# PROTECTION OF XML DENIAL-OF-SERVICE AND FLOODING ATTACKS IN SOAP-BASED WEB SERVICES USING MIDDLEWARE TOOL

ABBAS AHMED ALI QASSEM AL-ASRI

UNIVERSITI KEBANGSAAN MALAYSIA

# PROTECTION OF XML DENIAL-OF-SERVICE AND FLOODING ATTACKS IN SOAP-BASED WEB SERVICES USING MIDDLEWARE TOOL

ABBAS AHMED ALI QASSEM AL-ASRI

DISSERTATION SUBMITTED IN PARTIAL FULFILMENTS OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

FACULTY OF INFORMATION SCIENCE AND TECHNOLOGY
UNIVERSITI KEBANGSAAN MALAYSIA
BANGI

2018

# PERLINDUNGAN SERANGAN NAFI KHIDMAT (DOS) DAN SERANGAN FLOODING BAGI XML PADA PERKHIDMATAN WEB BERASASKAN SOAP MENGGUNAKAN ALATAN PERISIAN TENGA

ABBAS AHMED ALI QASEEM AL-ASRI

DISERTASI YANG DIKEMUKAKAN UNTUK MEMENUHI SEBAHAGIAN DARIPADA SYARAT MEMPEROLEH IJAZAH SARJANA SAINS KOMPUTER

FAKULTI TEKNOLOGI DAN SAINS MAKLUMAT
UNIVERSITI KEBANGSAAN MALAYSIA
BANGI

2018

## DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged.

07 July 2018

ABBAS AHMED ALI
QASSEM AL-ASRI
P83728

## ACKNOWLEDGEMENT

First and foremost, praise be to Almighty Allah for all his blessings for giving me patience and strength throughout the duration of this master research.

I would like to express my sincere gratitude to my supervisor Dr. Rossilawati Sulaiman for her guidance and support throughout this work. She has been a great source of inspiration to me. No word can express how grateful I am to her.

Moreover, I am grateful to my parents, my wife and to my brothers that have always been extremely supportive to me. All I can say is that it would take another thesis to express my deep feel affection for my family. Their prayers, patience, sacrifice, love and encouragement have upheld my conviction towards pursuing this course.

Last but not least, to all my dear friends who are directly and indirectly contributing to the success of this work, thank you for everything Let me also say thanks to all the members (lectures and staff) at the Faculty of Information Science and Technology in the National University of Malaysia for their valuable support and warm friendship.

## ABSTRACT

A web service is defined as a method of communication between web applications and clients. Web services are very flexible and scalable as they are independent of both hardware and software infrastructure. The lack of security protection offered by web services creates a gap that attackers can make use of. Web services are offered on the HyperText Transfer Protocol (HTTP) with Simple Object Access Protocol (SOAP) as an underlying infrastructure. Web services rely heavily on Extended Markup Language (XML). Hence, web services are most vulnerable to attacks that use XML as an attack parameter. Recently, a new kind of XML-based Denial-of-Service (XDoS) attacks has surfaced, which target web services. The purpose of these attacks is to consume the system resources by sending SOAP requests that contain malicious XML content. Unfortunately, these malicious requests go undetected underneath the network or transportation layers of the Transfer Control Protocol/Internet Protocol (TCP/IP), as they appear to be legitimate packets. In general, an XML parser is required for the web service engine to extract the required parameters from an incoming message. An attacker can exploit this parser to successfully perform DoS attacks. There are many different techniques that can be used to perform DoS attacks using XML-based message formats. In this research, a middleware tool is proposed to provide real time detection and prevention of XML-based DoS (XDoS) and flooding attacks in web service. This middleware tool focuses on attacks on the two layers in the Open System Interconnection (OSI) model, which are to detect and prevent XDoS attacks on the application layer and prevent flooding attacks at the Network layer. The rule-based approach is used to classify requests either to normal or malicious to detect XDoS attacks. Experimental results from the middleware tool have demonstrated that the rule-based technique has efficiently detected and prevented XDoS and Flooding attacks such as oversized payload, coercive parsing and XML external entities close to real-time such as 0.006 second over the web services. The middleware tool provides close to 100% service availability to normal request, hence protecting the web service against XDoS and distributed XDoS (DXDoS) attacks.

**ABSTRAK**

Perkhidmatan web adalah salah satu mekanisme komunikasi antara aplikasi web. Perkhidmatan web tidak terikat dengan jenis infrastruktur perkakasan dan perisian, kerana ia bersifat fleksibel dan berskala. Kekurangan ciri keselamatan yang disediakan oleh perkhidmatan web mewujudkan peluang bagi penyerang. Perkhidmatan web boleh berfungsi dengan menggunakan protocol-protokol dasar iaitu *Hypertext Transfer Protocol* (HTTP) dan *Simple Object Access Protocol* (SOAP). Perkhidmatan web sangat bergantung kepada *Extended Markup Language* (XML). Oleh itu, perkhidmatan web adalah lemah terhadap serangan yang menggunakan XML sebagai parameter serangan. Kebelakangan ini, sejenis serangan baru Denial-of-Service (DoS) berasaskan XML telah dikenalpasti yang menyasarkan perkhidmatan web. Serangan ini bertujuan untuk menyusutkan sumber komputer mangsa dengan menghantar permintaan SOAP yang mengandungi kandungan XML yang berniat jahat. Permintaan ini tidak dapat dikesan pada lapisan rangkaian atau pengangkutan dalam lapisan *Transfer Control Protocol/Internet Protocol* (TCP/IP), kerana ia dilihat sebagai paket yang sah. Secara umum, penghurai XML diperlukan untuk enjin perkhidmatan web bagi menapis parameter yang diperlukan dari mesej yang diterima. Penyerang boleh mengeksploitasi penghurai ini untuk melaksanakan serangan DoS. Terdapat banyak teknik yang boleh digunakan untuk melakukan serangan DoS dengan format mesej berasaskan XML. Dalam kajian ini, alat perisian tengah dicadangkan untuk menyediakan pengesanan dan pencegahan secara masa nyata, ke atas serangan DoS berasaskan XML (XDoS) dan serangan *flooding* dalam perkhidmatan web. Alat ini memfokuskan kepada serangan ke atas dua lapisan dalam model *Open System Interconnection* (OSI), yang akan mengesan dan mencegah serangan DoS pada lapisan Aplikasi, dan mencegah serangan *flooding* di lapisan Rangkaian. Pendekatan berasaskan petua digunakan untuk mengelas pertanyaan kepada pertanyaan normal atau pertanyaan berniat jahat bagi mengesan serangan XDoS. Hasil eksperimen dari alat ini menunjukkan bahawa pendekatan berasaskan petua dapat mengesan dan mencegah serangan XDoS berasaskan XML seperti muatan besar, penghurai pemaksa dan entiti luar XML yang menghampiri masa nyata ke atas perkhidmatan web. Dengan mengaplikasi alat ini, perkhidmatan web dapat menawarkan hampir 100% layanannya kepada pertanyaan berjenis normal, dan pada masa yang sama terlindung dari serangan XDoS dan serangan XDoS teragih (DXDoS).

**TABLE OF CONTENTS**

# LIST OF TABLES

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

| | |
|---|---|
| B2B | Business to Business |
| BEEP | Blocks Extensible Exchange Protocol |
| CPU | Central Processing Unit |
| CSQD | Cloud Service Queuing Defender |
| DDoS | Distributed Denial of Service |
| DOM | Document Object Model |
| DoS | Denial of Service |
| DPM | Deterministic Packet Marking |
| DTD | Document Type Definition |
| DXDOS | Distributed XML-Based Denial of Service |
| FAP | Fuzzy Associative Pattern-Based |
| FAR | Fuzzy Association Rule-Based |
| HTTP | Hypertext Transfer Protocol |
| H-DoS | HTTP Flooding Attack |
| HX-DoS | HTTP and XML DoS |
| IDP | intrusion Detection and Prevention |
| IDS | Intrusion Detection System |
| IPS | Intrusion Prevention Systems |
| OASIS | Organization for the Advancement of Structured Information Standards |
| QoS | Quality of Service |
| REST | Representational State Transfer |
| SAX | Simple API for XML |
| SOA | Service Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SOTA | Service-Oriented Traceback Architecture |
| SOTM | Service-Ooriented Traceback Mark |

| | |
|---|---|
| SQL | Structured Query Language |
| SSL | Secure Socket Layer |
| UDDI | Universal Description, Discovery and Integration |
| URL | Uniform Resource Locator |
| W3C | World Wide Web Consortium |
| WS | Web Services |
| WSDL | Web Service Description Language |
| WS-RX | Web Services Reliable Exchange |
| XDoS | XML-Based Denial of Service |
| XML | Extensible Markup Language |
| XSS | Cross Site Scripting |
| XXS | XML External Entity |

# CHAPTER I

# INTRODUCTION

## 1.1 INTRODUCTION

The dynamic use of the Internet services and applications has led to an increase on conducting businesses online. It is considered that among the most innovative inventions recently is the web technologies and Service-Oriented Architectures (SOA). From early 1990s up till now, Web Services (WS) have evolved and come along the way by offering rich functionalities, user friendly, ease of use and its integration with other applications and services irrespective of the different platforms. This trend has fascinated governments, banks and large organizations to offer their services via web applications (Tiwari & Singh 2011). Web application services include e-commerce portals, cloud computing, search engines (such as Google and Microsoft Bing), social networking and many more (Jaffe 2014).

WS uses standard protocols (XML and SOAP) for the communication. Hence WS is considered as a cross-platform that allow applications to connect to each other independent of platform and/or language (Java can talk with C#, Windows applications can talk with Unix applications) (Mouli & Jevitha 2016). WS usually is made interoperable using the SOA, where it consists set of concepts and techniques designing and developing software (Altmeier et al. 2015).

The web service can be accessed through an Internet connection in a self-contained and self-describing application. Web services are represented by Web Service Description Language (WSDL). However, metadata can be used as an initiator for the Universal Description, Discovery and Integration (UDDI). Usually web service applications are provided for a specific business function which can be re-

used within different settings. Simple Object Access Protocol (SOAP) is an eXtensible Markup Language (XML) standard used to exchange messages between Web services (Falkenberg et al. 2013).

Several vulnerabilities exist in messaging protocols such as SOAP and XML. They are exposed to several type of attacks, for example, XML injections, Denial of Service (DoS) attacks, Cross Site Scripting (XSS) attacks, Structured Query Language (SQL) injections, SOAP oversized payloads and buffer oversized exploits (Chana et al. 2015).

Based on the Open Web Application Security Project (OWASP 2017), XML, SQL injection, and DoS attacks are the most frequently occurred in Web and WS applications. They are classified as the top attacks since the year 2017. Moreover, using web services over Hypertext Transfer Protocol (HTTP) makes it difficult to block malicious web service traffic by firewalls, as firewalls cannot check the XML content for any suspicious packets (Bebawy et al. 2005; Rajaram & Babu 2013). Therefore, it is necessary to provide a security solution for SOAP message communication in the application layer itself in order to provide an end-to-end integrity, availability and confidentiality (Sindhu & Kanchana 2014).

According to Tianfield (2012), the availability of system resources is one of the difficult issues, as the server are assumed to provide services to customers continuously. Recently, several security standards are established to provide protection for web services, but they only address integrity and confidentiality aspects of web service security (Sindhu & Kanchana 2014). The lack of availability protection made it easy for attackers to launch flooding attacks that overwhelm servers and prevent them from delivering service to legitimate customers.

The DoS attack purpose is to impede availability of web services via making computer resources (such as network bandwidth, CPU time, etc.) totally overloaded and the services running on the server are unavailable to its intended users. An attacker usually uses a huge number of computers to trigger the Distributed DoS (DDoS) attacks. DDoS techniques are similar to DOS attacks, but they are launched

from multiple connected devices distributed across the Internet (Murugan & Vivekanandan 2015).

The availability of web services is an important factor for business continuity. Therefore, web services DDoS attacks introduce a challenging risk. DDoS attacks consumes a lot of the computational resources, such as CPU or memory, makes the system unavailable to legitimate users (Altmeier et al. 2015).

Usually, web service applications are open and available world-wide, which means that web servers deal with all types of users including (normal users and attackers) (Lin et al. 2008). Some have ill intention by taking advantage of insecurity of web servers to initiate flooding attacks to compromise availability of web services. As a result, legitimate users' requests will be blocked (Suriadi et al. 2011). According to Jensen et al. (2009) DoS attacks can be triggered with less efforts than those attacks targeting non-web service systems including (web applications, web sites that do not use web service). In web service application, DoS attack can be launched by a small malformed XML message (Masdari & Jalali 2016). As recently published by Akamai's Third Quarter 2017 Report on State of the Internet, showed that there has been a 69% increase in DDoS attacks over a year, with most attacks targeting web service applications related to financial organizations, high tech organizations, public sector agencies, media and entertainment organizations (Akamai's 2017).

Furthermore, CDNetworks (2017) stated that, flooding attacks have increased compared with the same quarter of 2016, where these attacks costed organizations significant losses in revenues and even customers. When dealing with flooding attacks on web servers the key endeavor is the web services architecture nature. Moreover, web services are characterized by rapid distribution of application components, which are deployed in different web servers and are offered by different service providers globally. These characteristics make it very challenging to set standard strategies in detecting and protecting against DoS attacks (Tiwari & Singh 2011).

So far, many proposals have been presented by security researchers demonstrating various mechanisms in defending against flooding attacks. Most of

these mechanisms help in lowering attacks, but still considered difficult to tackle flooding attacks because up till now there is no standard solution in addressing this issue (Suriadi et al. 2011).

In general, a set of computers are controlled remotely by the attacker in a DDoS attack. This control is done by using malicious software installed on each computer. Each of these computers is called "zombie". Zombies are controlled by an attacker to launch a DDoS attack against the victim's systems (Sandeep 2014). DDoS attack process can be initiated into two stages; the first stage is to compromise targeted systems which can be accessible through an Internet connection, and then the attacker installs some hacking tools into these compromised systems, which turns those systems into zombies. In the second stage, the attacker can control the compromised system by sending an attack command to the zombies using some sort of a secure channel to launch a bandwidth attack against the targeted victim. Figure 1.1 shows a generic scheme of a DDoS attack using a single attacker.

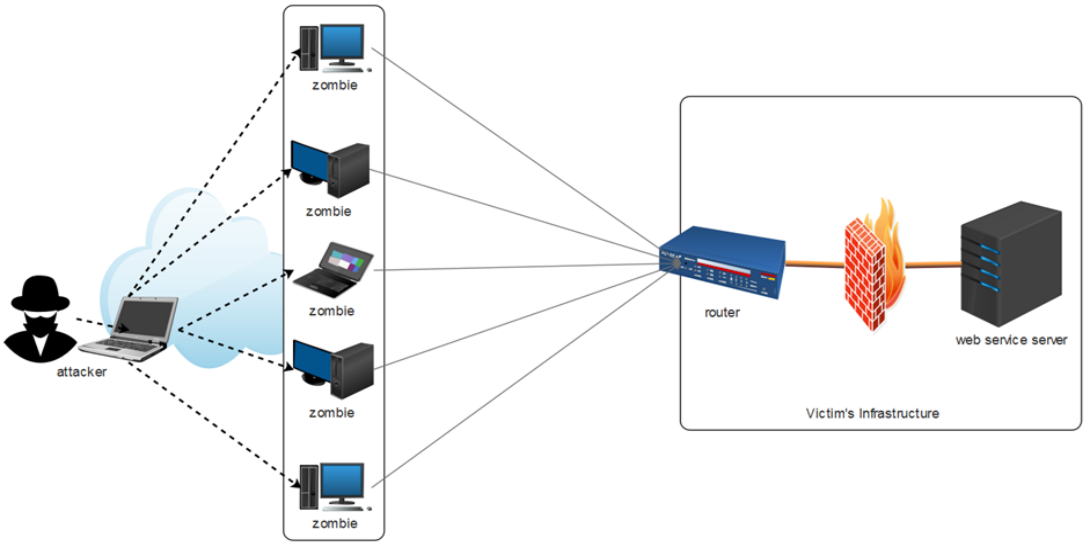


Figure 1.1 DDoS attack Architecture

The DDoS attack is one of the most famous and frequent web service attacks (Mouli & Jevitha 2016). In this research, a solution is proposed to prevent XDoS and flooding attacks on web services. Since DDoS was derived from DoS techniques, the word DoS attacks is also referred to both DoS as well as DDoS attacks, in the context of the thesis.

## 1.2 PROBLEM STATEMENT

Nowadays, web services are widely use due to their independent nature and code reusability. In a DoS attack, the attacker tries to exploit the web service to make it unreachable and cannot serve the legitimate users, then the system will be unavailable and crash at certain point of time due to resource overloading (Loganathan & Ramesh 2015).

Most of web services today use XML for transferring data between web service providers and consumers. However, many of the XML vulnerabilities have been recently discovered and reported (Jensen et al. 2009; Gupta & Thilagam 2013; Jan et al. 2016), as these vulnerabilities provided chances of DoS attacks. Therefore, many systems that depend on XML protocol are at risk if vulnerabilities are not properly mitigated (Jan et al. 2015).

An XML parser is required for the web service engine to extract the required parameters from an incoming message. An attacker can exploit this parser to successfully establish DoS attacks (Gupta & Thilagam 2013). The parser behavior can be influenced by adding a Document Type Definition (DTD) on top of the XML document. It is usually developed to specify the format or syntax of an XML document, which also may produce several attacks, such as DoS (Späth et al. 2016).

There are many different techniques used to perform DoS attacks using XML-based message formats. Given the complexity nature of an XML document and the resource consumption nature of the XML parsing, a small distortion in a SOAP message may cost a large amount of resources (Falkenberg et al. 2013).

XML is considered to be the fundamental technology of web services which provides web services with many advantages but at the same time cause many problems in the security point of view (Ibrahim & Shanavas 2014).

Because of the huge number of requests sent to the victim web servers, DoS attacks can degrade or completely disrupt web services. Flooding attacks are one of

the biggest concerns for security professionals (Baria 2014). As a result, several security approaches have been used to resolve this problem. However, DoS attacks are still considered among the least preventable attacks (Singh et al. 2017). In this research, a solution is proposed to overcome and specifically prevent the XML-based DoS (XDoS) and flooding attacks efficiently. These two attacks are chosen in this research because they are commonly implemented by attackers and they are very hard to detect. Since the attacks take place on the application layer, so each attack appears to be legitimate (Jan et al. 2015).

## 1.3 RESEARCH QUESTIONS

This research is conducted to answer questions related to the problem statement, such as the following:

RQ1: What are the current techniques proposed to overcome the XDoS and Flood attacks in SOAP-Based web services?

RQ2: What are the requirements and important parameter to overcome XDoS and Flooding attacks?

RQ3: Is a middleware tool able to efficiently protect the web services against XDoS and Flood attacks?

## 1.4 RESEARCH OBJECTIVES

This study aims to investigate web service attacks, especially XDoS and flooding attacks. On the other hand, this research will explore existing techniques proposed by other researchers in detecting and defending web service from DoS & DDoS attacks. As a contribution of this study, the researcher will develop a middleware tool to prevent XDoS and flood attacks on web services. Hence, we can formulate the research objectives in three main tasks:

1. To investigate existing methods used to detect and defend against XDoS and flood attacks on web service applications.

2. To design and develop a new mechanism (in a form of a tool) which will be used to prevent XDoS and flood attacks in web service applications.

3. To evaluate the efficiency of the proposed method by comparing the server response time for the selected SOAP requests.

## 1.5 RESEARCH SCOPE

There are different types of XDoS techniques that were suggested by previous researchers, but the real challenge in detecting such attacks is to validate whether the checked XML parser can be exploited by such techniques. It is indeed a complicated task due to the nature of this type of attacks where it could vary from time to time. For example, placing the payload of the DoS attack at one location inside the XML document may confuse the parser and make the DoS attack successful (Altmeier et al. 2015).

The scope of this work is to improve web service security by preventing XDoS and flooding attacks. Therefore, other web services vulnerabilities or web server weaknesses are beyond the scope of this research. The tool that will be developed is to demonstrate how XDoS, as well as flooding attacks can be detected and blocked.

## 1.6 MOTIVATION

One of the essential principles of the Service Oriented Architecture (SOA) technology is to develop a system comprising of devices and machines connected through web services. A service is the key component of SOA foundation. SOAP is a typical web service technology to establish SOA (Kaur et al. 2017). Web services based on SOAP are developed on top of the independent markup language XML platform. They are usually deployed in business to business (B2B) integrations and are supported by the industry large vendors such as IBM and Axway (Altmeier et al. 2015).

Web services availability in SOA implementations is of a vital requirement. Hence, DoS attacks introduces a significant risk on web services, as they attempt to consume a huge amount of computational resources such as Central Processing Unit **(**CPU) or memory, with the aim to make web service unavailable. Moreover, several approaches and techniques are used to perform DoS attacks, which lead to a motivation to apply a new mechanism that prevents this type of attack.

## 1.7 SIGNIFICANCE OF THE RESEARCH

In existing research to prevent web service attacks, the researchers only addressed SOAP messages security such as confidentiality, integrity and non-repudiation. There are few security mechanisms provided for XDoS, where most of the current techniques do not provide a complete solution to prevent such attacks. These solutions are limited in preventing and containing this attack because they are mainly focus on the application layer of the OSI model to detect and prevent these attacks. These solutions can be considered as not enough, because the attacker could use the flooding attack (flooding the server with thousands of requests per second), which causes the service to be shut down due to server's resources consumption and most of these approaches do not provide a deep XML analysis, which is to find more detail features to classify packets. Hence, a middleware-based solution to overcome these attacks is proposed and developed in this work.

## 1.8 THESIS ORGANIZATION

The thesis is being organised into five chapters which are elaborated as follow:

**Chapter I** depicts the outline of the study where an introduction the research is described, the problem statement is formulated, the research objectives are developed, and the scope of the research is identified.

**Chapter II** provides a comprehensive literature review on the field of XML-based DoS attacks in web services. This can be represented by identifying the concept of web service, SOA and XML. Then this chapter describes the structure of web service

and the standards that would are related with web service. In addition, this chapter explores the existing techniques to detect the XDos attacks.

**Chapter III** illustrates in detail the research methodology, beginning from identifying the problem to achieving the objectives of this study. Hence, this chapter highlights the research methodology by illustrating the main components of the method. These components consist of three aspects; client side, middleware side and server side.

**Chapter IV** highlights the experimental results obtained by the proposed method. This can be represented by describing the experiment setting in which the parameters of the experiment are being illustrated including tools, computers, and programming language used. In addition, the evaluation method that would be carried out to assess the proposed method is also described.

**Chapter V** highlights the conclusion of the study where a summary of the whole thesis is illustrated. Moreover, an implication of the research contribution is also elaborated. Finally, the future directions that could be motivated by this study are discussed.

## 1.9 CHAPTER SUMMARY

This chapter concentrates on the core of the study in which the problem, research questions, objectives, scope, motivation and significance of the research were determined and illustrated. Next chapter will discuss the literature review behind the study.

# CHAPTER II

# LITERATURE REVIEW

## 2.1 INTRODUCTION

This chapter aims to review numerous collection of literature with respect to DoS attacks addressing web service applications, especially XML syntax, SOAP message and flooding attacks. The chapter is divided into five main sections; the first section provides the relevant standards and technologies in the research subject. This include the Extensible Markup Language (XML), XML schema, XML parsing techniques, Service Oriented Architecture (SOA), along with concepts of web services (WS) including web service components and web service architecture. The second section addresses web services attacks and demonstrates in depth analysis of Web Service Security standards (WS-Security) which usually used to provide integrity and Confidentiality of SOAP messages. Moreover, a taxonomy of DoS attacks is analysed and described, with detailed explanation on DDoS and its variants. Several mechanisms of attack that attackers use to exploit the availability of resources are also explained in the third section. The fourth section discusses the detection techniques used to detect DoS attacks. The fifth section focuses on previous works used to detect and defend DoS attacks, where it explores and reviews the techniques that are already investigated by the cyber security community in the subject matter. This part will focus on the techniques used in detecting DoS attacks as well as techniques to defend against DoS attacks in web service applications.

## 2.2 EXISTING STANDARDS AND TECHNOLOGIES

This section describes a brief introduction to the relevant standards and technologies used in this research.

### 2.2.1 XML and XML Schema

Extensible Markup Language (XML) is a structured format developed by the World Wide Web Consortium (W3C), in which transmission, validation and interpretation of data is being set (Bray 2008). Data interpretation can be implemented independently in sort of software and hardware; thus, XML is considered to be more suitable for data exchange between different applications and organizations. XML structure of a document is defined by XML elements, where an XML element typically uses a start tag <*tag*> and an end tag </*tag*>. Other child elements can be included such as element attributes and text contents.

Figure 2.1 shows an example of XML file, which stores an order in a bookshop.

```
<?xml version=" 1.0 " encoding="ISO−8859−1" ?>
    <order>
        <customerID>1234</customerID>
        <ISBN>1234567234−3</ISBN>
        <Price>10.00 </Price>
    </order>
```

Figure 2.1 An example of XML file

As described by W3C, XML Schema is a component describing the structure of an XML document (Shudi (Sandy) Gao 2012). It is essentially a set of rules that describe the structure for each element in the container. It covers its allowed attributes (e.g., a customerID, ISBN and Price), the type of its value (e.g., a string or integer), a description of its allowed child elements (e.g., an order).

### 2.2.2 XML Parsing

The first portion of a web service is the XML parsers which process input from other web services or applications. The web service can be compromised due to a poorly designed or poorly configured XML parser regardless of how secure the web service is. So, it is vital to use robust and proven XML parsers (Singhal et al. 2007). An improperly configured or developed XML Parser is susceptible to many different attacks, such as the following:

1. Large or recursive XML documents can overload the XML parser and lead to a DoS. In the recursive attack, the parser receives an XML document which declares two entities calling each other in an infinite loop (Späth et al. 2016).

```
<!DOCTYPE data [
<!ENTITY a "&b;">
<!ENTITY b "&a;"> ]>
<data>&a;</data>
```

Figure 2.2 An example of XML Infinite Recursion

In Figure 2.2, the parser resolves the entity `a` to a reference of `b` and the entity `b` resolves to a reference of `a`. Therefore, the parser will loop indefinitely and consume CPU resources.

2. XML documents can be configured to refer to and use local files. This may help an attacker to gain knowledge about the local system or lead to DoS attacks.

```
<!DOCTYPE myFile [
  <!ELEMENT myFile ANY >
  <!ENTITY File SYSTEM "\\192.168.0.2\payload.txt">
]>
<myFile>&File;</myFile>
```

Figure 2.3 An example of XML external entity attack

In Figure 2.3 the parser will replace the external entity '&File;' with the content of the remote file '\\192.168.0.2\payload.txt', which is a large size file.

3. External references to other XML documents or XML schemas can be used to bypass XML validators such as file size limit. As shown in Figure 2.3, the attacker can use that weakness to load XML file to bypass XML validator.

There are two standard types of XML parsers which can be used across different platforms, which are DOM-based and SAX parser, explained like the following:

**a. DOM-based parsers**

In this approach, a tree structure of the whole XML document is created and can be called as document object model (DOM). A programmer can access the XML document and easily traverse through the XML tree, access, insert, and delete nodes. A node may be an element, an attribute, a text content, or a comment. The main disadvantage of this parsing approach is the high resource utilization. Because the whole document must be read and load into memory before the programmer can access its contents (Falkenberg et al. 2013).

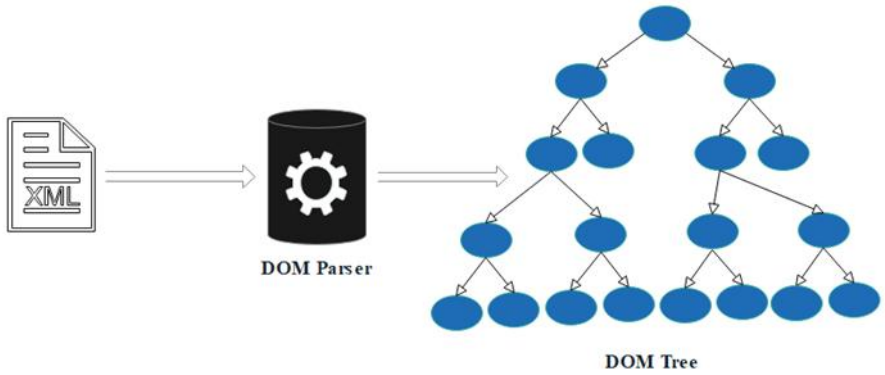


Figure 2.4 DOM parser

As shown in Figure 2.4, a DOM parser loads the complete XML file into memory and creates a tree structure where each node in the tree represents a component of XML file.

**b. SAX parsers**

Simple API for XML (SAX) parsers are event-based. XML tree can be used for SAX parser to operate once at a time on parts of the XML tree, so that whenever the parser encounters an XML node, an event is triggered. In order to process the XML document, the program's task is responsible to handle these events. Low utilization of resources is one of the main advantages of event-based parsing. Furthermore, event handlers can immediately start processing the data once an event is triggered without having to wait for the parser till the end of the XML document. However, some operations will be more complex compared to a DOM parser, for example, multiple parsers are required to sort an XML document, as the SAX parser only reads forward sequentially (Falkenberg et al. 2013).

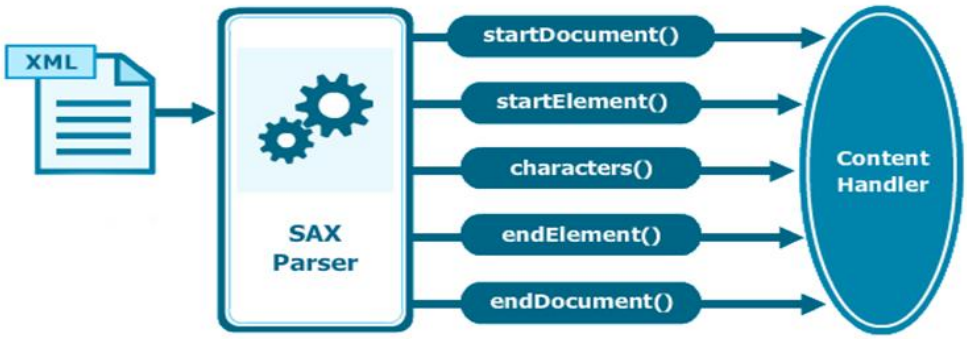


Figure 2.5 SAX parser

As shown in Figure 2.5, the SAX Parser parses the XML file line by line and triggers events when it encounters opening tags, closing tags or character data in an XML file.

### 2.2.3 Service Oriented Architectures

Service Oriented Architectures (SOA) is a famous architectural model that provides roughly composition of services. In the last decade, Service Oriented Architectures and Web Services were considered to be the most innovative technological advancement. When building web service applications SOA is currently considered as a blueprint (Simmonds et al. 2010). The architecture's main objective is to gain minimum dependency between software agents. It also supports reusability, composability and interoperability between different applications (Masood 2013). The main goals for the transition from the traditional approach (such as object-based approach) to service-oriented approach are mainly facilitated by SOAs consideration of real world factors such as trust boundaries, physical distribution and performance requirements. Moreover, SOAs boundaries are explicit, where services are autonomous with minimal dependency on interacting software modules. In addition, SOA uses standard internet technologies such as HTTP and XML to build distributed systems.

Distributed systems are usually defined as systems that consist of more than one independent computer, but they appear as one system to the user (Tanenbaum & Van Steen 2007). The design principle behind SOA is the use of loose coupling which isolate each service as an independent entity so as to provide a layer of abstraction between service providers and consumers. This promotes flexibility during implementation of different services without impacting consumers (Serrano et al. 2014).

Moreover, one of the primary goals of SOA is to automate business processes by allowing services to automatically discover one another, and immediately take advantage of the functionality offered (Singhal et al. 2007). The communication process starts with the service provider publishing a service description in WSDL document. The description includes information on what technologies a service provider supports. When a consumer needs a particular service, he has to retrieve the relevant WSDL document from web service provider and then start to invoke operations of the intended service (Suriadi et al. 2011). There are other ways to

retrieve WSDL file which include the use of UDDI platform, as the consumer can search the registry for the intended WSDL file to start requesting Web services (Shahgholi et al. 2011). Figure 2.6 illustrates web service discovery and invocation in SOAs.

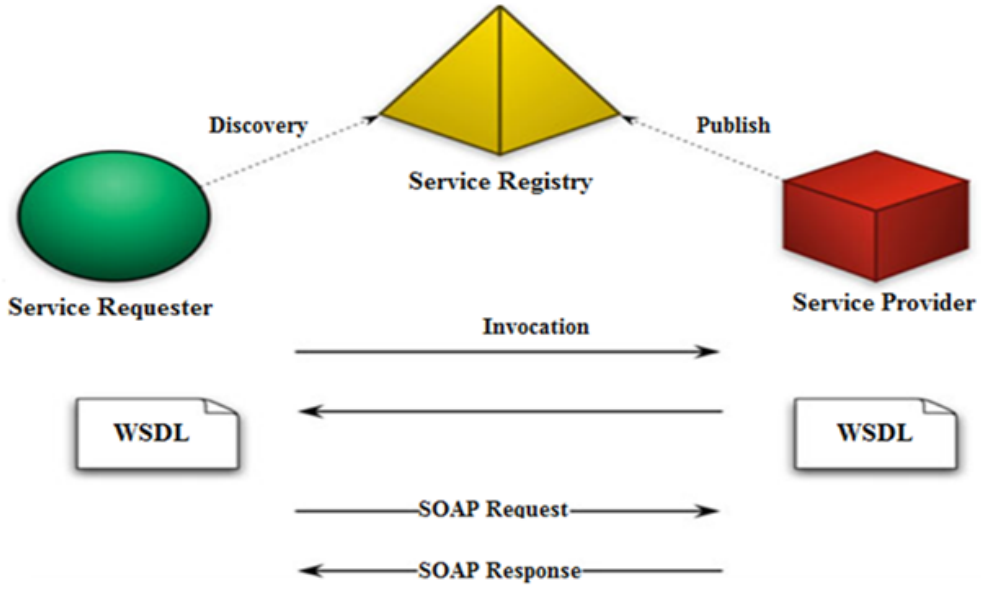


Figure 2.6 Web service discovery and invocation

As shown in figure 2.6, a web service cannot be discovered if it has not been published, because the web service discovery depends on service publication. Many mechanisms allow the service requestor to gain access to the service description (WSDL) and make it available to the application at runtime by invoking the WSDL file. A simple way is to use static discovery where the service requestor retrieves a WSDL document from a local file. This is usually the WSDL document obtained through a direct publish. Alternatively, the service may be discovered at design time or run time using a local WSDL registry, or a public or private registry such as UDDI (Kreger 2001).

### 2.2.4 Web Services

A web service can be described as an interface with a collection of operations that are network-accessible through standardized XML messaging. It can be described using a standard, formal XML notion as its service description. It can cover all necessary details to interact with the service, including message formats (that describes the operations), transport protocols and location (Kreger 2001).

Moreover, a web service is a technique to interact between processes over computer networks between different software applications that might work as an individual or combined entity with different services to compose a group of services (Altmeier et al. 2015). A web service is a platform which is language independent, and it could run on any platform and can be written in any programming language (Sindhu & Kanchana 2014). It can be implemented using different technologies, for example, Representational State Transfer (REST) or SOAP (Kaur et al. 2017; Rathod 2017).

SOAP is a W3C specification that defines the structure of XML messages and also a protocol to achieve machine-to-machine communication, and generally SOAP messages consist of a header and a body. The <*Header*> element includes message-specific data (e.g. timestamp, user information, or security tokens). The <*Body*> element contains function invocation data.

Web services achieve interoperability by using a set of XML-based open standards, such as WSDL, SOAP and UDDI. These standards provide a common blueprint of defining, identifying, locating, publishing and consuming web services. In addition, web services use SOAP to transfer the data, and WSDL is used for describing the services, whereas UDDI is used to access the services using service metadata (Ibrahim & Shanavas 2014).

### 2.2.5 Web Services Architecture

To view the web service architecture, there are two ways (Kreger 2001). The first one is to examine the individual roles of each web service actor, and the second one is to examine the emerging web service protocol stack.

**1. Roles in a Web Services Architecture**

Figure 2.7 shows a web services architecture base on roles. There are three major roles within the web service architecture.

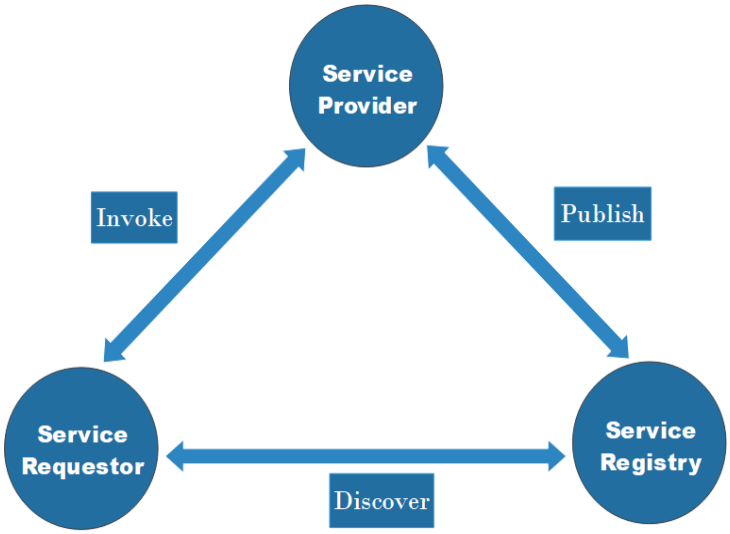


Figure 2.7 Web services architecture base on roles

a. Service Provider

It is the entity who provides the web service and implements the service and makes it available on the Internet.

b. Service Requestor

It can be represented by any consumer of the web service. The service requestor utilizes an existing web service by opening a network session, then sending an XML request.