CosmoS

Cosmos-B

Security

*Cosmos-B is a managed code operation system designed to run on the Cosmos kit. This document describes the security system. .*

SECURITY

Ben KloosterMan

# 0.Overview

So the Cosmos-B Kernel is a bunch of assemblies and we don’t have HW protection how do we get security , Tannenbaum even stated pick an OS with clear security boundaries?

Firstly we do provide a VERY explicit boundary in the form of Allocator Domains [Check] however they are enforced through software. Applications run in this explicit Sandbox and are limited everything else ( ie the preloaded DLLs that run the system) is the Kernel and is trusted.

Secondly the service/kernel ring 0 boundary is an illusion.. It was true 20 years ago but these days there is so much communication going on between services, the Kernel and user applications that those ring models are no longer appropriate. Serious attempts to subvert the system hence have often subverted a trusted service ( eg File System) or component especially user added trusted services eg the recent Flash hack .

Access is controlled via capabilities.

# 1.What are Capabilities

By definition a Capability is un-forgeable reference to a system resource. Since it is un-forgeable it provides direct access to such a system resource without checks. Furthermore it allows arbitrarily fine grained security and hence makes it easier to follow the Principal of Least Authority (POLP/POLA).

POLP requires that in a particular abstraction layer of a computing environment, every module (such as a process, a user or a program on the basis of the layer we are considering) must be able to access only such information and resources that are necessary to its legitimate purpose.

Capabilities are analogous to Unix file descriptor.

Most OS today run with a large amount of Ambient Authority eg all the files an application can open is limited to the user level this is a massive security risk since a hacked app now has access to everything.

Note than an ACL could go down to the application level ( and in fact does for in Windows there is an ACL for which application can access ports) but in practice with an ACL system this creates a significant administrative burden.

See <http://www.eros-os.org/essays/capintro.html> for a better intro.

# 2.COSMOS-B Capabilities

We use a very simple approach to Capability security that is successfully used in many software development languages that is Object references. If you don’t have a reference to the object you have to get it from somewhere or convince the application to give you a reference. We see this reference as a key and the key can be copied and passed around. This is a capability.

Previous systems to use Capabilities such as the AS400 line , EROS etc allowed native languages and hence these capabilities had to be tightly controlled by the OS else other app can just grab the memory and copy the code. However with Cosmos-B as we do not allow access to the machine and use a strongly typed language we do not have this issue.

The reference capabilities we will use will be simpler and faster than modern ACL systems. It is worth noting that the introduction of Capabilities requires the development API to change but since we need to do that anyway it is not a serious disadvantage.

Since we use the language to enforce this, programmers can achieve a higher level of granularity since the reference will only be available to a small subset of the total application code and the security can in some cases be limited to the object.

# 3. KEYRING

A Keyring is used to store capabilities this is an encrypted file. Each SIA has a separate keyring for each user that may invoke it. The SIA keeps a separate Keyring for each user that runs the program as well as a global one for all users ( eg to access a system log file) .

KeyRings

SIA /Global

SIA /User

User

UserGroup

Machine

If an SIA doesn’t have rights it will search and prompt the user whether it is ok to copy them ( ie always allow , allow once , deny) . When an application is installed it will make a list of Capabilities it needs access to for the user to approve just these will be copied from the User or UserGroups.

# 4. PERSISTING AND LOADING Capabilities

Primary Capabilities are persisted even if we use a non persistent heap. Capabilities are serialized and encrypted. Capabilities implement ICapability which in turn implements PropertyChanged [Fix]

When an application receives a capability it is immediately saved in the Keyring when it is required a wrapper object will request the capability. The wrapper object will request from the TCB to instantiate the Capability, the TCB will do this using the type of the service associated with the capability. Note they only need to be instantiated after the app has been shutdown.

XXX Garbage

# 5. APPLICATION CREATED CapabilitieS

Applications especially services can create their own capabilities by creating class which implement ICapability. Note these capabilities cannot reside in user created globally shared DLL only exe and non shared DLLs. The Kernel will call the relevant code when it needs to deserialize a capability.

# 6. REVOKING CapabilitieS AND COnFINEMENT

Revoking capabilities is quite simple see http://srl.cs.jhu.edu/pubs/SRL2003-02.pdf. In our cases we provide IManagebleCapability as an interface. Implementers will return a manageCapability instance when requested this needs to be stored when created as the creator may not have access later on.

Implementers then modify the capabilities methods when the IManageCapability requests it eg the constructor can fetch the ManageCapability and see if the class is still valid. The Property changed event can also check this. Note this allows us to immediately change capabilities and then later re-allow them.

For confinement factories can be used <http://www.erights.org/elib/capability/factory.html> all we really need to do is test that the factory is of form ICapabilityFactory and the trusted factory can work with capabilities from multiple sources without exposing them.

# 6. Capabilities across NETWORKS AND DIRECTORY SUPPORT

TODO.

# 7. USER USE OF Capabilities

It is likely that capability based systems will be used differently , why central control and granting Capabilities is certainly still possible it is likely that users will pass the security they need directly to their friends or colleagues rather than a system admin. Eg after the Admin creates the account and home directory her colleagues will give access to group directories, files and applications they need to do their work.

# 3.USER SECURITY OVERHEAD AND HOW TO MANAGE

The emphasis for security is on the application installation process which is often done by the administrator. When an application installs it provides a concise list of access it needs which the user can approve or deny it is recommended that this list be a minimum list though a secure options could be viable for administrators.

When the application does not have the security required it will check the User , his groups and any global rights on the machine.

If the capability is found in any of the above (except for everyone) then the user will be prompted whether it’s ok and whether to always allow this. If the users always allow this the Capability will be copied to the Application. Everyone (Machine) is always approved silently.

Note we desire to eliminate this prompting and a policy setting will allow silent approval, always deny or prompt.

Emphasis on security is on users assisting each other rather than system admins , users can share their rights to each other though the system works well with centrally managed.

# 3.IMPACT OF NO COMMAND LINE/CONSOLE

The fact the OS has no command line removes a massive amount of holes. We no longer need to convert arbitrary string values into file names and into capabilities or even have user context for the security.

Here applications directly pass references to each other. For example explorer has a browse capability when a user clicks on a document this file with browse capability is passed to the wordprocessor when this request fails the system will start looking for escalatable capabilities of the same Type ( eg FileCapability) and find the word processors read/write capability which it will then try.

This means that applications ( with per user settings) rather than the user becomes the most important security aspect , eg word has access to all word documents ( .doc and .docx) in the users document and share directory but nothing else. This significantly reduces ambient authority since even to use user privilege ( yet alone admin group) will result in a prompt whether to allow the application access.

If the admin installs it he could even give the user no R/W rights to his home directory ( only directory browse) but give the word Application access (which would get the capabilities from the admin). Obviously even if a browser can break its sandbox it still couldn’t access any such files.

Some applications like compilers do not use Path and will mix configuration with settings eg setting the lib Directory tree will set the libs for the compile and add the Capabilities to the compiler.

# 3. SOME CapabilitIES IN COSMOS-B

Primary Capabilities are persisted. Secondary’s (\*) are not (unless part of a heap persist) and are runtime created.

File System

* File
* Directory
* DirectoryTree

Provides access to every file in the directory tree . Note directories can be set to deny these objects focing individual access. These can be by file type eg word document , give rw access to all word documents in this tree.

* DiskQuota

Process Control

* STPTree\*
* CapabilityStore
* STP\*
* AllocatorDomain
* SoftwareIsolatedApplication (SIA)

Resource Control

* CPU Resource
* MemoryRegion
* Disk resource
* Network resource

HW Control

* DMA
* Read/Write to IO port Region
* PowerManagement

System Control

* ThreadScheduling
* CapabilityManagement
* DeviceDriverControl
* ServiceControl

IPC

* IPCQueue
* IPCPipe\*

Security

* User\*
* Group\*
* Machine\*

NetworkPort ( Tcp stack)

# SECURITY POLICY

Encompasses setting and configuration Password strengths‘

# GRANTING OF CAPABILITIES

Capabilities are granted by the current holder of the capability and access is granted through the transfer of a capability through IPC.

FileDialog can transfer the users rights directly to the application as the users action automatically grant use.

Installer/Configurer – this reads the metadata and will also transfer the users rights to the application it can cover many things such as port usage. When the user wishes to reconfigure the app he invokes the configure..