CWASAR: a European Infrastructure for Secure Electronic Commerce

Ciarán Bryce, Winfried Kühnhauser

GMD - German National Research Center for Information Technology,
Institute SET-RS, Schloß Birlinghoven,
53754 Sankt Augustin,
Germany.
(email: Ciaran.Bryce@gmd.de, Winfried.Kuehnhauser@gmd.de)

Rémy Amouroux, Mauricio López

BULL - GIE DYADE, INRIA Rhône-Alpes,
655 avenue de l’Europe,
38330 Montbonnot,
France.
(email: R.Amouroux@frec.bull.fr, M.Lopez@frec.bull.fr)

Harry Rudnik

ECO, Société für Wirtschaftsfragen GmbH,
Kaiserstraße 101,
53113 Bonn,
Germany.

Abstract

This short paper introduces the CWASAR project. The goal of CWASAR is to design and implement a low cost European infrastructure for a secure electronic marketplace. The project was preceded by a market analysis in Germany, Spain and France, of the basic user requirements for a European oriented system. From the results of this analysis, we have defined the main functional and architectural components of CWASAR. The description of these components is the subject of this paper. In particular, we focus on the features specifically designed to cater for the varying security requirements of end users, as well as the problems posed by the differing legal positions held by individual EU states on the use of security techniques. The CWASAR project is sponsored by the European Union.

Key Words: European electronic commerce, security architecture, commerce service model.
Introduction to CWASAR

CWASAR (Cooperative Wide-Area Service Architecture) is a European Union (EU) sponsored project whose aim is to offer an electronic commerce infrastructure to European small and medium sized enterprises. The project was preceded by an extensive market analysis in Germany, Spain and France, on the user requirements for a European-oriented electronic commerce system. The results of this analysis showed three of the main technical requirements to be: 1) reduced network access costs, 2) streamlining of services so that the information they furnish is as relevant as possible, and 3) security. By security, we especially mean the ability of information and service providers to specify and have enforced their own rules governing access to their information.

The first phase of the CWASAR project formally ran from January to December 1996 and covered design and demonstrator development issues. The design aims to satisfy the user requirements mentioned above, and also to include sufficient flexibility so that further feedback from our end-user base through presentations of the user front-end and demonstrator may be taken into account. Our goal in this paper is to overview the CWASAR design. In particular, we discuss the problems posed and solutions adopted to integrating security. Section 1 describes the CWASAR service model and its security features. The service model is the system’s functional specification: it provides a view of the system’s information base and describes how electronic services are built over this base. Such a structuring model is needed to handle the complexity brought by the large amount of information in the system, and thus it has a similar goal to the client-server model for distributed applications. In section 1, we use an example service, the Business Yellow Pages, to help illustrate our ideas. Section 2 describes the technical architecture onto which the CWASAR service model is mapped. Section 3 offers some concluding remarks.

1 The CWASAR Service Model

The main purpose of the CWASAR service model is to define the classes of system data, the basic data handling functions which manipulate this data, the types of users, and the security functions which control use of all data [1][2]. The basic schema of the CWASAR service model is illustrated in figure 1.
1.1 The Classes of System Data

Referring to figure 1, the first class of data in CWASAR is General Data. This is defined as the set of all up-to-date company related information in the system. All general data is stored in a unique logical general database (GDB) and this constitutes the system’s information base. All information needed by CWASAR services is extracted from the GDB. Further, each company that is a participant in the CWASAR system will have its own entries in the GDB.

Source Data is company data which is stored and manipulated outside of the CWASAR system. This data forms part of the service model because data entered into the GDB originates from source data, and so a data acquisition function is needed to support the correctness and security of the transition from source data to general data.

Finally, Domain Data is data which is extracted from the general data for use by a particular electronic commerce service. Domain data is stored in a set of logical domain databases (DDBs). Moreover, the data of a particular domain is structured in a way which best suits the service using that domain. The goal here is that a service’s DDB should be able to satisfy most - we say 80% - of information requests to the service, thus tackling the “streamlining of services” requirement mentioned in the opening paragraph. As we shall explain in the next section, the DDBs are located nearer to the end-user and his service front-end, in the aim of reducing network access costs.

General and Domain data are organized around the notion of Document, corresponding to a top-level object class from which all other information classes are derived [2]. The Document class has been defined to allow the representation of structured data such as company profiles, as well as unstructured data, such as a picture of a company’s product.

1.2 The Classes of System Users

There are several kinds of users in the service model, each identified by the functions that he executes, or has the right to execute. Referring again to figure 1, a Data Provider processes source data, and enters this into the GDB as general data. A Data Consumer queries the system to obtain domain data or general data. A Domain Data Administrator derives domain data from general data; his role includes the definition of domain data schemas, their data structures and data location.

Note that a user might be a human acting behind a user interface, a program, or a combination of both. For example, a data consumer who executes the DD-Query and GD-Query functions defined below, is a human user who executes specially provided database software to perform these
1.3 The Data Handling Functions

There are four data handling functions of the service model (figure 1), all of which may be implemented by standard database tools and techniques. The **Acquisition** function implements the procedures to extract data from the source data set, and to insert this in the correct format into the GDB. This function possesses the user interface operations *put*, *remove* and *update* for entering, deleting and modifying documents in the GDB.

The **Derivation** function implements those procedures which extract data from the GDB, and inserts this data into a DDB. This function may only be executed by the DD-administrator or his software. This software also permits DD-schemas, and thus DDBs, to be defined and modified.

The **DD-Query** function provides those procedures used to query a DDB. It is implemented as a query language with user interface operations *query* and *get*. The *query* operation allows the user to selectively retrieve a list of Document IDs which match a search criteria specified in an input parameter. The *get* operation provides access to the actual information associated with a given Document ID.

The **GD-Query** function provides those procedures used to query the GDB. Like for the **DD-Query** function, it is implemented by a query language tool with user interface operations *query* and *get*.

1.4 The Security Functions

One of the important roles of the **Cwasar** service model is to enforce security, *cf.* the S of the functions in figure 1. Before describing the features that the model integrates for this purpose, we outline the security requirements which these features are designed to satisfy.

1.4.1 Requirements for security

Firstly, security constraints in **Cwasar** can be both *discretionary* and *mandatory*. By discretionary we mean that each member company of **Cwasar** may specify its own security rules governing access to the documents that it has entered into the GDB. By mandatory we mean that the system administrators can insist that supplementary security rules also be obeyed in addition to any discretionary constraints. In particular, we do not equate mandatory with any multi-level
security rules here, but rather mean any constraints that must be obeyed. One example of furnished mandatory constraints is given in section 1.5 where we overview the Business Yellow Pages service.

The principal reason for supporting mandatory security constraints is that Cwasar is operating in a European context. EU member states each have their own laws on computer security, and the states’ laws are sometimes conflicting. For instance, French law requires that a government license be sought to use encryption whereas in Sweden, one is obliged to have all electronic information relating to an individual encrypted. As another example, some countries forbid companies when citing their products to compare these products to those of competing companies. Hence, the service model must be able to enforce particular mandatory security constraints within each EU country.

A further requirement which we already mentioned for the security framework is that it must permit a wide range of user security requirements, or policies, to be expressed and enforced. For instance, companies might want to express policies of the form “Access to our company information is dependent on the clauses of an electronic contract which potential information accessors have signed” or “access is only permitted during certain times of the day and if the requesting user has authenticated himself to our authentication server S”. More generally, one should be able to express security policies which are more closely related to the data model semantics of the service; for instance, access constraints to a hotel’s reservation server may differ for regular clients who are allowed to book at short notice.

1.4.2 Approach to security

The basic approach taken to security is to represent a security policy as a program code unit termed a custodian [6]. The custodian is evaluated each time an access to data is attempted, executing the relevant security procedures, e.g., access control, audit, authentication. Further, a custodian is encapsulated and so no illegal tampering with the security rules and mechanisms within it may happen. The primary advantage of representing policies as program modules is that one can specify security policies which are more expressive than those that can be specified using the traditional data structure mechanisms such as capabilities or ACLs.

The custodian attached to a GDB document or DDB item is in most cases invoked directly. That is, the operation of the data handling function is first transferred to the custodian, where security actions take place, and from there a nested call is made to the data handling function software where the actual data handling operation takes place. This is the schema employed for example
in the _get_ operation of GD-Query or the _remove_ operation of Acquisition. Examples of security actions include audit measures (where for instance one might log accesses made to the Query services), access control to documents, integrity measures (for ensuring that modifications to GDB and DDB meta-data are correctly done) and authentication (of data consumers or administrators).

For the operations _query_ of GD-Query and _update_ of Acquisition, the invocation schema is slightly different due to non-determinism. Both these operations return a set of document identifiers where the documents are selected according to a selection criteria passed as an operation parameter. The set of document IDs returned cannot thus be known in advance. In these cases, the data handling software executes the operations, and then invokes the custodians associated with these documents in order to verify that the security policies contained in the custodians permit the return of these IDs.

Another feature of CWASAR security requirements is the definition of security for domain data items. A domain data item is dynamically instantiated from any number of GDB documents. The function that creates a DDB, DDBdefine [1], specifies code which is used to dynamically determine the GDB documents from which a new DDB item is derived. Ideally, one might like the security policy governing the new DDB item to be consistent in some way with the security actions defined in the custodian attached to each GDB document. For this reason, the DDBdefine function allows one to specify a custodian expression as parameter [4] - this expression is used to construct a new custodian from those belonging to the composing documents, and is then linked to the new DDB item.

A more detailed description of how security is integrated into the data handling functions is found in [3].

1.5 The Business Yellow Pages Service

The Business Yellow Pages (BYP) is one CWASAR service designed in accordance with the service model. BYP offers the retrieval of branch related company information and then the use of the information retrieved to set up a "point-to-point" connection between companies over the CWASAR network.

The principle behind BYP is that CWASAR member companies are each assigned to a single commercial branch, e.g., banking, hoteliers, shoe manufacturers. A domain data schema (DDB) is specially defined for each branch to reference the company information of the branch's members. A BYP DDB has a fixed data format which models mainly company names, points of contact and
product information [2]. A typical user service interaction scenario starts with a user issuing a DD-Query for say shoemaker information in the aim of finding a shoemaker near him who specializes in a certain class of shoes. The interaction starts by instantiating the DDB (using the Derivation function) and most future interactions will consist of querying and getting data within this DDB. Requests which cannot be satisfied by the shoemaker DDB are forwarded to the GDB through the GD-Query function. Once the user has found the coordinates of the shoemaker he needs, he may set up a point to point electronic connection with this shoemaker.

Two classes of mandatory security requirements were identified for BYP in the design stage’s market analysis: closed user groups and secure point to point communication. The closed user group abstraction is a closed BYP service making information available to only a subset of Cwasar members. For instance, only a shoemaker who has registered with the closed user group “private shoemakers” may have access to the company information of other BYP registered members of “private shoemakers”. The requirements for point to point communication are that only registered Cwasar members may use this message exchange service and that the message must remain unreadable except to the communicating partners.

2 Mapping the Service Model to its General Architecture

The preceding section outlined the Cwasar service model, describing how the system’s information base is organized and the manner in which services are structured on top of this base. Our goal in this section is to refine our model to a wide area network system since this is what Cwasar essentially is, and in so doing, to illustrate how the features of the model are implemented.

2.1 A Wide Area Network Architecture

The Cwasar architecture is foremost a wide-area network (WAN) (figure 2) which partly explains some of the user requirements that guided the service model’s design. The network is an interconnection of several publicly and privately owned networks where no access price regulation necessarily exists, and so services must be designed to keep network traffic as small as possible. A second feature of this architecture is the presence of multiple independent administrative domains. There is thus an absence of a global trusted authority, and as we mentioned, the actions of users in different domains may be governed by different laws.

A further feature of a WAN architecture is the presence of proximity servers. Their role is to
address the access cost issue, as well as performance overheads, by acting as a cache of information and services for client systems in their geographical vicinity. Employing a fixed number of proximity servers also has the effect of placing a threshold on the amount of caching that the system needs to control. A proximity server has a unique administrator which could be a telecom provider or a private company who charge for its usage. As we shall outline, the CWASAR service model's approach to security relies on the use of proximity servers.

2.2 CWASAR Architectural Configuration

The service model is implemented using the client-server approach. In order to facilitate system administration, our initial implementation objective is to store the whole of the GDB on a CWASAR server. There will be one such server placed in Spain, Germany and France; the reason for employing three separate CWASAR servers is that the information stored on each will be in the natural language of each host country. CWASAR servers are trusted by clients since they contain all company information. Figure 3 shows the general architecture of a CWASAR server.

Access to the server is made through the secure communications port. In the first version we use SecuDE [7] for communications security, that is, for client authentication and also for message authentication through digital signatures. SecuDE is a public domain software toolkit providing basic encryption functions (RSA, DES, hash functions), X.509 certificate handling functions and PEM processing features.

Each of the four data handling functions has servers which implement their operations on the CWASAR server. The custodian process implements security by ensuring that access to documents is mediated by the custodian set as described in the previous section.

In order to reduce network costs, clients using a CWASAR service access the information base through a proximity server placed in their locality. There will be a fixed number of proximity servers, and therefore many clients will use the same server. Clients may trust proximity servers in the sense that these servers are physically secure and run CWASAR software. The general architecture of client and proximity sites is shown in figure 4.

On the client side, which might just be a PC/modem pair, users formulate queries or acquisition commands through an interface language and these requests are then sent to the nearest proximity server. The proximity server has proxies for each of the four data handling functions of the model; proximity servers contact the server when they do not possess the information needed to satisfy user requests. Also, GDB modification requests made through the acquisition function are forwarded
to the Cw asar server for processing, and then eventually to other proximity servers. We have no fixed rule governing information consistency between the proximity servers; this depends on the kind of information being handled and the semantics of the services using this information. The Business Yellow Pages service for instance does not require strong consistency.

2.3 The Security of the Architecture

Cw asar and proximity servers contain custodian server processes implementing the security aspects of the system’s service model. The schemas of figures 3 and 4 illustrate the simplicity of the approach to security; notably, use of custodians is independent of the database software used to implement the data handling functions. Nevertheless, custodians possess important properties which are needed to help meet the security requirements.

Recall that a custodian encapsulates all security rules and procedures pertaining to access to company information. A custodian need not just contain an access control policy, but also audit and authentication mechanisms. Inclusion of the latter means for instance that custodians may generate session keys for users and servers that are used by the secure communications module. As said, by representing custodians as programs, security policies can be more expressive than the traditional data structure mechanisms. Such expressiveness is important in order to support the legal mandatory constraints that we can expect. Further, each security function may be tailored to the particular security requirements of a data or service provider. For instance, a provider who desires all accesses by foreign sites to its data to be logged, would program the audit sub-function of the custodian module for the *get* operation as

```
begin if caller ≠ German then write(logfile, caller) end-if.
```

A specific custodian will exist in each administrative domain that is used to locally implement security requirements particular to that domain, e.g., ensure that communications are encrypted. As the encryption problem of European law highlighted, there can be conflicts in the requirements of different domains. By using custodians to be solely responsible for security, these conflicts only arise concerning how custodians interact. For the example on conflicting encryption requirements between France and Sweden, there would be a third intermediary custodian placed on a proximity server in a neutral venue which exchanges encrypted messages with Sweden and decrypted messages with France.

A further goal of the security framework is extensibility. As things currently stand, companies can specify any custodian program (policy) for their company document entry. On access to
information in this document, the custodian process on the server will execute this program along with the custodian subprograms housing any mandatory security policy. Moreover, though we have chosen SecuDE as the current communications security package, this can easily be replaced by a package which is more oriented towards electronic commerce (e.g., packages with payment protocols). The current framework is firstly designed to address the requirement of supporting more expressive security policies governing access to data, as well as the ability of the system to enforce security in countries without breaking any legal imperatives.

3 Concluding Remarks

This paper has overviewed the design of the Cwasar infrastructure for providing European electronic commerce services. Two of the basic requirements for the system stemming from a market analysis in European countries were reduced network costs and security. The former requirement is handled in Cwasar through the use of proximity servers and domain data schemas which respectively store and structure information for each service in an efficient way. The security requirement is tackled through the use of custodians. This security mechanism is designed to allow for richer classes of security policies to be expressed by service and information providers, as well as to more easily integrate the individual computer security legal requirements of EU member states. On this final point, we should say that conflicting legal positions on security is an issue relevant to any system being designed to run in an international context, and not just in Europe.

Our current technical work in the project includes database integration, and in particular, the development of tool support for a general acquisition function able to collect information from a heterogeneous set of existing databases for integration into the GDB [1][2]. Another line of work which we have been following is security engineering [3]; this is concerned with the design of tools to aid users design and verify their custodian (policy). To do this, the user is first furnished with a library of policy component C++ classes from which he specializes policy units such as access matrix and audit logs to form his own policy [5]. Further, access control policies are first declared in a special type definition language for which a code generator and security proof system have been developed [4], permitting access control policy requirements/properties of the policy’s type to be verified. This library has been used to develop the initial version of the BYP service policy. Thus by employing code generation, we can benefit from security model support yet avoid difficult to find security flaws hidden in program code.
Acknowledgments This work is partially supported by the European Commission Telematics program under contract number 1005. Cwasar is a joint venture of Bull S.A. (France), CitiusNet (France), Derbi (France), Eco GmbH (Germany), Effort S.L. (Spain), and GMD - Forschungszentrum Informationstechnik GmbH (Germany).

References


Figure 1:
Figure 2:
Figure 3:
Figure 4:
**Figure 1**  Schema of the Cwasar Service Model.

**Figure 2**  A wide area network system.

**Figure 3**  Cwasar server structure.

**Figure 4**  Client site and proximity server structure.