performed. The demand of ITWS is a complex request. Composite Web service technology provides the possibility to implement the proposed architecture and the ability to meet complex demands.

III. Classical fault tolerance techniques: Intrusion tolerance and fault tolerance are common principles. Both of them focus on service continuity in abnormal conditions. Fault-tolerant techniques can provide appropriate policy to create a conceptual framework, that theories are developed during intrusion tolerance. To establish appropriate mechanisms for intrusion tolerance in Web services, fault tolerance techniques are used [14]. In the proposed architecture, redundancy, design diversity and replication techniques are used.

3.1 Components of the Architecture

The main motivation of the proposed DDITWS architecture is based on the facts that software systems development and maintenance cannot be without vulnerabilities. Behavior-based intrusion detection systems are not able to provide intrusion-tolerant system. To provide the continuity of services, it is necessary that the impact of attacks be managed. The overall view of the proposed architecture is shown in ("Fig. 1").

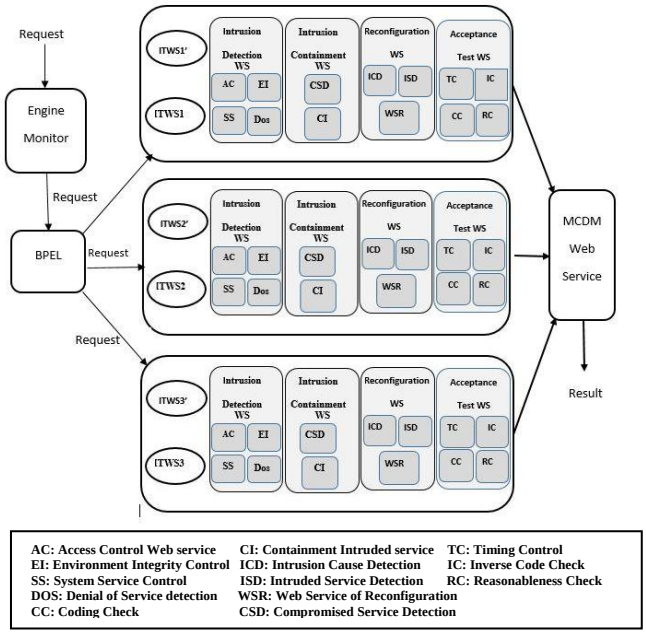

Fig. 1. Overall view of the DDITWS architecture

Intrusion tolerance capabilities through Web services in the DDITWS architecture can be constructed in the forms of features for "intrusion tolerance", "intrusion containment", "reconfiguration", and "decision maker". Appropriate intrusion-tolerant mechanisms causes the continuation of services and preventing security failure in system. Each intrusion tolerance feature in the DDITWS architecture is embodied in the form of a Web service. The composition of these Web services is achieved in the DDITWS architecture. The components of the composite Web service in the DDITWS architecture are explained in the following.

3.2 Intrusion Detection Composite Web Service

The use of resilient mechanisms is necessary for providing the service availability in systems [9]. Resilient mechanisms are meant to have a facility in the event of penetration. Intrusion-tolerant and reconfiguration mechanisms are resilient mechanisms. Intrusion detection methods are possible in knowledge-based or behavior-based. [8]. In the DDITWS architecture, intrusion detection is done by a component of the composite Web service that is based on behavior changes. Intrusion-detection Web service has a main role in providing intrusion tolerance in the DDITWS architecture. Using intrusion detection composition Web service is appropriate for new or unknown attacks.

In the DDITWS architecture, the tasks of the intrusion-detection Web service are as follows:

- Detecting a variety of attacks on Web services among either known or unknown.
- Updating the records of attacks and intrusion patterns.
- Provisioning of the analysis and detection of attacks and system failure causes.
- Acceptance testing on the response of Web service request.

- Identification of the denial-of-service (DoS) attacks on Web service.

According to ("Fig. 1"), intrusion detection composite Web service to perform its tasks includes multiple Web services as follows:

- The Web service of access control to resources (AC): This Web service controls accesses to resources and checks whether it is as expected or not.
- The Web service of environment data integrity control (EI): This Web service by comparing the data files in the operating environment while providing services with the information of data files before providing services, determines whether or not an attack is occurred.
- The Web service of system services controller (SS): This Web service checks certain system services that all of them had already determined, ordered and fully executed.
- The Web service of detect DoS attack: In traditional confronting techniques to DoS attacks, there are two basic steps as: (1) detecting real or fake IP addresses, and (2) detection of traffic conditions for DoS attacks [8]. In the DDITWS architecture, Web service DoS has a task of detecting of the traffic conditions of DoS attacks. The DoS attack occurs in each of the following two modes:

1- *(required-time to respond to a previous request) + (last request-time) > input during Web service request*
2- *Threshold number of requests > [(request-time – arrival time of the first request) / number of requests]*

3.3 Intrusion Containment Composite Web Service

The aim of intrusion containment is the encapsulated area of intrusion and preventing intrusion. This will result in reduce the service level of the system. Intrusion graph is an efficient tool of intrusion containment [15], which is used in DDITWS. Intrusion graph is a directed graph that shows intrusion propagation paths from a service to another service [15]. In intrusion graph, each node represents an intrusion target and any edge that is used to show dependencies between intrusion targets. Intrusion alerts sent by intrusion detection components, are mapped onto the intrusion graph. Intrusion containment components use intrusion graph and breaks through a communication channel section and other sections and prevent the spread of intrusion.

In DDITWS, containment composed Web service has two functions as locating and stopping the spread of intrusion through. To limit the intrusion, it is necessary to prepare the intrusion-graph data. Intrusion containment feature and limiting the intrusion is an important attribute in intrusion-tolerant systems. According to ("Fig. 1"), intrusion containment composite Web service is composed of multiple Web services as follows:

1- The Web service of *compromised service detection* (CSD): This Web service after getting information about intrusion to Web service identifies the compromised service in the attacked Web service.
2- The Web service of *containment intruded service* (CI): After the detection of the compromised service, this Web service gives the information of the compromised service and breaks their communications.

3.4 Recovery and Reconfiguration Composite Web Service

In intrusion-tolerant systems, after intrusion detection and containment, it is necessary that the compromised sections be inactive and the compromised components be reconfigured. In DDITWS, it is the responsibility of the recovery and reconfiguration composite Web service.

In DDITWS, for recovery from intrusion, several techniques, such as fragmentation, scattering and data redundancy can be used. The fragmentation and scattering techniques makes it possible if there is an unauthorized access to the data, all the valuable data should not be available. Using data redundancy technique makes it possible that after intrusion detection, intrusion masking is possible and the system returns to an optimal state. In DDITWS, for reconfigure a compromise component, the level of active service on the Web service is checked. If the service level is not satisfactory, the redundant service will be replaced and the compromised service will be reconfigured. Until replacing, service will be in the *graceful degradation* state. Intrusion recovery Web service and the compromised component reconfiguration are among intrusion-tolerant system attributes. Intrusion containment composite Web service is very important in DDITWS. It is necessary that reconfiguration and intrusion recovery be performed automatically. If reconfiguration is not automatic, the occurrence of distributed DoS (DDoS) attacks is prohibited. The recovery and reconfiguration composite Web service is a main component in DDITWS that is a key component of intrusion tolerance.

According to ("Fig. 1"), the reconfiguration composite Web service is composed of multiple Web services as follows:

- The Web service of *intrusion cause detection* (ICD): This Web service determines the main cause of the intrusion to the Web service.
- The Web service of *intruded service detection* (ISD): This Web service gets the information about intrusion and determines the intruded service.
- The Web service of *reconfiguration* (WSR): This Web service is responsible for an important task, that is, after intrusion detection and containment, it is necessary that the intrusion should be covered and the compromised service is managed by using replication technique.

3.5 Acceptance Test Composite Web Service

The acceptance test composite Web service checks the response of Web service requests. According to ("Fig. 1"), the acceptance test composite Web service is composed of multiple Web services as follows:

- The Web service of *timing control* (TC): This Web service checks whether a Web service deadline is expired or not.
- The Web service of *inverse code check* (IC): This Web service checks the correctness of the response to the Web service's request. If the answer is incorrect, then attack to the Web service is announced.
- The Web service of *control results* (RC): This Web service checks whether the results of the Web service is within the acceptance range or not.
- The Web service of *coding check* (CC): This Web service checks the validity of data transfer operations. This Web service uses encoding data technique.

3.6 Multi-Criteria Decision Maker Web Service

Based on the DDITWS architecture, for each request, three redundant Web services are provided and in this case, to determine the final outcome, using a decision maker Web service is necessary. ("Fig. 2") shows the structure of the multi-criteria decision maker composite Web service.

In multi-criteria decision maker composite Web service, for organizing redundant voters and acceptance monitors, N-self checking technique is used. Based on N-self checking technique, at any time, only one voter is enabled. There are three voters in decision maker composite Web service. All voters in decision maker composite Web service have an equal number of inputs, output type and input types. In multi-criteria decision maker composite Web service, in addition to the input values, also Web service trust value are part of inputs and the end result is effective.

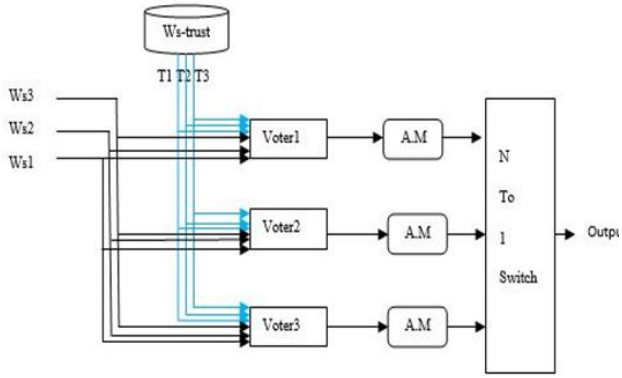

Fig. 2. The structure of the multi-criteria decision maker

3.7 Structural Features of DDITWS

The structural features of the DDITWS architecture are as follows:

1- Using redundancy technique in system causes the enhancement of the system reliability. In the proposed architecture to respond to any requests, three redundant Web services are used.
2- In order to reduce the probability of similar vulnerabilities in redundant Web services (i.e., *ITWS1*, *ITWS2* and *ITWS3*) and increasing the tolerance against similar attacks, the design diversity technology is used.
3- Any of the main Web services (i.e., *ITWS1*, *ITWS2* and *ITWS3*) along with several other Web services that provide the intrusion tolerance, are constructed as composite Web services.
4- Using replicated Web service technique, recovery from critical Web services is possible.
5- The recovery strategy is used in the proposed architecture based on using intrusion masking and replication techniques.
6- In each of the main Web services (i.e., ITWS1, ITWS2 and ITWS3), intrusion containment and reconfiguration composite Web service are used. Their main tasks include intrusion containment and the influence of recovery and reconfiguration of compromised component.
7- Using redundant Web services in the proposed architecture, shows the necessity of using the decision maker Web service within it.

3.7.1 Relationship between Intrusion Tolerance and Design Diversity Technique in DDITWS

Intrusion tolerance means that service continues in the presence of attacks. Intrusion tolerance is a non-functional requirement in systems. Having an intrusion tolerant Web service is a complicated demand. The intrusion tolerance capability of the DDITWS architecture, as several subsidiary Web services is combined with a main Web service. To achieve complete intrusion tolerance, it should be combined the intrusion avoidance and intrusion tolerance capabilities. The DDITWS architecture of intrusion prevention capabilities uses redundancy and design diversity techniques. The use of design diversity technique in the development of a variety of components reduces the same vulnerabilities in the components. Reducing the same vulnerability of the redundant components, similar attacks are successfully reduced DDITWS. Similarly, using design diversity technique in decision-maker Web service reduces the vulnerability of voters. The fundamental role of using design diversity technique is to increase the intrusion tolerance in the DDITWS-based Web services.

3.7.2 Structure of the DDITWS in BPEL

The structure of composite Web service involves all internal Web services, the order of the execution of the internal Web services and data transferring between them [16]. For each composite Web service, defining specific rules on the application level means the determination of composite Web service structure [16]. The BPEL4WS by defining specific rules on the application level specifies Web services participating in composite Web service, the order of the execution of them and data transferring between internal Web services.

There are different tools in design area of BPEL that they may make the use of the composition and execution of Web services in composite Web services of the DDITWS

architecture. Designing of ITWS can be done in the BPEL environment. (“Fig. 3”) shows the structure of ITWS based on the DDITWS architecture in the BPEL form.

As in the DDITWS architecture, each composite Web service involves several internal Web services as shown in (“Fig. 4”). Internal Web services are organized by the structures such as “sequence”, “flow”, “scope” and so on of BPEL.

4. Modeling and Evaluation

4.1 The Security Behavior of DDITWS

For modeling the security behavior of the DDITWS architecture, modeling the behavior of the “attacker” and “system response to attack” are necessary. ITWS’s security behavior can be modeled by using the definition of security states, the interaction of them and the state-transition diagram (STD). (“Fig.5”) shows the STD of Web service’s behavior in the DDITWS architecture. Successful exploitation of security holes by attacker is a main factor causing active attacks occurs.

In the DDITWS architecture, several strategies are intended to create different level of security. According to (“Fig. 5”), in the security behavior model of the Web service in DDITWS architecture, at the beginning, the Web service is in “good” state. Create new conditions such as change of Web service information, change user permissions, prolonged duration of services, change in accounting rules and not properly performed system services, causes Web service state change into “vulnerable” state.

Fig. 3. The structure of the ITWS in BPEL

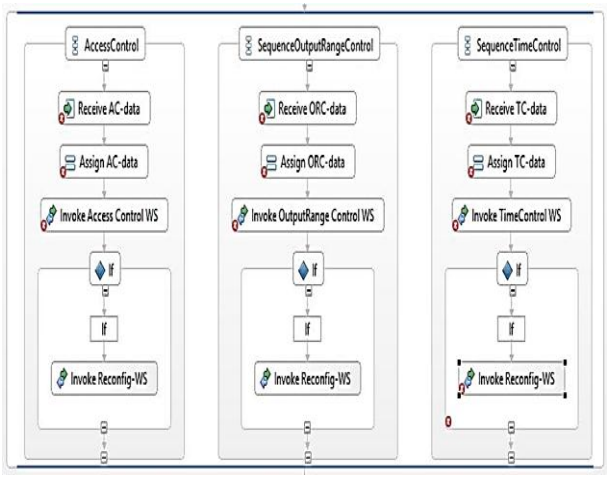

Fig. 4. Instance of internal Web services in DDITWS

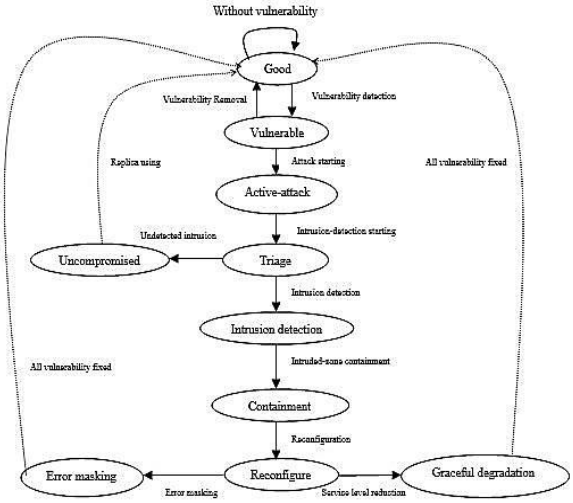

Fig. 5. The state-transition diagram of DDITWS

The vulnerability can be identified as a security hole. Make any sense of vulnerability in the Web service means a violation of Web service’s security policy. In fact, after vulnerability identification, it may be exploited be attackers.

The realization of the possibility of exploiting security holes; it will put Web service in “active attack” state. With each entry into active attack mode, the Web services become influential. In vulnerable mode, using tools such as firewall, may lead to identifying and eliminating the vulnerabilities and Web services can still be in “good” mode. If the error has not been covered in Web service and the compromised component fails, the Web service will be in “uncompromised” mode.

In the DDITWS architecture, intrusion recovery strategy uses intrusion masking and replication techniques. Based on the reinforced intrusion recovery strategy, the data used intrusion masking technique involves data fragmentation, data scattering and redundancy against intrusion. Critical services’ recovery is mostly done through replication technique. Using data redundancy technique causes Web service to be in “good” mode and the service delivery of the Web service will be continued.

According to (“Fig. 5”), if the intrusion is detected in the use mode, Web service goes to “triage” mode. In this context, the following two conditions are most likely:

1- A compromization has taken place and the intrusion detection mechanism cannot detect that, so it is in “uncompromised” state.
2- The intrusion detection component successfully detects intrusion, so Web service is in ‘‘intrusion detection” mode.

After intrusion detection, appropriate message will be sent to intrusion recovery and compromised component reconfiguration, messages received through intrusion detection component, Web service to deliver ‘‘intrusion containment” mode for limiting the restricted area and prevent intrusion expansion.

Another security state is the “reconfiguration” state. In the reconfiguration state, reconfiguration process is based on the defined policies. Also, by using data redundancy and replication techniques, Web services are in “masked-error” state. The reconfiguration mode may lead to new security mode that is named “graceful degradation”. In this case, only essential services will continue and other services will be stopped.

Essential services are the services that continue and will not stop in system, even in intrusion mode. If all strategies are predicted to fail, Web service mode will be the “failed” mode. This condition should not occur in ITWS. The Web service that returns to normal mode after a successful attack is shown in Web service security behavior model as dashed-line.

4.2 The DDITWS Architecture Modeling and Formal Analysis Using CPN Tools

For the analysis of the functionality of the proposed architecture, we have modeled it using coloured Petri nets (CPNs or CP-nets) and then, the behavioral characteristics are analyzed. For this purpose, we have used CPN Tools.

4.3 Modeling and Analysis of the Architecture

The main model (i.e., the home page in CPN Tools terminology) of the DDITWS architecture is shown in (“Fig. 6). Because coloured Petri nets are graphical models, this feature provides the opportunity to review the changes and it can help to investigate how each of the sections in the system works. [17]

The home page of the model consists the units named *intrusion detection, intrusion containment, intrusion recovery, reconfiguration, and decision maker* that each of these units are modeled by substitution transitions in the model.

Places in the model are ports for inputs and outputs. In home page, the *start* place is a driver for input Web service request. There is a token inside it, causes a *Get_WS_Access_Archive* transition be enabled.

The incoming Web service request is transmitted to the place named WS_*Access_Archive*. By placing a token in the *WS_Access_Archive* place, the *dispatcher* transition sets *WS1*, *WS2* and *WS3* places simultaneously.

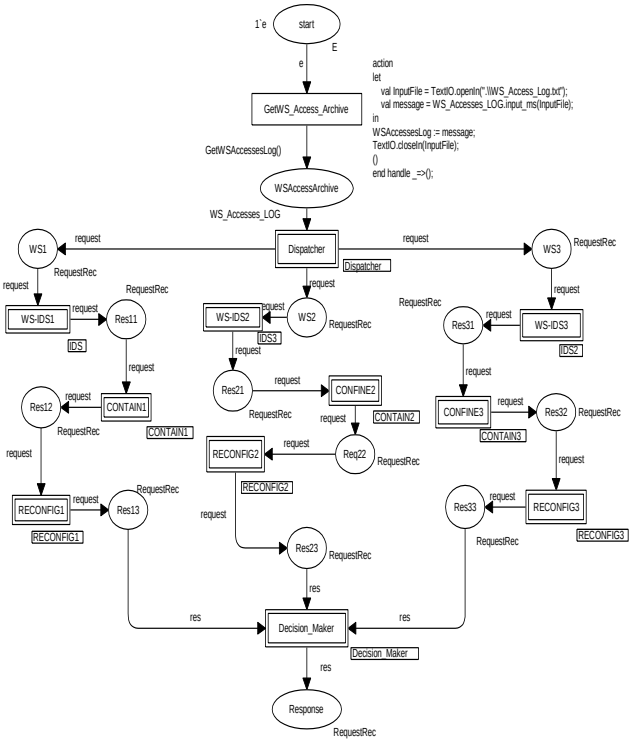

Fig. 6. The home page of the DDITWS CPN model

By submitting to three redundant of Web services at the same time, each of the *IDS1*, *IDS2* and *IDS3* transitions has the responsibility for the result of the corresponding Web service. The *IDS1*, *IDS2* and *IDS3* transitions have the duty to compare the behavior of the system in service with the expected behavior. Each of the *IDS1*, *IDS2* and *IDS3* transitions have equivalent intrusion detection units in the DDITWS model. In case of intrusion detection, the corresponding intrusion containment units (i.e., *CONFINE1*, *CONFINE2* and *CONFINE3*) are active.

Each of the intrusion containment units by limiting the infected area, prevents the spread of influence and sends the intrusion messages to correspond reconfiguration transition (i.e., *CONFIG1*, *CONFIG2* or *CONFIG3*).

Based on the determined tasks for the reconfiguration unit, intrusion recovery and compromised component reconfiguration are done. At the end, the results of Web services, in order to determine the final result, will be sent to the decision maker unit. The decision maker unit is an intrusion-tolerant composite Web service that determines the final result of the redundant Web services.

4.4 Properties of the Architecture

In the CPN Tools environment, it is possible that measuring the behavioral characteristics of the specified model. [19] The results of formal analysis of the behavioral characteristics in DDITWS are shown in (“Fig. 7), which are as follows:

- “None” value for the *Dead Marking* attribute shows that each transition in the proposed model

are live and the system of the DDITWS architecture is deadlock-free.
- "None" value for the *Dead Transition Instances* attribute shows that the system based on DDITWS is non-terminating in all states.
- The value "136" for the *Live Transition Instances* attribute shows that the modeled system is live.
- The value "No infinite occurrence sequences" for *the Fairness Properties* shows that all transitions are executed fairly.
- The value "No infinite occurrence sequences" for the *Fairness* attribute in system model shows that the system will execute in all possible states.

The behavioral characteristics of the model ensure the correctness of the functionality of the architecture.

State Space	**Liveness Properties**
Nodes: 3672	Dead Markings
Arcs: 15840	*None*
Secs: 9	Dead Transition Instances
Status: *Full*	*None*
SCC Graph	Live Transition Instances
Nodes: 3672	136
Arcs: 15840	Fairness Properties
Secs: 0	*No infinite occurrence sequences*.

Fig. 7 The results of the analysis of the characteristics of DDITWS

4.5 Evaluation of the Measures

Two measures, i.e., "reliability" and "mean-time-to-security-failure", are important in the Web services based on the DDITWS architecture. We examine these measures in this section.

4.5.1 Evaluation of the Reliability Measure

The reliability of a Web service based on the DDITWS architecture may be evaluated without their implementation details using a stochastic Petri net (SPN) model and the SHARPE modeling tool. For the evaluation of DDITWS's reliability block diagram is shown in ("Fig. 8"), which includes the following:
- The proposed architecture consists of three redundant Web services.
- In construction of redundant Web services in DDITWS architecture, design diversity technique is used, so the reliability of the replications is different. Using design diversity technique reduces the same vulnerabilities in the replicated Web services.
- To determine the final result of the redundant Web services in the DDITWS architecture, a decision maker unit is used. The decision maker unit uses three redundant voters on an N-self checking structure. In the construction of redundant voters, design diversity technique is also used, so the reliability of these redundant voters will also be different.

By Eq. (1), the reliability of the DDITWS architecture can be calculated as follows:

$$R_{Web\ services} = R_{ws1} * R_{ws2} * R_{ws3} - (R_{ws1}.R_{ws2}) - (R_{ws1}.R_{ws3}) - (R_{ws2}.R_{ws3}) + R_{ws1} + R_{ws2} + R_{ws3}$$
$$R_{voters} = R_{voter1} * R_{voter2} * R_{voter3} - (R_{voter1}.\ R_{voter2}) - (R_{voter1}\ .R_{voter3}) - (R_{voter2}.\ R_{voter3}) + Rvoter1 + R_{voter2} + R_{voter3} \quad (1)$$
$$R_{system} = R_{Web\ services} * R_{voters}$$

For evaluating the reliability of DDITWS, experimental values of the reliability of each component is given in Table 1.

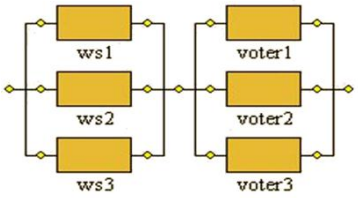

Fig. 8. Block diagram of the DDITWS architecture

Table 1. Experimental values of the reliability of each components

Bock-name	**Reliability**
Web service1	0.81
Web service2	0.78
Web service3	0.82
Voter1	0.91
Voter2	0.93
Voter3	0.95

As shown in ("Fig. 9), the reliability of each Web service is less than the reliability of ITWS. The reason is due to using the design diversity technique in the DDITWS architecture. By repeating the experiment with new values for Web service's reliability, the overall result will not change. The overall result is the reliability of ITWS greater than the reliability of each Web service. As expected, this reliability evaluation has assured that the security of the DDITWS architecture is increased.

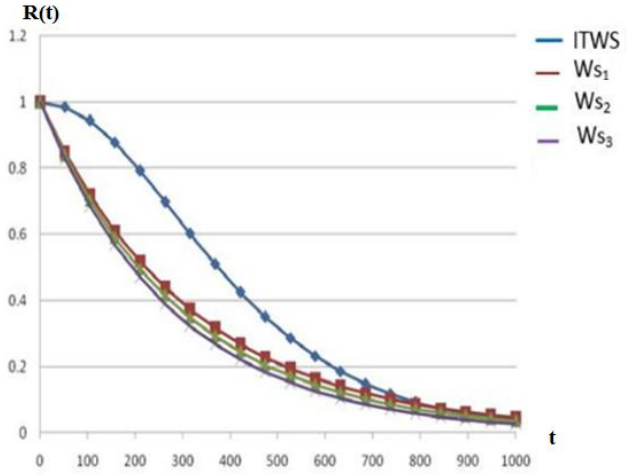

Fig. 9. The diagram comparing the reliability of three Web services with the ITWS

4.5.2 Evaluation of the MTTSF Measure

Mean-time-to-security-failure (MTTSF) is an important measure in the survivability evaluation of intrusion-tolerant systems. The SPN model of the

DDITWS is shown in (“Fig. 10”). The corresponding Markov model is also shown in (“Fig. 11”).

The evaluation of the system performance is often related to their behavior after long time, until a system steady state is achieved. In system steady state, the impact of initial conditions and system behavior into a state regulated system of compensation. In (“Fig. 11), the transfer rate from the BPEL state to the execute state of each of the Web services (i.e., WS1, WS2 and WS3) are equal.

(“Table 2”), shows the results of the solution of the model using SHARPE tool. This table shows the selected transition rates used in the model, too.

In (“Fig. 12”), the MTTSF of a traditional Web service is compared with DDITWS.

In explaining the attributes of the DDITWS architecture, it was said that using design diversity technique in developing redundant components causes reducing the Web service common vulnerabilities and increasing the Web service intrusion tolerance. The higher level of design diversity used, the lower the failure rate of the redundant Web services. This case causes increasing the Web service MTTSF. Increasing the MTTSF in Web services based on the proposed architecture is a factor to increase the availability and intrusion tolerance of the Web services based on the architecture.

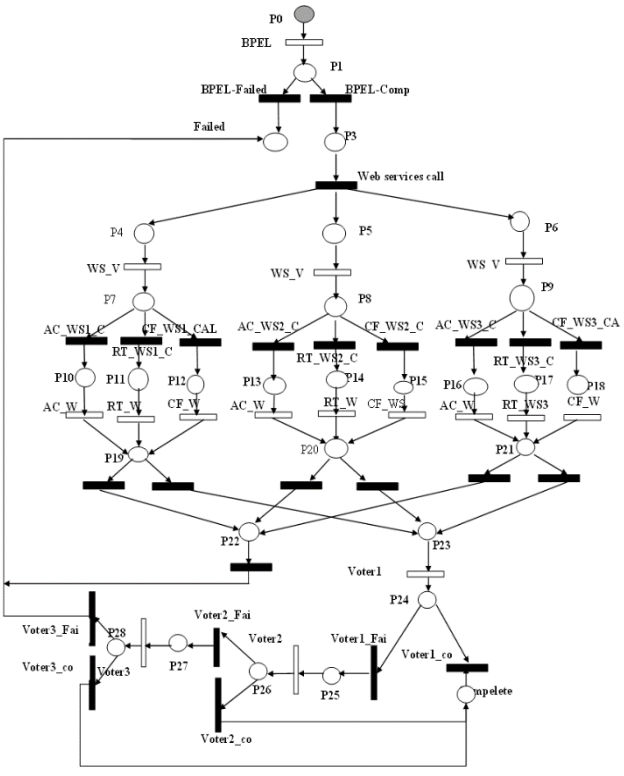

Fig. 10. The SPN model of the ITWS

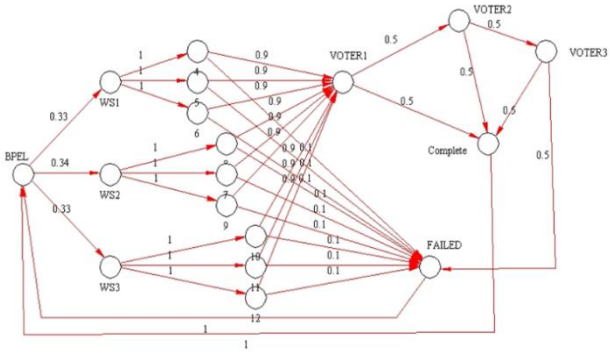

Fig. 11. The Markov chain model corresponding to the SPN model of Fig. 10.

Table 2. Transition rates and calculated MTTSF measure

IN1	IN2	MTTSF in DDITWS	MTTSF in DDITWS
0.1	0.9	39.1	29.0
0.2	0.8	20.1	14.0
0.3	0.7	13.7	9.0
0.4	0.6	10.5	6.5
0.5	0.5	8.6	5.0
0.6	0.4	7.4	4.0
0.7	0.3	6.5	3.3
0.8	0.2	5.8	2.8
0.9	0.1	5.2	2.3

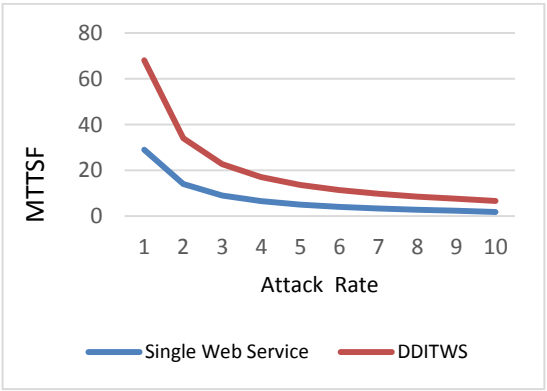

Fig. 12. The MTTSF values of the DDITWS compared with a traditional Web service

5. Conclusions

In this article, we presented a new architecture for intrusion-tolerant Web services (ITWSs). The proposed architecture, abbreviated by DDITWS, uses classical fault-tolerance techniques, such as design diversity, redundancy, N-self checking and acceptance testing. Also, the approach uses the theoretical concepts of intrusion-tolerant systems, which are used in the proposed architecture. In the proposed architecture, composite Web service technique is used, with several Web services. Composite Web service structure consists all internal Web services, their execution order and how to the data is transferred between them in the application level.

In order to understand the security behavior of Web services in the DDITWS architecture, the interaction is evaluated against the attempts of attackers.

To study the components functionality, Web services in the DDITWS architecture are modeled using coloured Petri nets. Behavioral characteristics of the proposed architecture are also analyzed. The results show that the functionality of all components is correct.

The mean-time-to-security-failure (MTTSF) and the reliability measure for the proposed architecture are evaluated using stochastic Petri nets and the SHARPE tool. Evaluation results show that the reliability and MTTSF of the proposed architecture has also increased.

The proposed architecture is a complex one and is dedicated to SOAP-based Web services. There are multiple components, i.e., subsidiary Web service, in the proposed architecture. These multiple Web services should be designed based on design diversity rules and techniques. In practice, achieving diverse versions for the same software is quite difficult. Therefore, this is the main disadvantage of the proposed architecture.

The proposed architecture may be used for other types of Web services, such as RESTful Web services. It can also be used to devise intrusion-tolerant architecture for other types of software systems.

In future, we intend to implement a prototype of a real Web service, such as an electronic commerce application. This prototype implementation can be used to evaluate other aspects of the proposed architecture.

References

[1] E. Dunrova, Fault Tolerant Design: An Introduction, Department of Microelectronics and Information Technology, Royal Institute of Technology, Stockholm, Sweden, 2008.

[2] E. Cerami, Web Services Essentials, First Edition ed., United States of America: O'Reilly, 2002.

[3] D. Gorton, Extending Intrusion Detection with Alert Correlation and Intrusion Tolerance, M.S.Thesis, Chalers Universityof Technology, Sweden, 2003.

[4] D. Gorton, "Using WS-BPEL to Implement Software Fault Tolerance for Web services," in Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, 2006, pp. 126-133.

[5] E. Martin and M. Salas, "Security Testing Methedology for Vulnerabilities of XSS in Web Services and WS-Security," *Electronic Notes in Theoritical Computer Science*, Vol. 302, 2014, pp. 133-154.

[6] J. Reynolds, "The Design and Implementation of an Intrusion Tolerant System," in Proceedings of the International Conference on Dependable Systems and Networks, 2002, pp. 285-290.

[7] M. Abdollahi Azgomi and E. Nourani, "A Dependable Web Service Architecture Based on Design Diversity Techniques and WS-BPEL," Iranian Journal of Electrical and Computer Engineering (IJECE), Vol. 11, No. 1, 2013, pp. 1-4.

[8] P. Veríssimo, N. Neves and M. Pupo Correia, "Intrusion-Tolerant Architectures: Concepts and Design," Lecture Notes in Computer Science, Vol. 2677, 2003, pp. 3-36.

[9] L. Quyen, S. Nguyen and S. Aurn, "Comparative Analysis of Intrusion-Tolerant System Architectures," IEEE Security and Privacy, Vol. 9, No. 4, 2011, pp. 24-31.

[10] T. Giuliana Santos, L. Cheuk Lung and C. Monetez, "FTWeb: A Fault Tolerant Infrastructure for Web Services," in Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference, 2005, pp. 95-105.

[11] Z. Aghajani and M. Abdollahi Azgomi, "A Multi-Layer Architecture for Intrusion-Tolerant Web Services," International Journal of u- and e-Service, Science and Technology, Vol. 1, No. 1, 2008, pp. 73-80. .

[12] A. Sood, "Securing Web Servers Using Intrusion Tolerance (SCIT)," in Proceedings of the Second International Conference in Dependability, 2009, pp. 60-65.

[13] W. Barry Johnson, Design and Analysis of Fault-Tolerant Digital Systems, University of Virginia: Charlottesville: Addison-Wesley Publishing Company, 1989.

[14] W. Yu-Sung, F. Bingrui, Y.-C. Mao, B. Saurabh and S. Eugene," Automated Adaptive Intrusion Containment in Systems of Interactive Systems," Computer Networks, Vol. 51, 2007, pp. 1334-1360.

[15] L. Chen, "A Method for Analyzing and Predicting Reliability of BPEL Process," Journal of Software, Vol. 4, No. 1, 2009, pp. 11-18.

[16] D. Mukherjec, P. Jalote and M. Gowri Nada, "Determing QoS of WS-BPEL compositions," Lecture Notes In Computer Science, Vol. 5364, 2008, pp. 378-393.

[17] "CPN Tools," CPN Group, University of Aarhus, [Online]. Available: http://wiki.daimi.ac.dk/cpntools.

[18] K. Jensen and L. M. Kristensen, Coloured Petri Nets, Modeling Validation of Concurrent Systems, Springer, 2009.

Sadegh Bejani received B.Sc. in Computer Engineering (Hardware) (1985) from Tehran University and M.Sc. and Ph.D. degrees in Computer Engineering (Software) (1996 and 2015, respectively) from Imam Hossein (a.s.) University. His research interests include software security, intrusion-tolerant systems, analytical modeling and computer simulation. He is an Assistant Professor at School of Information and Communication Technology, Imam Hossein (a.s.) University, Tehran, Iran.

Mohammad Abdollahi Azgomi received B.Sc., M.Sc. and Ph.D. degrees in Computer Engineering (Software) (1991, 1996 and 2005, respectively) from Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. His research interests include modelling and evaluation of security, privacy and trust, and dependable and secure software development. He is an Associate Professor at School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.