Issues in Developing Security Wrapper Technology for COTS Software Products

Dean, John; Li, L.

*Issues in Developing Security Wrapper Technology for COTS Software Products**

Dean, J. Li, L.
February 2002

* published in: Proceedings of the 1st International Conference on COTS-Based Software Systems, Orlando, Florida. February 4-6, 2002. pp. 76-85. NRC 44924. (LNCS 2255, Springer).

Issues in Developing Security Wrapper Technology for COTS Software Products

John C. Dean[1], CD and Li Li[2]

[1]National Research Council Canada, Software Engineering Group, Building M-50, Montreal Road, Ottawa, Ontario, Canada, K1A 0R6
John.Dean@nrc.ca
[2]Entrust Technologies, 9 Auriga Drive, Nepean, Ontario, Canada, K2C 7Y7
Li.Li@entrust.com

Abstract. The use of Commercial Off-The-Shelf (COTS) software products as components of large-scale systems has become more and more pervasive. One of the interesting questions that has arisen is "Can you build secure applications using insecure components?" We have been investigating ways to protect data that is shared between two or more independent, insecure applications. Our initial attempts to accomplish secure data storage and transfer have been directed toward building data encryption tools that interact with various COTS products. The goal was to test our theory that security wrappers for COTS products are feasible. This paper describes a security wrapper technology that we have implemented for selected (COTS) software products. The technology focuses on interchangeability for COTS software components, portability for the wrapper, and security for communications between applications via the wrapper. By applying this security wrapper technology, one COTS software component to be wrapped can be replaced by another without significantly modifying the wrapper; the wrapper can work with a variety of operating systems; and data can be encrypted and stored temporarily or permanently.

1. Introduction

The use of COTS software products as components of large-scale systems has become more and more pervasive. Wrapper technology for these components is not new. It has been widely used but most of the current wrappers were concerned with providing data translation services and inhibiting direct access to the component. Little research has been done on the implications of securing data transmission between individual components. When security issues are considered, one of the interesting questions that has arisen is “Is it feasible to build secure applications using insecure components?” Our goal is to build a generic security wrapper for insecure COTS products, thus allowing us to implement secure systems using these wrapped products.

The wrapper project focuses on three issues: interchangeability, portability, and security. The ideal situation is that by applying our security wrapper technology, one COTS software component can be replaced by another, without significantly modifying the wrapper. In addition, the wrapper should be portable to a variety of operating systems and hardware platforms with minimal modification. Finally, it should provide for secure data transfer between components.

This paper describes the security wrapper technology that we have implemented for selected (COTS) software products. Discussions cover feasibility analysis, experimental solutions, results and directions for further research.

2. Background

The goal of this research was to determine if we could somehow create a security interface that would allow a system integrator to provide security services and secure data transfer between two insecure applications. This capability must be able to bypass the normal user interface because these COTS products must be invoked programmatically, that is, by another application or by the system controller software.

Given this constraint it is interesting to consider a novel point of view about the relationship between common security systems and the user. With most third party security implementations such as Entrust, PGP, etc. the user acquires a persona in the form of a "profile" that consists of, at a minimum, a security signature and a pair of encryption/decryption keys (public and private). These uniquely identify this user within the security envelope. For example, by supplying the public key to acquaintances, the user can receive and decrypt data that has been encrypted using his/her public key and sent to him/her. In addition the security signature allows the user to sign documents and forward them to a recipient. The recipient can confirm that the document actually came from the sender by verifying the signature against a certificate held by the Certifying Authority.

The problem we face in working with COTS applications is that there is normally no human user intervention in the operation of these applications when they are embedded within a larger scale system. The applications are invoked programmatically and acquire data from other applications directly. Most of the applications do not have an integrated security system for encrypting and decrypting data and this functionality must be supplied in order to ensure secure and trusted data transfer.

We proposed to develop security wrappers for COTS products that take on the role of the user within the COTS-based system. In other words the wrappers acquire the security 'persona' and perform the actions of encryption and decryption and signing data to be transferred between two cooperating COTS applications within the system. This wrapper would provide a purely programmatic interface between the COTS product and both the Certificate Authority and the other cooperating COTS products within the system. Each COTS product could acquire a security persona (a profile) by having the wrapper query the Certificate Authority to either establish a new profile or access an existing one. These personae would then be used to allow for secure communication (data transfer) between applications via the wrappers.

In general, wrappers are software modules that are responsible for connecting individual information sources to external users. For example, in some database access wrappers for a specific system, a wrapper receives and translates instructions written in the corresponding language into executable local repository commands. In a security wrapper, the information sources to be translated are raw (decrypted) data and the external information is encrypted data. We designed this COTS security wrapper to provide a level of security in the wrapper that is not present in the COTS components. That is, the wrapper is used to add security to non-secure components. We chose several COTS software products as our elements of experiment, according to the following COTS software properties:

? it is received from the distributor and used "as is" with no changes to the source code;
? it is installed in multiple sites within different organizations; and
? the users and/or integrators do not control the evolution and maintenance of the software.[1]

We chose the following software as our COTS products: Entrust and the Entrust Java Toolkit, and a number of other commonly used desktop applications (MS Word, MS Excel, StarOffice). The platform environments for the experiments are three typical operating systems: Windows NT, Solaris UNIX, and RedHat Linux.

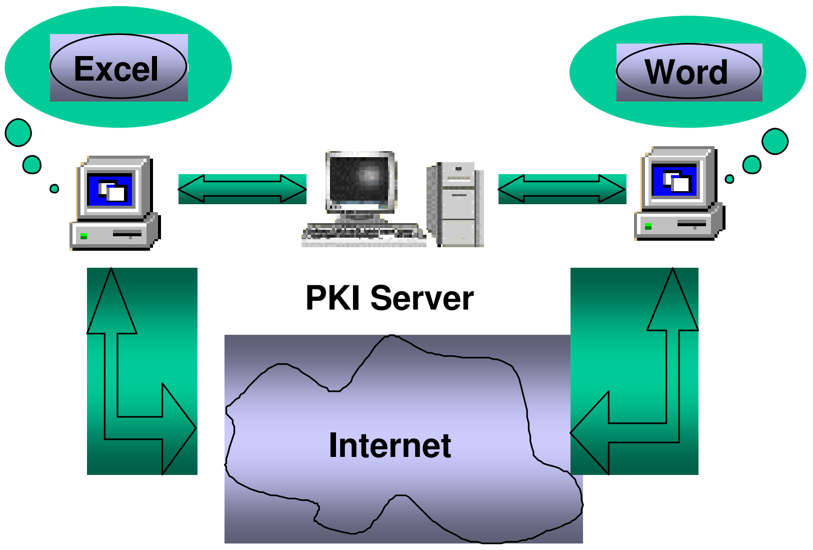

Fig. 1. Communication between distributed components

2.1 Wrapping Software Components

There are two key methods by which one can provide security for insecure COTS products. One method is to create a security layer between the application and the infrastructure support for the application. This layer might provide the interface between the application and the operating system or, for distributed systems, between the target application and the network transport. In either case the solution does not really involve isolating the product completely. Rather it involves intercepting input and output data at the infrastructure boundary and manipulating it by either encrypting or decrypting the data[2].

The second method is to provide a custom wrapper that encloses the product and isolates its interface from the user. With this method individual wrappers are responsible for encryption and decryption of data as necessary. This transfers responsibility from the platform to the wrapper itself and, in doing so, provides for the implementation of portable wrappers. We chose to investigate the individual wrapper method because it allowed us to implement our concept of the acquisition of a persona by a COTS application within the wrapper.

2.2 Entrust and Profiles

We chose to use Entrust technology because Entrust provides a Java toolkit that fits in our need for portability. The Toolkit works in conjunction with the Entrust Public Key Infrastructure architecture and the standard Java Development Kit (JDK) to form a framework for implementing Entrust security. Figure 2 shows the relationship between the various security components. The JDK supplies the class libraries used to create the wrapper infrastructure, including its interaction with the input/output functions of the current operating platform.

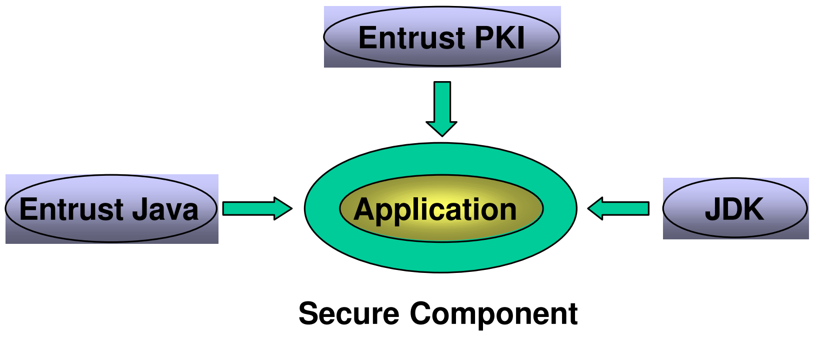

Fig. 2. Security Components

The toolkit provides a set of APIs that enables Java application to perform security-related tasks. The toolkit is implemented as a collection of functional classes. The fundamental class is the Entrust Profile. Keys and certificates that identify a certain user are stored in this class. Each Entrust Profile represents a user. When a user instantiates an Entrust Profile object that represents his or her Entrust profile, and logs on this profile, the Entrust profile can be used to perform security operations. Each profile is personalized and will merge its unique persona into the wrapper. By using individual Entrust profiles, different users will be using the same wrapper but with different persona.

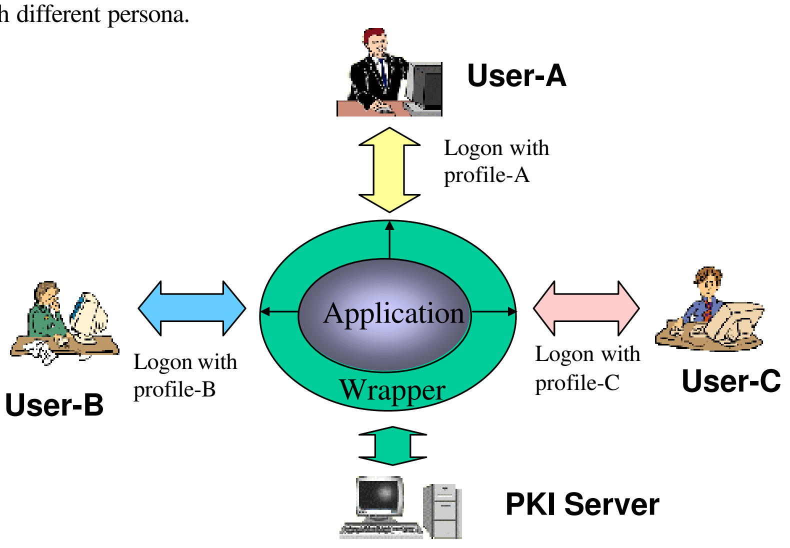

Fig. 3. Multiple persona usage

Figure 3 illustrates how these personae interact with a wrapper that wraps an individual application. Users who want to access the application must logon to the wrapper first. Each user logs on with his or her unique profile. The wrapper then validates the keys and certificate of the profile by contacting the PKI server. This ensures the keys and certificate in the profile are always updated. Note that although the Figure shows these 'users' as people they could, and in fact probably commonly would, be other applications within the COTS-based software system.

3. Implementation Criteria

Our plan was to implement a COTS security wrapper that would allow us to examine three criteria: interchangeability, portability and security. The ideal situation would be that the wrapper works with a variety of operating systems; the COTS application can be replaced inside the wrapper easily, without significantly modifying the wrapper; and that the security of data transferred between cooperating applications was highly ensured. This section will discuss the three criteria and describe the techniques employed to fulfill them.

3.1 Interchangeability

High interchangeability means that the wrapper should be able to wrap different applications with minimal changes to the wrapper code. There were two cases we needed to consider. First, we needed this capability because we recognized that the applications themselves may not be portable between operating systems. For example, Microsoft Excel does not run on the Linux operating system and therefore wee would have to substitute a competing product if we wish to use the wrapper on another platform. So, we wanted to ensure that substituting the StarOffice word processor for Corel WordPerfect was feasible. Secondly, we wanted to be able to substitute applications on the same platform as well. This would allow us to confirm the generic attributes of the wrapper. For example, we wanted to be able to substitute Microsoft Excel for Microsoft word with as little modification to the wrapper as possible.

The most important concept is that we needed to minimize the interface between the wrapper and the COTS product as much as possible. Since Microsoft Word, Microsoft Excel, StarOffice are COTS products, these components should be used on an "as is" basis. We accomplished this in our initial experiments by only allowing the COTS applications to be invoked using a command line call. This allowed us to effectively transparently substitute applications at will.

However, with every advantage gained there is a cost. The cost of implementing the wrapper in this way was that we introduced a complexity in intercepting the applications i/o calls. In addition to being platform dependent the input/output mechanisms are application-dependent as well. We implemented a 'brute-force' style algorithm to temporarily overcome this problem. Current research efforts are focused on developing a more efficient and effective solution.

3.2 Portability

Portability means that the wrapper can be transferred easily from one operating platform to another. The wrapper should be able to work with a variety of operating systems without significantly modifying the wrapper itself. This is affected by primarily the choice of programming language for implementing the wrapper. While this was could be difficult to accomplish with many programming languages, we chose to use Java which allowed us to write the code once and install it on multiple platforms. Again this approach has limitations in that the Java environment does not provide sufficient low-level access to the file system on the various platforms. As we become more dependent on operating system functionality we pay a price in portability.

We implemented two versions of the wrapper, a GUI version that allowed for direct user intervention and interaction and non- GUI version that was intended to be invoked programmatically. The GUI version was only implemented in order to allow us to demonstrate the concepts while we intended the non-GUI version to be the one most commonly used within a COTS-based system. In order to ensure that both versions exhibited the required portability we implemented the GUI component using Java Swing components, which have advantages of speed, keyboard navigation and plugable, system independent look-and-feel.

3.3 Security

The security is ensured by a dynamic method - a wrapped application acquires a persona via a profile, which is created at runtime, and uses the profile to encrypt and identify its output data to other "wrapped", trusted applications. The wrapper is password-protected. The wrapper itself does not store the password information. Entrust profile does instead. This is a significant advantage of using the Entrust profile scheme. Since each Entrust profile represents a user, the wrapper allows users to create new users or recover themse lves via the Entrust PKI server. The procedure is implemented interactively with an Entrust PKI server. That way the Entrust profiles can be created or recovered at run time. Once the Entrust profile is saved locally, user can either log on to a Entrust profile online or offline. The wrapper will check the password and profile given by the user, verify that this is a valid user. Otherwise the log on attempt will be rejected and an error message will be given.

The encryption scheme is relatively straightforward. All raw data saved from the wrapped application is intercepted and encrypted transparently to the COTS application. Data files read by the wrapper for input to the wrapped COTS product are decrypted before being passed to the application. This is completely effective because the application cannot be invoked directly but only via the wrapper.

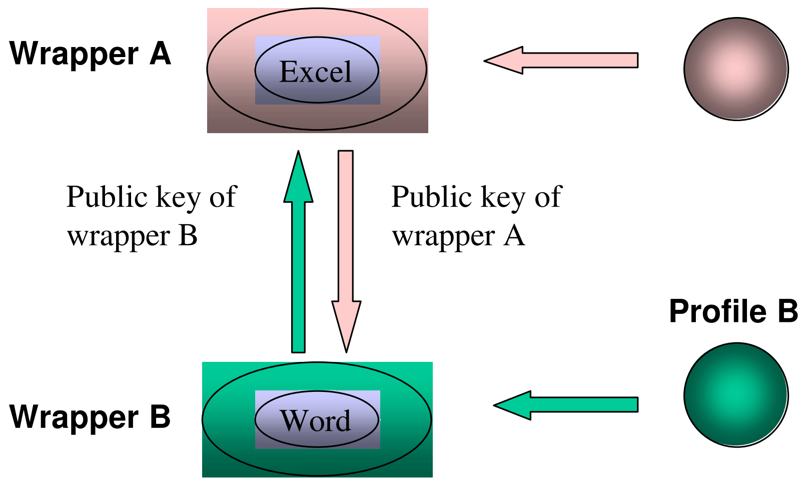

Fig. 4. Communication using different profiles

4. Experiments

We divided the experiment into three sections in order to verifying the interchangeability, portability and security capabilities of the wrapper. This section describes some of our results to date. As this is an ongoing research effort we anticipate being able to elaborate on some of the results in the future.

4.1 Interchangeability

To wrap different components, the wrapper has to find the main characteristics of a software component in order to recognize it, memorize it, and then recall it when later use. We define the main characteristics as the application's executable file path and the file format. We enable the wrapper to read them in, save them, and apply them. To configure the wrapper, the installer has to browse the operating system to find the exact executable file to be invoked. The installer also has to know the file format of this particular software. In addition, the wrapper was constructed in such a way to allow for different command line structure on different platforms by ensuring the command string could be provided via the configuration file and not hard coding it into the wrapper code itself. In this way we were able to easily switch applications as well.

The wrapper was tested with a number of applications, notably Microsoft Word, Excel and PowerPoint on the Windows NT machine. We also tested it using the StarOffice 5.1 suite of applications on the Linux and Solaris systems. It performed flawlessly in all cases. The only modifications required were to a configuration file at installation. However we intend to further automate this capability in a future experiment.

4.2 Wrapper Portability

We choose three commonly used operating systems: Windows NT, Solaris UNIX, and RedHat Linux, to experiment with the portability of the wrapper. The wrapper was built using pure Java and Java Swing components, and no extra layer was added to the local system. The wrapper was originally developed for the Windows NT operating system and was ported later to the Linux and Solaris systems. In order to port the application we had to move the Entrust Java support files to the new platforms. This was accomplished by simply copying the files to an appropriate location on the new machine and setting the classpath to the correct value. The porting required no changes to the custom Java code, however we did need to change one configuration file in order to account for the different paths to the executable being wrapped. We stress that wrapper portability will likely become an issue as we extend the call interception mechanism. We foresee that in the future there may be two components to our wrapper, a portable one and one that is platform dependent.

4.3 Security Analysis

During the implementation the application is invoked by the wrapper, simply by calling the executable file. The wrapper subsequently keeps track with the application's read and write actions by recognizing file format and timestamps. Any change made to local or remote files by the application is protected by automatic encryption and decryption within the wrapper.

There are two ways to use the Entrust profile: logging onto the profile either online or offline. The difference between online and offline may have some significance when user is due for a key rollover or a Distinguished Name change. In this case working online is also a part of the certificate verification proce ss, and the wrapper may end up encrypting messages for users with revoked certificates or trusting signatures that user should not trust.

We enabled the wrapper to work both online or offline and, in the case of the demonstration system, allowed the user to make choices at initialization either to create a new persona or to assume an existing one. When the user chooses to create a new user, the new user ID is securely generated when users enter a one-time use reference number and authorization code via the wrapper's GUI. The wrapper provides users with a transparent and automatic way of creating their own identities. By applying the Entrust Toolkit, the wrapper provides a broad range of algorithms developed both in North America and in Europe:

- Symmetric: Triple-DES, DES, CAST, RC2 and IDEA
- Asymmetric: RSA, DSA, Diffie-Hellman and Elliptic-curve
- Hashing: SHA-1 and MD

This was not a significant issue for our non-GUI wrapper except that in that case the profile choice is pre-programmed because we do not normally expect user intervention.

There are a number of unresolved issues with the wrapper as it is currently implemented. The most significant of these is the inability of the wrapper to track and encrypt temporary files created during the operation of the wrapped application. For example, Microsoft applications create a number of buffering temporary file in which intermediate data is stored. One of the uses for this data is to enable rollback of modifications (the undo function). To be truly secure our wrapper will have to eventually account for this. Again this is a topic for future research.

5. Conclusions

We have successfully implemented a COTS security wrapper and used it to construct building a security COTS mini-system using insecure COTS components. The wrappers demonstrate that it is feasible to build a security property into the architecture of a COTS-based system, while all the components integrated remain untouched.

The wrapper functioned effectively in three areas:

- Interchangeability
- Portability
- Security

Our implementation of the wrapper allowed us to interchange COTS products at will, often quite transparently to the external system. We also were able to port the wrapper to a number of platforms with no changes to the core code and minimal changes to the configuration files. Finally we were successful in implementing encryption and decryption for all permanent input and output data.

The optional graphical interface of the wrapper is entirely separated from the implementation of the wrapper itself. It remains independent of the underlying component so that wrapper substitution is possible. The interface is also under configuration management.

This research is in a preliminary stage and it is obvious that there are a number of issues yet to be investigated. They include developing a more efficient method of intercepting the input-output data streams (possibly by hooking interrupts) and examining the communications between multiple applications. We also believe that there are significant issues in embedding multiple cooperating wrappers in an actual system. However, the evidence we have presented implies that these types of wrappers are feasible and our efforts are now being directed towards evaluating this technology further.

References

1. Tran, Tam, Interoperability and Security Support for Heterogeneous COTS, Master's Thesis, United States Naval Postgraduate School Monterey Ca, 2000.
2. Badger, L, Feldman. M., Ko, C., Secure Execution Environments, Generic Software Wrappers – DARPA/ITO Project Summary, September 2000. http://www.pgp.com/research/nailabs/secure-execution/wrappers-darpa.asp
3. Mitchum, Terrence, Hypervisors for Security and Robustness, Secure Computing Corp Roseville Mn, 1999.
4. Meeson, Reginald, Analysis of Secure Wrapping Technologies, Institute for Defense Analyses Alexandria Va, 1997.
5. Dean, J.C., Security Wrapper Technology for COTS Software Products, 13th Annual Software Technology Conference, Salt Lake City, Utah, May 2001.