# A GENERIC ARCHITECTURE FOR WEB APPLICATIONS TO SUPPORT THREAT ANALYSIS OF INFRASTRUCTURAL COMPONENTS

Lieven Desmet, Bart Jacobs, Frank Piessens, and Wouter Joosen
*DistriNet Research Group, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium*

Abstract: In order to perform a useful threat analysis of a web application platform, some architectural assumptions about such applications must be made. This document describes a generic architecture for typical 3-tier web applications. It serves as the basis for analyzing the threats in the most important infrastructural components in that architecture, presented in the following papers.

Key words: Web applications, architectural overview, 3-tier model

## 1. MOTIVATION

Web applications are an interesting target for security attacks on the Internet. They are easily accessible through the HTTP-protocol, and often company-critical assets are part of the web application infrastructure. Moreover, while web applications infrastructures are fairly complex, basic technology for building web applications is easily accessible. Hence, web applications are often designed and built by developers with little or no distributed system security background. Therefore, useable guidelines for building secure web applications are highly useful.

This document describes a generic architecture of modern web applications, as commonly used in practice by Independent Software Vendors. Hereby, the most commonly used components within the

infrastructure and their interactions are presented. The goal of this document is to define a common architectural view for web applications, which can be used to conduct a thorough threat analysis for each of the infrastructural components. Such an analysis will be presented in the following papers ([6. 7,8,9,10]), and will serve as the basis for a set of guidelines to support Independent Software Vendors in building secure web applications.

## 2. WEB APPLICATIONS

In the early days of the World Wide Web, web servers offered static content to end users visiting the website with a browser. But today, static web servers are more and more replaced by *web applications:* dynamic websites that use the browser as a user interface to a server-resident application. Typical examples of such web applications are e-commerce sites, or front-ends to business processes, databases or existing legacy systems.

A variety of technologies for building web applications exists today. Older technologies such as CGI (common gateway interface) provided a simple standardized interface between a web server and an existing application. The application was started on the web server for every dynamic request in order to process the request, introducing a big startup and shutdown overhead. In newer technologies such as Java Servlets, JSP and ASP.NET, dynamic requests are handled by components that can be plugged into the web server. The real processing work can be delegated to a separate application server, leading to better performance and manageability. Moreover, the application server can offer support for non-functional requirements of the application such as transactional behavior, synchronization, access control and so forth.

Because of these advantages and the widespread adoption of application servers in building complex web applications, only this last technology is considered in this document.

Web applications are distributed applications [11], using the HTTP transport protocol. The system architecture is a client-server model. Both the client (e.g. a rich client) and the server can take part in the processing of information, known as client side and server side processing.

In this paper, we distinguish between web *applications* and web *services*. Web services expose functionality through an XML-based messaging protocol (most often the SOAP protocol [13]), and are very often run on top of HTTP. Whereas web applications are intended to be used by end-users through a standard browser, web services are intended to support machine-

to-machine communication over the Internet. Web services can be an important infrastructural component for building web applications.

## 3. ARCHITECTURAL OVERVIEW

Our generic web application architecture consists of a client at the end-user side, and a 3-tier processing server side (presentation tier, business tier and back-office tier) as shown in Figure 1. Firewalls can be placed between each tier to enable network perimeter security. Often, this architecture is simplified by omitting FW3 and/or FW2, and by implementing two or more tiers on the same machine. Also, in some deployments the authentication server is directly connected to the web server.

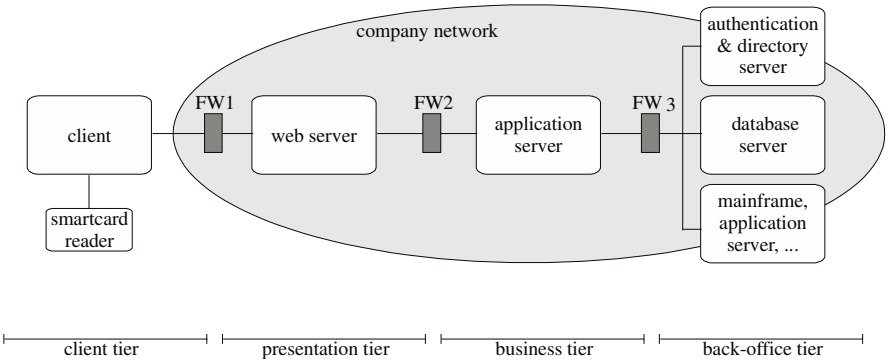


*Figure 1.* Architectural overview

### 3.1 Client tier:

Basically, the client tier consists of a recent web browser, possibly extended with client-side application components downloaded from the web server (such as Java Applets or .Net assemblies). In the latter case, the client is referred to as a rich client within this document. The client tier interacts with the web server through simple HTML over HTTP, or, in case of a rich client, the client can act as a web service entity and use SOAP over HTTP interactions with the web server.

Furthermore, the client can be equipped with a smart card reader to interface with a smart card or other security token. Such tokens can be used among others for authenticating the user and protecting the requests.

## 3.2 Presentation tier:

The presentation tier is responsible for formatting the processed information before returning it to the client, and for handling client requests by performing input validation and delegating them to the appropriate units within the business tier. Usually, the processed information is formatted using the markup languages HTML (in case of a client browser) or XML (in case of a rich client).

Infrastructural components within the presentation tier are typically the web server, sometimes accompanied by a web connector of the application server.

## 3.3 Business tier:

The business tier contains the application server. The application server implements the actual business logic. In order to achieve its functionality, several services can be provided to the application server from the back-office tier.

The web server from the presentation tier can interact with the application server by using remote procedure calls, web services or a proprietary application server protocol.

## 3.4 Back-office tier:

The back-office tier provides some basic services to the business tier, such as a database system and an authentication and directory service. The SQL query language is mostly used in requests towards the database system, and LDAP [14] in communication with the directory service. The communication protocol for the authentication service depends on the authentication system used (e.g. Kerberos [12]).

The back-office tier can also contain back-end systems including mainframes, wrapped legacy applications and interfaces to remote application servers.

The architecture presented here does not include some of the more advanced features of web applications, such as the dynamic discovery of services, business integration and associated trust relationships. These features are out of scope for this document, as this is typically not an Independent Software Vendor task.

# 4. A SIMPLE EXAMPLE

In this section, our architecture for web applications is mapped to actual technologies on Microsoft platforms. Each entity is assumed to be equipped with a recent version of Windows (for instance Windows XP on the client, and Windows Server 2003 on the servers). An architectural overview is illustrated in Figure 2.

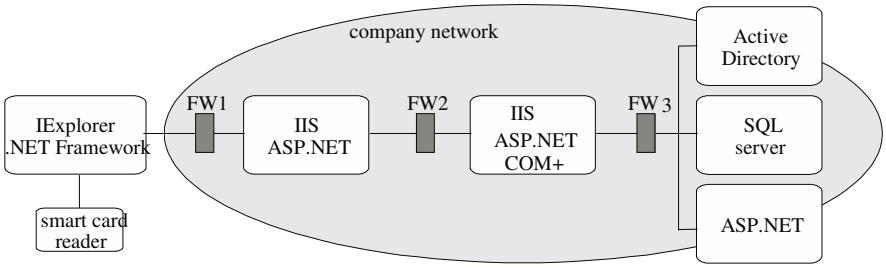


*Figure 2.* Architectural overview instantiated with Microsoft technologies

The client system has a recent version Internet Explorer and the .NET framework. The client interacts with the web server using simple HTML over HTTP or SOAP over HTTP. In the latter case, the client downloads and executes .NET assemblies within the browser. The client can use a smart card reader for authentication and encryption purposes.

The presentation tier hosts a web server running IIS and ASP.NET. The web server interacts with an application server in the business tier, using web services. The application server runs IIS, ASP.NET, COM+, ADSI, Visual Basic etc.

A directory server, a database server and a remote application server are located within the back-office tier. The directory server in a Microsoft environment is typically Active Directory. The business tier uses SQL to interact with the database server, running SQL Server. Connection to a remote application server or wrapped legacy application is done via SOAP.

# 5. ACKNOWLEDGEMENTS

This document is a result of the *Designing Secure Applications* (DeSecA) project, funded by Microsoft. Partners within this project are the Università Degli Studi Di Milano [1], the Technical University of Ilmenau [2], the University of Salford [3], and the COSIC [4] and DistriNet [5] research groups of the Katholieke Universiteit Leuven. Each of these partners

performed a threat analysis on one particular infrastructural component or technology within the presented web architecture, focusing of course on the technologies provided by Microsoft. The results of these studies are reported in the following papers, respectively, for SQL server [6], ASP.NET [7], Active Directory [8], security tokens [9] and web services [10].

## 6. REFERENCES


1. Departemento di Informatica e Comunicazione, Univesità degli studi di Milano, Italy; url: http://www.dico.unimi.it
2. Technical University of Ilmenau, Research Group Multimedia Applications, Germany; url: http://www-ifmk.tu-ilmenau.de/mma
3. Information Systems Security Research Group, University of Salford, UK, url: http://sec.isi.salford.ac.uk/
4. COmputer Security and Industrial Cryptography (COSIC), Department Electrical engineering (ESAT), Katholieke Universiteit Leuven, Belgium; url: http://www.esat.kuleuven.ac.be/cosic/
5. DistriNet Research Group, Department of Computer Science, Katholieke Universiteit Leuven, Belgium url: http://www.cs.kuleuven.ac.be/cwis/research/distrinet/
6. E. Bertino, D. Bruschi, S. Franzoni, I. Nai-Fovino, and S. Valtolina. Threat modelling for SQL Servers. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp189-201
7. R. Grimm and H. Eichstädt. Threat modelling for ASP.NET – Designing Secure Applications. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp175-187
8. D. W. Chadwick. Threat Modelling for Active Directory. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp203-
9. D. De Cock, K. Wouters, D. Schellekens, D. Singelee, and B. Preneel. Threat modelling for security tokens in web applications. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp 213-223
10. L. Desmet, B. Jacobs, F. Piessens, and W. Joosen. Threat modelling for web services based web applications. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp161-174
11. G. F. Coulouris, J. Dollimore and T. Kindberg. Distributed Systems: Concepts and Design, third edition. Addison-Wesley, 2001.
12. J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5), RFC 1510, September 1993, http://www.ietf.org/rfc/rfc1510.txt
13. W3C Note, SOAP: Simple Object Access Protocol 1.1, May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
14. M. Wahl, T. Howes, and S. Kille. Lightweight Directory Access Protocol (v3), RFC 2251, December 1997, http://www.ietf.org/rfc/rfc2251.txt
15. R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile, RFC 2459, January 1999, http://www.ietf.org/rfc/rfc2459.txt