



SETECS, Inc. Internet Security Technologies

Security Architecture for Development and Run-Time Support of Secure Network Applications

Sead Muffic, President/CEO

SETECS, Inc.

Silver Spring, MD 20910, USA

E-mail: sead.muffic@setecs.com

The white paper describes a generic security framework, which represents the concept, methodology, and development environment for secure network applications. It may be used as a design approach, as a development platform, and at the same time as a run-time support environment. The components of the framework are generic security objects providing security methods (functions) needed in many applications. Those objects perform simple cryptographic functions, create more complex encapsulation formats, provide local and remote security protocols, support secure communications, and implement various user interfaces. The framework is fully integrated with secure Linux operating system and with two secure communication protocols: SAML and SSL. Usage of the framework has many advantages: easier development, verification, testing and debugging of new secure applications; sharing of security credentials between multiple applications; uniform functioning of secure applications with respect to available security mechanisms and their level of security. The framework has been designed, developed, implemented and tested with several diverse network applications.

1 Current Development of Secure Network Applications

Currently, secure network applications are usually developed first with their basic functionality and security is added later, if ever, as special extensions and as add-on features. This is the main reason why most of the applications today have very limited or no security at all. If some already completed and operational application is to be enhanced with security, then the usual approach today is to invoke application programming interfaces (APIs) of some crypto library ([1], [2]) or some so called crypto services provider ([3], [4]). However, security tools and libraries today are not broadly available, sometimes not fully functional, and usually very complicated to use.

The current approach of adding security to the already established and functional applications has several disadvantages:

- Security development procedures are long and detailed;
- Verification and testing of security functions in the produced applications is complicated;
- Cryptographic and other APIs are usually at a very low level and therefore require expertise knowledge of security mechanisms and principles;
- Uniform secure functioning of applications is not achieved, because APIs are at a very low level;
- Libraries usually do not provide encapsulation modules, ASN.1 encoding/decoding procedures, padding and filtering, i.e. functions regularly required by all security standards;
- Security protocols, such as protection of passwords [5], certification protocol [6], access to smart card functions ([7], [8]), etc. are usually not available; and
- Compatibility, cross-platform availability, interoperability and strict compliance to standards are questionable.

One of the most popular and most frequently used security protocols in computer networks today is Secure Socket Layer (SSL), mainly for the reason that it is readily available in application components performing http protocol (browsers and Web servers). SSL is a good example of the approach that if some security feature is readily available within an application, it is easy to activate and use, and if it is adequate for particular security needs, then it is broadly used. This approach should be followed with all other network applications and their security features, mechanisms and protocols. However, in order to make security technologies available to users, they must be first available to developers of network applications. If developers have at their disposal fully functional, complete, correct, verified, and easy to use security development environment, they will develop security enhanced applications as a result of their regular application development activities. Furthermore, if security is already incorporated in tools and components of the development environment, then applications using these components will be inherently secure, i.e. immediately after completion of their development. In that case, no special additional activities are needed to enhance them with security.

This is in fact the motivation and the approach with the security framework described in this paper. It is incorporated in applications development methodology and development environment. Thus, it produces secure applications by a standard process of their development, and not through additional special activities. Furthermore, the framework is compliant with and supports creation of secure network applications functioning within comprehensive network security architecture. It directly links applications to the components of that architecture, which provide functions and protocols necessary for scaling and direct deployment of applications in global network environments.

Finally, as described in this paper, the framework is fully integrated with secure Linux operating system and with two secure communication protocols: SSL and IPSec. Thus the framework itself and all applications developed with the framework run in strongly secure run-time environment comprising secure operating system, protected application resources (programs and data), and secure data communications.

2 Network Security Architecture

A security architecture supporting global and integrated secure network applications may be described in the form of three layers, as shown in Figure 1:

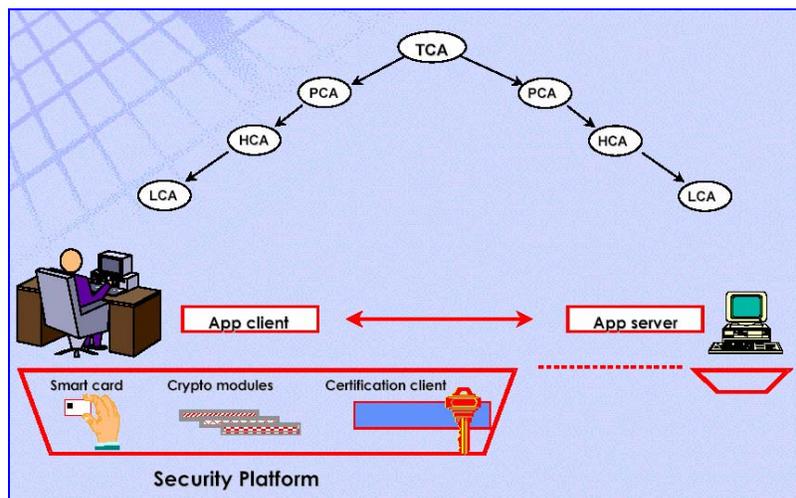


Figure 1: Layered Global Security Architecture

At the top of the system is a hierarchical public-key infrastructure (PKI) comprising of multiple Certification Authority (CA) servers: Top Level CA servers, Policy CA servers, (optional) Hierarchy CA servers and Local CA servers at the bottom of the hierarchy. PKIs may be established as a single hierarchy or as multiple cross-certified hierarchies. Top CA servers may also be used as the Bridge CA servers, while Local CA servers may be used as a stand-alone, single CA servers.

In the second layer “under” the PKI are various secure applications comprising security enhanced clients, security enhanced application servers, and various security protocols between them. Those may be standard security applications (like secure E-mail or secure WWW) or proprietary applications extended with security features and components.

Security clients and secure application servers are “on the top” of the supporting layer, called security platform, which provides cryptographic, encapsulation, certification, smart cards and other basic security mechanisms and functions.

Secure applications at the second layer may be further structured into two separate sub-layers, as shown in Figure 2: security administration (registration and certification) and client/server transactions. Registration and certification (as security administration functions) include registration of users and servers, their certification, and smart cards administration. The transactions layer underneath comprises security enhanced application clients, security enhanced application servers, and secure protocols between them.

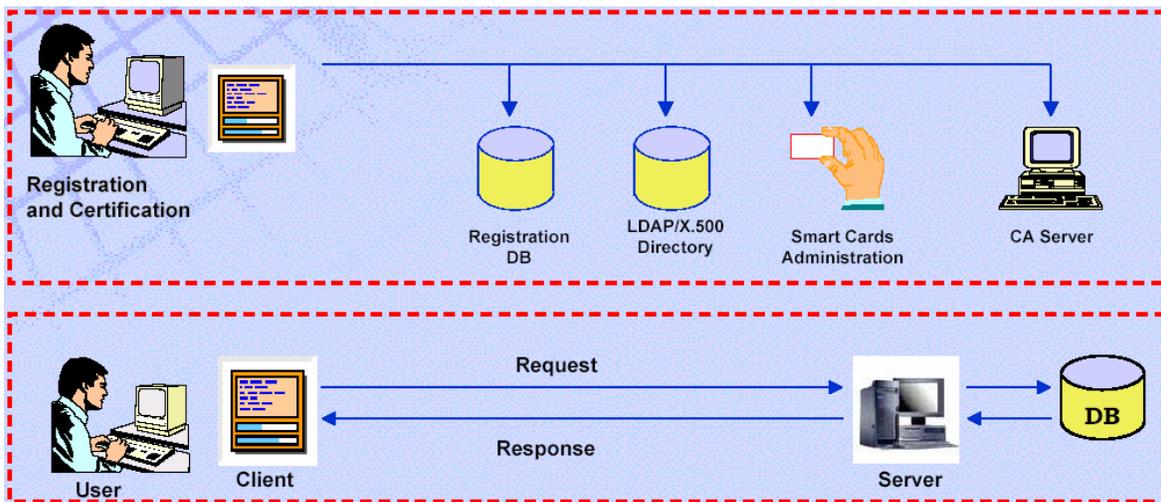


Figure 2: Layered Structure of Secure Network Applications

3 The Concept and Characteristics of the Security Framework

Security framework provides all components and functions described in the previous section in the form of secure objects and their methods. In addition, as explained in detail in section 4.4, it also provides key distribution protocol for secure group applications, which cannot be implemented as a simple extension of secure client-server key exchange protocols.

The framework is based on the following principles:

- There is a number of security and other supporting *functions* which are needed by every secure application;
- These functions are performed by certain *components* organized in the form of an architecture, each with specific protocols and interfaces;
- Each security application itself uses certain *resources* (interfaces, configuration files, audit logs, etc) needed for its standard or for its security enhanced functions.

Most of those components, functions, protocols, and features, which are needed to provide and/or support various security functions, are *common* to all secure applications. Therefore, the concept of the security framework was created with the goal to provide those common functions and components to multiple network applications. The framework is a collection of security objects, security protocols, and security system components, which provide development and run-time support for various secure network applications. It also includes template classes and methodology for creation of new security enhanced applications. The framework is complete, in terms of the methods and modules it provides, easy to understand and use, portable to multiple platforms, scalable, and reliable for high-value secure applications.

The security framework is more than just a collection of cryptographic objects, simply replacing equivalent cryptographic libraries. It first represents a *concept* by which development of network applications using objects of the framework will produce secure applications “out-of-the-box”. Furthermore, it is also a *methodology* where applications are developed by simple adjustments and modifications of several ready-made framework template classes. It enables the creation of instances and different versions of secure applications by a very simple and short process. By simple editing of their source code, template classes may be easily and quickly converted to specific, customized secure networks applications. Finally, the framework contains many ready-made operational modules and components performing the most common security functions needed by almost all network applications.

Based on the structure of a comprehensive security infrastructure supporting various secure applications, shown in Figures 1 and 2, the framework includes three groups of security objects and protocols:

- *Security platform*, a collection of various *security objects* and other utility objects, which constitute the supporting layer for each secure networks application;
- A number of various *application interfaces* for user and administrative functions: login, error and information messages, access to local protected resources, etc, and also several secure *protocols* between application components: strong (challenge/response) remote authentication, exchange of certificates and certification chains, protection of messages, etc., and
- *Security protocols* between application components and the supporting security infrastructure.

The objects comprising the framework are generic in a sense that the same object is used in several different forms, for many different purposes, in many instances, and performing several different functions. Thus, the complete framework comprises relatively small number of objects, yet provides a variety of security functions, which are usually needed by most of secure network applications. Initial concept of security objects are described in [9].

Therefore the security framework has the following distinguished features:

- It provides the concept, methodology, and a set of ready-made components for rapid development of security enhanced network applications;
- Additional features and functions of each application may be easily combined with the standard components of the framework in order to produce specific, customized secure networks applications; and
- The framework is flexible and offers the possibility of creating applications with different components of supporting security architecture, with different security functions, and with different assurance levels.

Applications produced using the framework are easy to configure, activate and use, since they share all security functions and resources within an integrated security environment.

4 The Components and Usage of the Security Framework

4.1 Development and Activation of Secure Network Applications

Secure applications are developed using the framework by first modifying several ready-made template classes in order to specify all user interfaces and functions of the new application. After this step, all functions specific for a new application show immediately in the user's interface (provided by the framework) as drop-down menus with their corresponding menu items. The next step is to develop modules performing those functions specific to a new application. Even for this step there are template classes to show how to develop new modules compatible with the framework. In the course of that development all major types of functions needed by an application, like accessing data bases, accessing LDAP/X.500 directories, using smart cards, transfer/communication of messages, etc. are supported by objects available within the framework. Since all these objects perform their functions in a secure way, their usage will result in all application functions being immediately enhanced with security features. Thus, in principle, development of new applications consists of editing of several template classes and creating some application specific classes, both tasks based on resources already available in the framework.

After a new secure application is developed and integrated into the framework, it is activated through activation of the framework. Alternatively, applications may also be launched individually, by linking their icon with the startup file of the framework. The framework first performs initial security activation functions: login, activation of a smart card (if used), fetching of certificates, etc. When those functions and steps are successfully completed, the framework displays security interface, which is used by the user to invoke either some other standard security functions of the framework or security functions specific for the selected application. The interface is very similar to the standard Windows Explorer interface with several security drop-down menus and the corresponding accelerating buttons. Some major security modules and the sequence of their invocations are described in the rest of this section.

4.2 Generic Local Login

Every secure application based on the security framework is initiated either by activating the framework or through its own application icon. In both cases the framework is activated first and it always starts with the local login procedure, whose panel is shown in Figure 3:



Figure 3: Local Login Panel

The local login procedure is generic, since the same procedure may be used:

- for users as well as for system administrators;
- with or without central registration data base;
- with or without certificates, and
- with or without smart cards.

Generic login module is very flexible: if smart cards are installed, then it will use smart cards; otherwise, an encrypted password will be used. If a central registration database is accessible, user registration data will be fetched from that database. Otherwise, a local registration file will be used.

The local login procedure creates personal user security profile (activates private keys, opens smart card, builds user security profile, etc.), which is further used in all other security protocols.

4.2 Generic Security Interface

After successful login, generic security interface is displayed. It provides several groups of functions:

- Switching (selecting) secure applications, which is performed using the first drop-down menu "Secure Applications";
- Selecting and manipulating the resources of various secure applications, which are shown on the applications tree;
- Access to components and functions of the supporting security infrastructure, performed using drop-down menus at the top of the interface;
- Access to local security components and functions (configurations, smart cards, audit logs, etc) which are also performed using drop-down menus at the top of the interface; and
- Display of information, messages, data forms and other aspects of secure applications using data panel of the security interface.

The layout of the interface is equivalent to the interface of the Windows Explorer. It has applications/resources tree on the left side, drop-down menus, acceleration buttons, and the same conventions for a single-click, double-click, and right-click actions, so it is very familiar, user-friendly, and easy to use. Various sections of the generic security interface are shown in Figure 4:

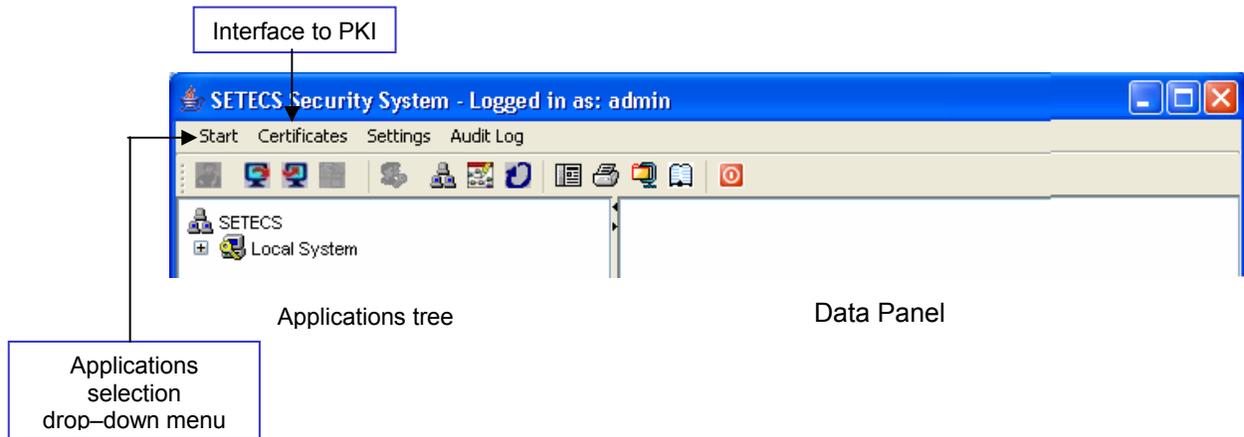


Figure 4: Generic Security System Interface

All secure applications are activated using the generic security system interface. The interface automatically displays in the "Start" drop-down menu all of the installed applications, so switching between applications is very simple. If an application has several sub-applications or if certain sections of an application are accessible only to certain predefined roles, then the security interface will also display applications sub-selection drop-down menu. Figure 5 illustrates activation of the Group Security Architecture (OneGroup™) system – a secure group communication application.

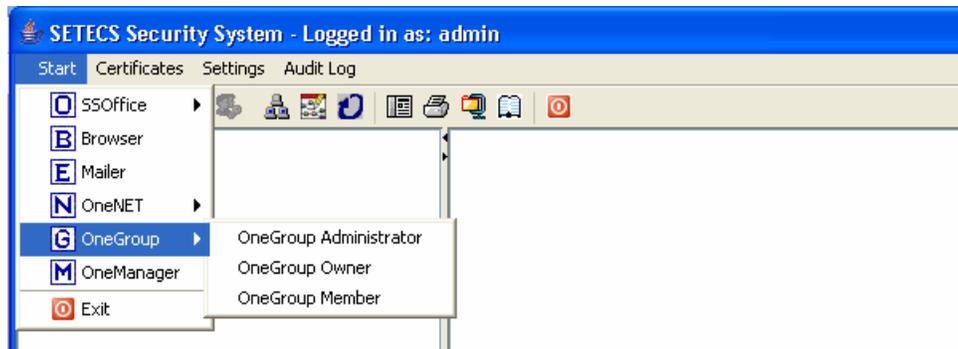


Figure 5: Activation of Secure Applications from the Generic Security System Interface

4.3 Other Generic Modules : Action Listeners

Besides generic login module and generic security system interface, the framework contains a number of modules, each performing one specific function invoked as so called action listener through drop-down menus appearing on the top of the security system interface. As an example we show functions of the "Certificates" drop-down menu:

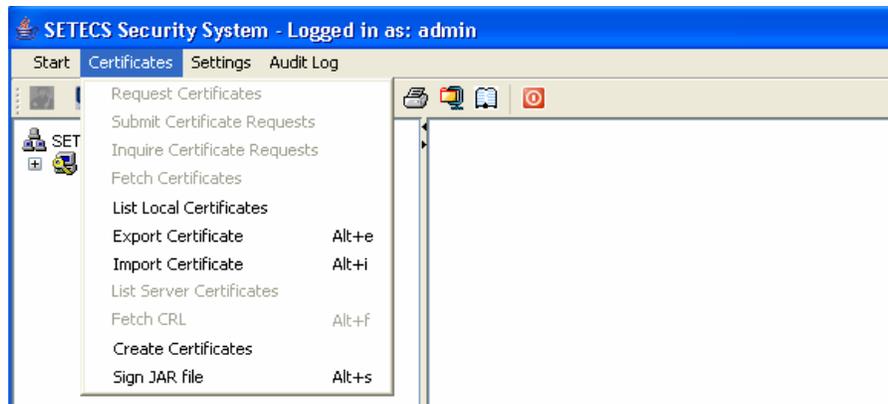


Figure 6: Certificates Functions

As already indicated in section 3 above, all these action listeners are always available as part of the framework, and resources which they handle (registration data, configuration data, certificates, CRLs, etc.) are shared by all applications. If locally configured CA server is not available, corresponding drop-down menu items are dimmed (as shown in Figure 6). The system is integrated ("sensitive" to instantiation of various secure objects), so as soon as the server is started, those drop-down menus will be activated. SETECS PKI system provides on-line issuing of certificates – low assurance (Request Certificates menu) or off-line issuing of certificates – high assurance (Submit Certificate Request, Fetch Certificates). The system automatically detects assurance level and adjusts availability of corresponding drop-down menu items.

4.4 Secure Applications

In section 3 it was indicated that each secure application performs two groups of functions:

- A set of common security functions, and
- A set of application specific security functions.

Common security functions for all secure applications are performed using drop-down menus of the generic security interface. Those drop-down menus are shown in the Figure 4 above. Application specific functions are performed using additional drop-down menus, which are automatically added to the generic interface when an application is selected (see Figure 5).

As an example, we use secure group application which has been developed using methodology and templates of the security framework. That application supports three roles: group administrator, group controller, and group member, each using individual application interface. Development process of an application is simply editing of several template classes specifying application-specific drop-down menus, their menu items and, of course, development of action listeners to handle those new application tasks. When ready, application is added to the security interface for invocation.

For instance, selecting "Administration" section of the *OneNET™ network security system* application will expand the generic security interface to include additional drop-down menus, specific for that section of the *OneNET™* application. These additional drop-down menus are shown in Figure 7:

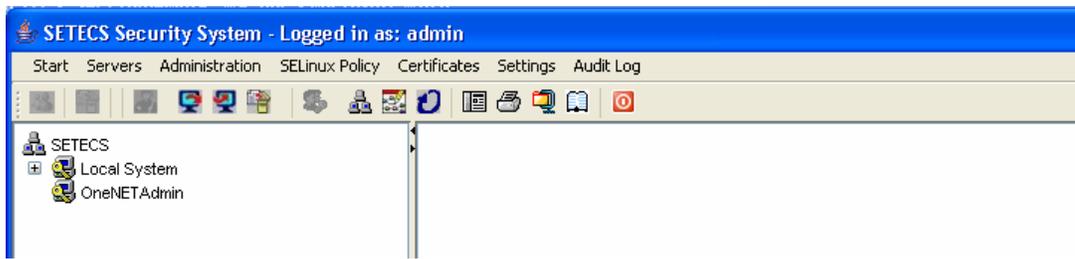


Figure 7: Application-specific Drop-Down Menus

If another section of the *OneNET™ network security system* is selected, for instance “OneNET Client”, then the generic security interface will be automatically updated with new drop-down menus, specific for that section of the *OneNET™* application, as shown in Figure 8:



Figure 8: Automatic switching of Application-specific Drop-Down Menus

It may be noticed that generic functions of the framework offered through the initial set of drop-down menus remain available for all sections of individual applications and for all applications accessible through the “Start” drop-down menu. More important, all resources, components, and data being handled through those drop-down menus are shared between all applications functioning within the security framework. Figure 8 also shows that application tree contains all active applications, so user may use simultaneously several applications at once, if authorized.

5 Additional Features and Benefits

One of the very important aspects of the security framework is that it is written in Java and therefore available for all operating systems. As such, security interface may be also be extended with other “Windows-like” functions, not always available in other operating systems. For instance, security framework currently offers the full protection of local files and documents using simple right-click menus. Verification is automatic, performed by a double-click on a protected file/document.

Another important feature of the security framework is that it is integrated with security of *Linux operating system* and with the two secure *communication protocols*. Security in Linux environment is based on security-enhanced Linux (SELinux) [10]) and secure communications are supported by SAML or SSL. The combination of the security framework and secure operating environment is the following:

- Security framework is used for distributed and consistent administration of SELinux policies in a networking environment and for configuring parameters for secure network associations based on SSL and SAML;

- SELinux and SSL are used as secure underlying operating platform for security framework itself and all its applications in a networking environment.

Thus, a combination of secure Linux, secure communication protocols, security framework and secure applications represents very good, comprehensive and integrated network security system, which is easy to install, configure, maintain, and extend with new applications and features.

Security framework has already been used for development of a number of secure applications, all shown in Figure 5. *SSOffice* provides security extensions of OpenOffice (or StarOffice) for automatic and transparent protection of all files and resources, based on user-configurable security policy. Browser and Mailer are security extensions of Mozilla browser/mail, supporting automatic activation of SSL, S/MIME protection of E-mail letters, and encryption of the mail address book. *OneNET™* is generic client-server application supporting strong authentication protocol, SSL and S/MIME communication security between a client and multiple *OneNET™* (proxy) servers. *OneGroup™* is a family of secure group applications: secure instant messaging, secure whiteboard, and secure sharing of documents. Finally, *OneMAN™* is integrated security management system performing registration of users and applications, their certification, management of SAML authorization roles and policies and issuing of smart cards to users. Other, additional secure applications may be easily added to the "Start" drop-down menu simply by extended one of the security framework classes.

References

- [1] OpenSSL, www.openssl.org
- [2] RSA Security, Inc. "BSAFE™", www.rsasecurity.com
- [3] SUN Corporation, *Java Cryptographic Extensions (JCE)*, www.sun.com
- [4] Microsoft Corporation, *Cryptographic Services Provider (CSP)*, www.microsoft.com
- [5] NIST, *Federal Information Processing Standard (FIPS) 112*, www.nist.gov
- [6] Internet Engineering Task Force (IETF), "Certificate Management Protocol", RFC 2510, www.ietf.org
- [7] Open Card Framework (OCF), "Programmers Guide", www.opencard.org
- [8] Dray, J. et al., "Government Smart Card Interoperability Specification", NIST Internal Report 6887. www.nist.gov, June 2002
- [9] Morogan, M.C., "Generic Security Objects: A Comprehensive Java Security System for Open Networks ", Licentiate Thesis, Royal Institute of Technology, Stockholm, Sweden, November 2000
- [10] NSA, "Security Enhanced Linux (SELinux)", www.nsa.gov/selinux
- [11] Kent, S., Atkinson, R., "Security Architecture for the Internet Protocol", RFC2401, Nov 1998, www.ietf.org