Market Requirements Document

Feature Name: Encrypted Pages/Objects

Version: 2 **Date Submitted:** 01/31/05
Version: 1 Date Submitted: 01/14/05

Completed By: Leon Guzenda

Description of the Problem

This MRD is a part of a strategy aimed at producing a secure version (Objectivity/Secure) of Objectivity/DB. There was an ITSEC C2-compliant version of Objectivity/DB at Release 2, but it was withdrawn before deployment because the costs of certifying it were too high once the project that demanded it had been canceled.

The only standard security mechanism that is currently supplied by Objectivity is the GRANT/REVOKE mechanism within SQL++, which is not applied by the other language interfaces. SLAC introduced a Generalized Security Architecture [GSA] mechanism within the OOFS interface, but it has only seen limited deployment by Objectivity personnel.

A secure, object server architecture is deprecated because it would remove our competitive advantages over other central server products, such as Oracle and Versant.

Description of the Requested Feature

Introduction

Objectivity/Secure should add at least the following mechanisms:

- Authentication of client identities
- Security administration
- Encrypted pages or objects.

Authentication

The mechanisms discussed in this document assume that the identity of a client can be reliably authenticated. Industry standard authentication mechanisms, such as Kerberos, should be employed wherever possible. Installations should be able to supplement or replace the standard Objectivity authentication mechanism(s) to suit local needs.

Security Administration

There are many security models and whatever we deliver needs to provide hooks for incorporating local mechanisms. Our standard offering should, at least, provide tools for defining client users, groups, organizations and roles. For example, user “Jane” may be in the “Engineering”, “QA” and “IT” groups. “Jane” may also be assigned the roles of “Engineer” and “DBA”.

The standard offering should also define a set of target secured items, a set of secured operations and a default matrix of relationships between the items, operations and clients. For example, databases might only be erasable by DBAs. There should also be tools for Security Administrators to modify the matrix of relationships. For example, only a “Customer Support Engineer” might be able to create and modify objects of class “SPR”.

A fully secure product must also provide tools for logging all changes to the security domain and for monitoring attempts at security violations. We have a full Release 2 System Requirements Specification for these tools, but it will need to be updated to match current product specifications and Government regulations.

Page or Object Level Security

Tagging individual objects with a security label can be prohibitively expensive. Labels in some domains may be 128 bytes long, which is an absurd overhead for small objects. An alternative mechanism could be to use object or page level encryption. Page level encryption may be harder to implement than object level encryption, because objects will have to be moved to another page if their security classification changes. On the other hand, some security regimes will demand that their be no mixing of objects at different security levels within the same physical location (page, file, machine etc.). Object level security could be implemented alongside of, or instead of, file level security..

A client would authenticate itself (using Kerberos or an equivalent protocol) during the “open federated database” operation. All subsequent communications with servers would also use Kerberos. An authenticated client would then be given a public key to use in all subsequent encryption/decryption operations. The key would only work for that client. Almost all relevant operations on an object ultimately involve opening or closing the object and that functionality is constrained to very few places in the kernel code. The encryption and decryption would occur at those places. Each object would contain a four byte security tag that would identify the exact set of security rules that apply to it.

If there is no physical separation of objects at different security levels then there is still an opportunity for a malicious client to legitimately open an object and then manipulate the virtual address information to corrupt the other objects in that page. We could protect the other objects against that threat by including a checksum in the object and page. There

would have to be some additional intelligence in the AMS to verify that all objects have correct checksums before writing pages back to the database.

Feature Summary

a) Objectivity/Secure will add interfaces and tools to support a Security Administrator role.

b) A Security Administrator will be able to define organizational structures, including individuals, groups and roles and assign metadata, data and security information access privileges to the nodes within the organizational structure.

For example, everybody within Marketing might be allowed to read, but not update or delete, Product_Description objects. The VP Marketing would be allowed to create, update and delete Product_Description objects and to assign that privilege to one or more individuals within the Marketing organization, but not to others.

c) The Objectivity kernel and servers will prevent unauthorized users from reading, updating, creating or deleting secured data types or instances. “Updating” is used broadly here to include actions such as adding or removing associations, names and index entries.

d) It must be impossible for a client to obtain or modify the “clear” content of data or metadata that it is not authorized to access.

e) The rules governing the security model should be encapsulated within user replaceable hooks so that individual sites can enforce their own security regimes.

f) There must be minimal impact on product performance and storage overheads.

g) The mechanism must complement and be compatible with the current GRANT and REVOKE mechanisms in SQL++.

Part of an existing feature or does it require another feature, if so, which one?

Objectivity/Secure should be an optional addition to our standard products.

How is this problem being solved now, and why isn't that acceptable?

Current federations can be crudely protected using the operating system file protection mechanisms. Currently this means that a database can be read or write protected against individual usernames or groups of usernames. This mechanism can be made more secure using OOFS, the SLAC GSA hook and Kerberos. At Release 9 we will support container files (or files that hold many containers).

Database or container level security may be adequate for many applications. However, as any client can read the whole of the catalog, it doesn't hide the existence of particular databases or containers from a malicious client. It may be worth changing the catalog operations (such as the iterators that run across all databases or containers) to attempt to open the file before returning database or container objects to the client.

What languages must support this capability?

- C++
- Java
- Smalltalk (later)
- SQL++ (later)

Which platforms must be supported?

- Tier 1 platforms, then others as required.

Do any competitors already have this feature?

- DB2, Oracle and Sybase have equivalent mechanisms, but we also support similar mechanisms in SQL++.

Customers who require this feature

- Federal customers.
- Some financial applications.

Revenue at risk, or which could be won

- The Government and the business community is becoming increasingly worried about security issues. We are debarred from many environments that have to share information with other organizations (rather than being closed systems) because we do not have a secure product.

When is this required?

- Release 10.

Additional Notes

1. We will also need:

- Marketing collateral, including promotion, a special area on our web site and a press release.
- Technical Publications.
- New QA material.
- Peer review from at least two defense contractor customers.

2. Objectivity/Secure could be introduced in two phases:

- Phase 1 - Access Control, with minimum administrative capability. This would include authentication of users and page level encryption.
- Phase 2 - Additional tools to manage keys and to log and monitor security access violations.