SECURE COMPONENT DISTRIBUTION USING WEBCOM

Simon N. Foley, Thomas B. Quillinan, John P. Morrison
Department of Computer Science,
University College,
Cork,
Ireland.
{s.foley,t.quillinan,j.morrison}@cs.ucc.ie

Abstract WebCom is a distributed computing architecture that may be used to distribute application components for execution over a network. A practical trust management system for the WebCom architecture is described. KeyNote-based authorization credentials are used to determine whether a WebCom server is authorised to schedule, and whether a WebCom client is authorised to execute, mobile application components. Secure WebCom provides a meta-language for bringing together the components of a distributed application in such a way that the components need not concern themselves with security issues.

Keywords: Trust Management; Mobile Computation; Distributed System Security;

1. INTRODUCTION

WebCom [12, 11] is a client/server based system that may be used to schedule mobile application components for execution across a network. Applications in WebCom are programmed as hierarchical *Condensed Graphs* of mobile components/code. Condensed Graph components represent tasks, or graphs of sub components, that are made available for execution across a network. These graphs are used to specify component synchronization and communication, and constrain component scheduling in a demmand driven (pull) or data-flow driven (push) manner.

WebCom masters schedule the components of an application graph to connected WebCom clients/servers, which in turn, execute scheduled components. A number of security concerns arise when using WebCom to schedule and/or distribute components across a network. In this paper we are concerned with the management of the trust relationships between WebCom Servers and WebCom clients.

A WebCom master must prove to a client that it is authorized to schedule a particular component to the client. Note that a WebCom master schedules more than just mobile code: it schedules mobile computations that are comprised of tasks and their inputs. A client must determine whether it should trust the master for the task that it schedules along with the inputs to the task that the master provides and also the subsequent output that is returned to the master. A WebCom client must prove to the WebCom server that it can be trusted to execute the particular component, along with its associated input data.

In this paper we describe how the KeyNote trust management system [2] can be used to manage authorizations and trust relationships between WebCom masters and clients. Trust Management [2, 14] is an approach to constructing and interpreting the trust relationships among public keys that are used to mediate security critical actions. Credentials are used to specify delegation of authority among public keys, and are used to determine if a signed request complies with a local authorization policy. KeyNote [2] is an expressive and flexible trust management scheme that provides a simple credential notation for expressing both security policies and delegation. KeyNote has been used to provide trust management for applications that include active networks [4] and to control access to Web pages [1].

An added advantage of integrating KeyNote in WebCom is that application components need not use the KeyNote API and, therefore, need not be aware, in a programmatical sense, of the trust management system. Secure WebCom acts as a form of reference monitor, using the contextual information in the Condensed Graph of an application to construct KeyNote queries to determine the authorization for the application components. This loose coupling of functionality and trust management leads to applications that are easier to develop, understand, maintain and secure.

The paper is organized as follows. Sections 2 and 3 outline the KeyNote and WebCom architectures, respectively. Section 4 describes how KeyNote is integrated into WebCom by using KeyNote credentials to determine the authorization of X509 authenticated SSL connections. Section 5 outlines some of the implementation issues. Section 6 provides general observations and discussion. An earlier version of this work was presented as [7].

2. TRUST MANAGEMENT

When a request from an untrusted principle (key) is made to a networked application to execute a particular action, then, authentication notwithstanding, the application must determine whether the key(s) that made the request is authorized. Authorization comes in the form of digitally signed public key credentials that bind public keys to the authorization to perform various actions. In practice, authorization is achieved by a collection of credentials that exhibit the necessary trust relationships between their keys. Given a policy (public keys, trusted in known ways), and a collection of credentials, a network application must determine whether a particular public key is authorized to request a particular operation.

EXAMPLE 1 A simple purchase order processing application runs on a server and accepts requests from clients. The request for operation prop is made to propose a new order while the request issue is used to inspect and issue the order. We expect that, in practice, a clerk will have the authority to propose orders and a supervisor will have the authority to validate orders; this authority will be delegated by their manager.

Assume that the owner of public key `Kmgr` is trusted to make requests to the order processing application. This is specified by the following KeyNote credential.

```
Authorizer: "POLICY"
licensees:  "Kmgr"
Conditions: app_domain=="OrderApp" &&
           (oper=="prop" || oper=="issue");
```

This is a special *policy* credential that defines the conditions under which requests from the licensee key `Kmgr` may be trusted by the application `OrderApp`. These conditions are defined using a C like expression syntax in terms of the *action attributes*, in this example, `app_domain` and `oper` which are used characterize the circumstances of a request.

The owner of public key `Kmgr` has the authority to delegate this trust to other keys and does so by signing the following credential for a clerk who owns public key `Kalice`.

```
Authorizer: "Kmgr"
licensees:  "Kalice"
Conditions: app_domain=="OrderApp"
            && oper=="prop";
```

In signing this credential, authorizer `Kmgr` delegates authority for proposing orders to the key `Kalice`. When Alice proposes an order (signed by `Kalice`), she presents this credential as proof of authorization. We can confirm that this key is indeed authorized since, by default (policy), we trust `Kmgr` to propose and issue orders and `Kmgr` has delegated some of this trust to `Kalice`, by virtue of signing the credential. △

An application may use a Trust Management (TM) scheme such as KeyNote [2] to determine whether requests to it are authorized, without the application having to know about how that determination is made.

EXAMPLE 2 When the `OrderApp` application (Example 1) queries the KeyNote TM system to determine whether it is safe to execute a particular request, it must specify the circumstances of the query. These circumstances include: *action authorizers*, corresponding to the key(s) that made the request; *action attribute set*, which is a set of action attribute name and value pairs that characterize the request; *policy* credentials, representing the keys that are trusted, and other *credentials* as provided by the requester and/or PKI.

For example, when `Kalice` requests an order proposal then the order application queries KeyNote with action autoriser `Kalice`, action attribute set $\{\texttt{app\_domain} \leftarrow \texttt{"OrderApp"}, \texttt{oper} \leftarrow \texttt{"prop"}\}$, the policy credential for `Kmgr` above, and a set of signed credentials provided by Alice. KeyNote must determine if the given request is authorized based for the circumstances provided. The application interacts with KeyNote via calls to the KeyNote API. $\triangle$

The KeyNote architecture provides a level of separation between the provision of security policy authorization and application functionality. As a software engineering paradigm, techniques that support separation of concerns for security [2, 5], synchronization [10], and so forth are desirable since they lead to applications that are easier to develop, understand and maintain.

KeyNote provides this separation of concerns at a conceptual level. However, as illustrated in Example 2, calls to the KeyNote API must still be coded as part of the application system. While ensuring cohesive applications, there remains a coupling within the application between the functional and security concerns, that is, we do not have complete separation of concerns at the code level, since security-critical calls to the trust management API are intertwined with the functionality.

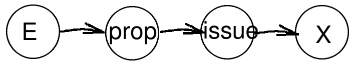
3. THE WEBCOM DISTRIBUTED SYSTEM

The WebCom system [12] uses a variant of the client/server paradigm to distribute operations for execution over a network. The system uses implicit program parallelism, separates the application program from the underlying computation engine and is efficient yet compact. The heart of the WebCom system is the Condensed Graphs computational model that it employs [13]. Applications are coded as hierarchical graphs which provide a simple notation in which lazy, eager and imperative computation can be naturally expressed. There are two types of distributable operation: nodes that represent atomic tasks and condensed nodes that represent subtasks encapsulated as subgraphs.

The top-level architecture of WebCom consists of a master and an arbitrary number of clients. Clients connect to the master, which in turn assigns them operations for execution.

An advantage of developing distributed applications as Condensed Graphs is that their implementation (graph) can be coded independently of the underlying system and/or network architecture. Atomic operations are value-transforming actions and can be defined at any level of granularity, ranging from low-level machine instructions to mobile-code programs such as applets, COM objects or COTS components. Atomic operations need not address synchronization or concurrency concerns: such details are implicitly specified in the Condensed Graph and managed by WebCom. Therefore, WebCom provides for a separation of concerns at the code level, allowing a loose coupling between functionality and control.

EXAMPLE 3 Condensed Graphs may be used as a distributed job control language to describe application system scheduling requirements such as workflow. Order processing (Example 1) may be defined as the following graph. This graph defines how the application components prop and issue should be scheduled to client workstations.



By definition, a condensed node such as order is constructed as a graph with a single enter node (E) and a single exit node (X). A WebCom master schedules atomic actions prop and issue to appropriate connected (WebCom) clients. △

4. INTEGRATING KEYNOTE INTO WEBCOM

Figure 1 illustrates how the KeyNote trust management scheme is integrated into WebCom by regarding WebCom as an application. The WebCom master authenticates its clients and uses their credentials to determine what operations it may schedule to them. Each WebCom client has a trust management architecture that is similar to the KeyNote architecture, authenticating the master and using the master's credentials to determine whether it is authorized to schedule the operation.

EXAMPLE 4 We assume that there is just one root public key Kwebcom as specified by the following KeyNote policy credential.

```
Authorizer: "POLICY"
licensees: "Kwebcom"
Conditions: app_domain=="WebCom";
```

This credential uses just one action attribute—`app_domain`—to specify that `Kwebcom` is unconditionally trusted to delegate authority related to the WebCom application. △

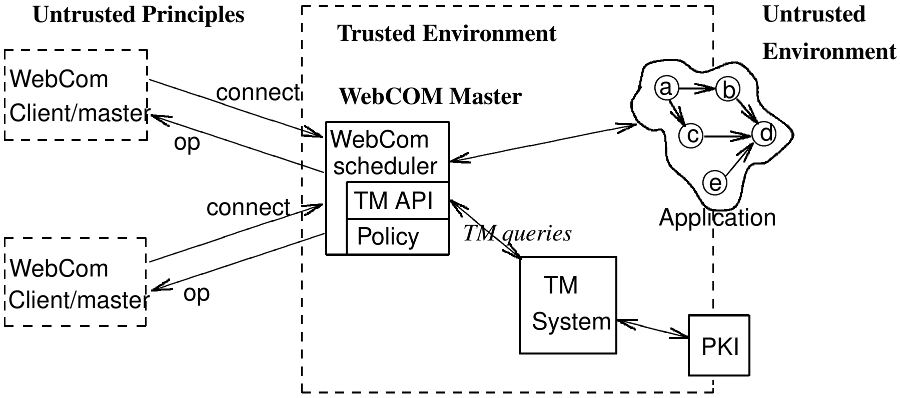


Figure 1. WebCOM-KeyNote Architecture

Unlike KeyNote, the Secure Secure WebCom architecture allows applications, if desired, to be regarded as untrusted since they do not contain any security critical code. The WebCom scheduler is the security-critical component and acts as a form of reference monitor, interfacing with the KeyNote system to decide how to distribute, control and synchronize applications.

4.1. WebCom Master Authorization

WebCom masters schedule operations to be executed by WebCom clients. Some of these operations may access local resources on the client and therefore the WebCom master must prove that it has the authority to do so by furnishing the appropriate credentials to the client. In our current prototype, X509 certificates are used for authentication and KeyNote credentials are used for authorization.

The connecting WebCom client establishes an SSL connection with the master, and in the process, the master presents an appropriate X509 certificate chain to its public key K_{master}. If the protocol run is successful then the client knows that it is communicating with K_{master}. The master schedules operation op to the client over the SSL connection and sends the necessary credentials that prove that K_{master} is authorized to schedule op. The WebCom client uses KeyNote to determine if the schedule is authorized, given the credentials of the master.

When a WebCom client uses KeyNote to determine whether it is safe to execute a scheduled operation or action, it must specify the circumstances of the query as an *action attribute set*. This is a list of attribute names and value pairs that are agreed between the application writers and the writers of the related credentials. In addition to the existing attribute `app_domain`, two action attributes are proposed to identify the circumstances of a WebCom KeyNote query:

- Attribute name `ROLE`, which can have, for example, values `schedule` or `execute`, indicating whether the principle is authorized to schedule or execute.

- Attribute name `NODE`, which can have values identified by the operations that can be scheduled/executed.

A standard code-signing approach such as that used by Java [8], would not provide sufficient security infrastructure support for WebCom. A WebCom master schedules more than just code: it schedules mobile computations that are comprised of tasks and their inputs. A client must determine whether it should trust the master for the task it schedules along with the inputs to the task that the master provides and the subsequent output that is returned to the master.

4.2. WebCom Client Authorization

WebCom client authorization is done by the master to determine whether the client has the necessary authorization to execute certain operations whose inputs or outputs may be security critical. In the current prototype, X509 certificates are used for client authentication and KeyNote credentials are used for authorization. Master certificates and credentials are used as in the previous section. However, clients must now present their credentials when the secure connection is first established if they wish to have operations scheduled to them.

A WebCom master maintains a list of clients that are currently connected and available to be scheduled operations. Before scheduling an operation to a client, the master must determine if the client is authorized to execute the operation. This can done by the master making a query to KeyNote with an action attribute set based on attributes `ROLE` and `NODE`, and by the action autoriser which is the client key. If the client is not authorized to execute any of the operations scheduled by the master then the SSL connection can be terminated.

EXAMPLE 5 A WebCom master (`Kserver`) running on a trusted server is authorized to execute an order operation and schedule its components (Example 3). We assume that the `order` operation may have been scheduled to `Kserver` from some other WebCom master.

```
Authorizer: "Kwebcom"
licensees:  "Kserver"
Conditions: app_domain=="WebCom"
        && ((ROLE=="execute" && NODE=="order")
          ||(ROLE=="schedule"
             && (NODE=="prop" || NODE=="OK")));
```

We assume that a suitable X509 certificate for `Kserver` is also available. When an operation such as `prop` is scheduled to a WebCom client by an authenticated `Kserver`, then the client queries KeyNote with action attribute set: [`app_domain`←`"WebCom"; ROLE`←`"schedule" NODE`←`"prop"`], to determine whether this request from `Kserver` is authorized.

WebCom clients Alice (`Kalice`) and (`Kbob`) are authorized to execute `prop` and `issue` operations, respectively.

```
Authorizer: "Kwebcom"
licensees:  "Kalice"
Conditions: app_domain=="WebCom"
  && _ACTION_AUTHORISER=="Kalice"
  && ROLE=="execute" && NODE=="prop";
```

```
Authorizer: "Kwebcom"
licensees:  "Kbob"
Conditions: app_domain=="WebCom"
  && ROLE=="execute" && NODE=="OK";
```

The KeyNote reserved attribute `_ACTION_AUTHORISER` represents the principle making the current action request. In the credential above it is done to prevent Alice from further delegating her authorization.

When `Kalice` connects, requesting `prop` operations, she presents her credential(s). Her authorization is validated and the master schedules the execution of `prop` on her client system. Note that in this example, `Kalice` neither offers, nor has the authority, to execute `issue`. Similarly, Bob connects, offering to execute `issue`, which is scheduled. The WebCom master uses the same SSL secured connection to send operations to execute, and receive results from, the WebCom clients.

This is a example of a simple static separation of duty policy, provided by a combination of credentials and Condensed Graphs. We are exploring how dynamic separation of duty using KeyNote support for credentials specifying threshold signatures might be transparently supported within our framework. △

5. IMPLEMENTATION

To date, we have integrated the KeyNote trust management scheme described by this paper into the existing Java-based WebCom system described in [12]. This version of WebCom supports condensed graphs built in terms of both Java and/or Microsoft COM objects.

A distributed application is constructed as a condensed graph of components using WebCom's Integrated Development Environment (IDE). Figure 2 provides a snapshot of the development of the `order` application described in Example 3. In this case the implementation of Component `prop` uses a specially formatted Microsoft Excel® spreadsheet to capture order details; component `issue` displays the order details for validation and, if validated, prints and issues the order.

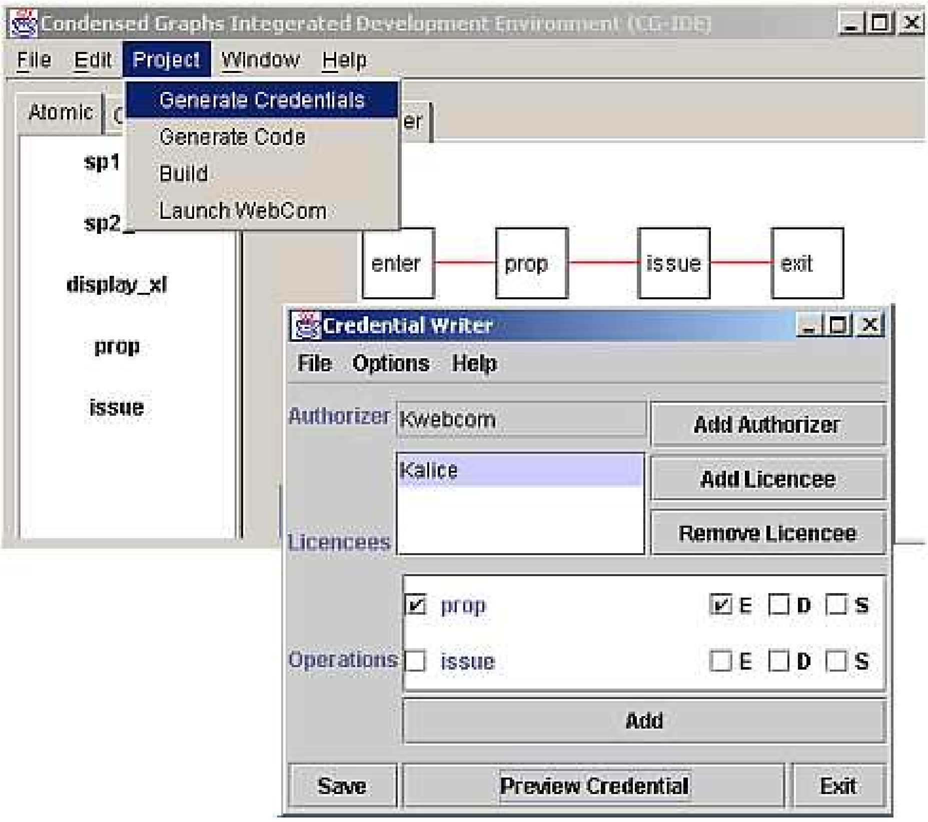


Figure 2. Developing a Secure Application

The WebCom IDE was extended to include a a simple credential writer that interrogates the condensed graph currently open and provides a credential template. This template allows an authorizer to delegate to a licensee, authorization to execute (E), further delegate (D) or schedule (S) the graph components. Additional conditions to be included in the credential may also be specified. The credential tool may also run as a stand-alone application, facilitating further delegation.

The integration of trust management into WebCom was straightforward and the current prototype uses existing packages where possible. The JCSI [9] package provides cryptographic X509 certificate and SSL support. KeyNote credentials are constructed using the keys extracted from their corresponding X509 certificates. The current reference implementation for KeyNote is implemented in C. The Java Native Interface is currently used to provide a Java API to KeyNote. It is planned to eventually replace this with a Java implementation of the Keynote Interpreter.

6. DISCUSSION

KeyNote credentials are used to determine the authorization of X509 authenticated SSL connections between WebCom masters and clients. Client credentials are used by WebCom masters to determine what operations the client is authorized to execute; WebCom master credentials are used by clients to determine if the master had the authorization to schedule the (trusted) mobile-computation that the client is about execute; this authorization covers not just the scheduled code, but also its input data and output.

The relationship between the WebCom trust management system and local operation system protection mechanisms is not addressed in this paper. For example, in what protection domain should a client execute an operation that has been scheduled by an authorized master? In [6] we propose an abstract protection model for Condensed Graphs; our integration of KeyNote into WebCom can be regarded as one step towards implementing this protection model.

We use KeyNote to provide trust management for WebCom because of its ability to support expressive and fine-grained authorization and delegation policies. KeyNote Credentials are typically short, yet expressive, and are, therefore, not expected to contribute significantly to the communication overhead between clients and masters. For the purposes of illustration, only simple examples are presented in this paper. The reader is referred to [3] for a discussion of the expressiveness of KeyNote.

Secure WebCom may be used to provide complete separation between a distributed application and trust management. We have not yet considered PKI issues such as how best to control the dissemination of KeyNote credentials, revocation, and so forth. However, since our architecture uses X509 certificates for authentication then we expect that their associated PKI's should a degree of support for issues such as key revocation. These are topics for future research

The current implementation described in this paper expresses security at the granularity of operation identifiers such as `prop`, `order`, and so forth. However, we expect that it will be straightforward to generalize this to include reference to the attributes used to pass input and output to and from these operations. During execution, the nodes in a Condensed Graph are represented as a triple: input parameter bindings, output binding and the operation applied. These attributes can be referred to within credentials and form part of the action attribute set during a KeyNote query or search. With such an extension, Example 3 could be generalized to support credentials that authorize Bob to validate orders up to some limit.

WebCom masters may promote clients to become client-masters by passing condensed nodes, the components of which are subsequently scheduled by clients of the promoted client [11]. We are currently extending the WebCom trust management architecture to exploit promotion. This uses credentials to control whether masters have the authority to promote clients, and whether clients, in turn, have the authority to be promoted. Writing credentials that specify this is straightforward, and can be done by enlarging the domain of attribute `ROLE` to include values `promoter` and `promotee`. In addition to promoting a client, the master must write the necessary credentials to delegate to the promoted client the authority to schedule the operations it will be managing.

7. CONCLUSION

In this paper we consider how KeyNote can be used to help secure the distribution of application components by the WebCom system. With this approach, application developers need not intertwine their application code with security-critical calls to the trust management system. Functional components such as `prop` and `issue` (Example 3) can be coded independently of security concerns; security concerns are expressed independently in terms of a credential based policy, and the Condensed Graph specifies the synchronization and control concerns between these components. Only operation identifiers couple these concerns. This means that applications may be regarded as untrusted: the security-critical WebCom scheduler acting as a security wrapper for operations and using the trust management system to help decide how to distribute, control and synchronize applications. We like to think of this as a security glue for mobile components.

Acknowledgments

We would like to thank the WebCom developers: Dave Power, James Kennedy, Keith Power and Colm Dineen, for providing access to the WebCom system and and patiently answering our many questions on its design and implementation. This research was supported by the Enterprise Ireland National Software Directorate.

References

[1] Apache-ssl release version 1.3.6/1.36. Open source software distribution. `http://www.apache.org`.

[2] M Blaze et al. The keynote trust-management system version 2. September 1999. Internet Request For Comments 2704.

[3] M Blaze et al. The role of trust management in distributed systems security. In *Secure Internet Programming: Issues in Distributed and Mobile Object Systems*. Springer-Verlag

Lecture Notes in Computer Science, 1999.

[4] M. Blaze, J. Ioannidis, and A.D. Keromytis. Trust management and network layer security protocols. In *Security Protocols International Workshop*. Springer Verlag LNCS, 1999.

[5] S.N Foley. A kernelized architecture for multilevel secure application policies. In *European Symposium on Research in Security and Privacy*. Springer Verlag LNCS 1485, 1998.

[6] S.N. Foley and J.P Morrison. Computational paradigms and protection. In *ACM New Computer Security Paradigms*, Cloudcroft, NM, USA, 2001. ACM Press.

[7] S.N. Foley, T.B. Quillinan, J.P. Morrison, D.A. Power, and J.J. Kennedy. Exploiting KeyNote in WebCom: Architecture neutral glue for trust management. In *Fifth Nordic Workshop on Secure IT Systems*, Reykjavik, Iceland, Oct 2001.

[8] L. Gong et al. Going beyond the sandbox: An overview of the new security architecture in the java development kit 1.2. In *USENIX Symposium on Internet Technologys and Systems*, pages 103–112, 1997.

[9] DSTC Security group. JCSI Java Crypto and Security Implementation. See `http://security.dstc.edu.au`.

[10] C.V. Lopes and K.J. Lieberherr. Abstracting process-to-process relations in concurrent object-oriented applications. In *European Conference on Object-Oriented Programming (ECOOP)*. Springer Verlag LNCS 821, 1994.

[11] John P. Morrison and David A. Power. Master promotion and client redirection in the webcom system. In *PDPTA, Las Vegas USA*, 2000.

[12] J.P. Morrison, D.A. Power, and J.J. Kennedy. A Condensed Graphs Engine to Drive Metacomputing. Proceedings of the international conference on parallel and distributed processing techniques and applications (PDPTA '99), Las Vagas, Nevada, June 28 - July1, 1999.

[13] J.P. Morrison and M. Rem. Speculative computing in the condensed graphs machine. proceedings of IWPC'99: University of Aizu, Japan, 21-24 Sept 1999.

[14] R Rivest and B Lampson. SDSI - a simple distributed security infrastructure. In *DIMACS Workshop on Trust Management in Networks*, 1996.