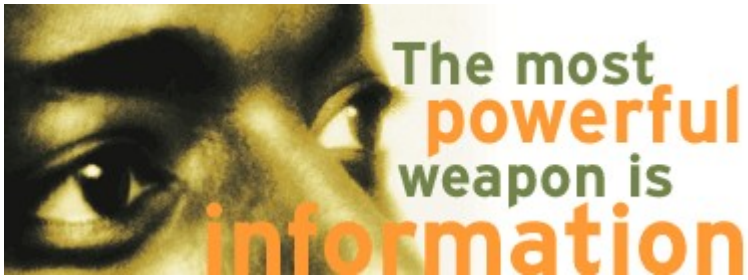
Martus Testing Project

Deliverable 4



November 14, 2013

Project Name: Martus: Global Human Rights Abuse Reporting System.
Call and Contact: charlestonmartustesting@gmail.com
martus website: http://martus.org
Project Duration: 9/3/2013 – 11/21/2013 (2 months and 18 days)
Project Roles:

- Planner: Jason Wilson
- Liaison: Steven Pilkenton
- Researcher: Bobby Jenkins
- Organizer: Tomoko Goddard

4. Testing Experience

4.1 New Progress

We've added 9 more test cases since our last report. These newly added test cases were derived from already tested methods as well as newly created ones. The script has been adapted to handle compilation and runtime errors without terminating execution. A build script was also added to the script's directory so that the user has the option to clean build the project under test. Thee test cases involving the methods “encrypt()” and “decrypt()” were combined as one as were the “writeRecord()” and “discardRecord()” methods.

4.2 Group Experiences

The group has continued to meet twice a week in an effort to finalize the deliverable and implement test cases. Meetings were basically a progress report or whats been done and some code consolidating here when needed. We intend to begin editing the rough draft of our final deliverable and altering code of the framework on an as needed basis.

4.3 Test Cases

Test Case	Requirement	Method Tested	Input Used	Output Expected
#1	Application must be able to encrypt data	common.crypto.MartusSecurity.encrypt()	A string to encrypt	String encrypted = true
#2	Application must be able to decrypt encrypted data	common.crypto.MartusSecurity.decrypt()	./temp/encryptedBytestring.txt	Decrypted bytestring = A string to encrypt
#3	Username must not match password	client.core.MartusUserNameAndPassword.validateUsernameAndPassword()	--username=Password --password=Password	org.martus.common.Exceptions$PasswordMatchedUserNameException
#4	Password must be at least 8 characters	client.core.MartusUserNameAndPassword.validateUsernameAndPassword()	--username=Username --password=1234567	org.martus.common.Exceptions$PasswordTooShortException
#5	Username must not be blank	client.core.MartusUserNameAndPassword.validateUsernameAndPassword()	--username= --password=Password	org.martus.common.Exceptions$BlankUserNameException
#6	Testing if setting a database key to draft works after its been initialized to "sealed" by default	common.database.DatabaseKey.isDraft()	true	true
#7	Testing if database keys are set to "sealed" by default	common.database.DatabaseKey.isSealed()	true	true
#8	Testing for difference in keys set to sealed vs keys set to drafted	common.database.DatabaseKey.isSealed()	true	false
#9	Application must be able to store user data	common.database.FileDatabase.createRecord()	--entries-to-create=5	Database record count = 5
#10	Application must be able to delete stored user data	common.database.FileDatabase.discardRecord()	./temp/databaseKeys.txt	Database record count = 0
#11	Application will try allowed secure ports until a connection is established	clientside.ClientSideNetworkHandlerUsingXmlRpc.callServer()	--good-port-middle=7	Number of tried ports = 3
#12	Application will try only enough secure ports to establish a connection	clientside.ClientSideNetworkHandlerUsingXmlRpc.callServer()	--good-port-first=7	Number of tried ports = 1
#13	Application will try all available ports to establish a connection	clientside.ClientSideNetworkHandlerUsingXmlRpc.callServer()	--fail-all=true --good-port-first=7	Number of tried ports = 5
#14	Application must create a universal ID from account and local IDs	common.packet.UniversalId.createFromAccountAndLocalId()	--account-id=someAccountID --local-id=someLoca	Account ID = someAccountID & Local ID =

			lID --prefix= --from-string=false	someLocalID
#15	Application must create a universal ID from account ID and a prefix	common.packet.UniversalId.createFromAccountAndPrefix()	--account-id=someAccountID --local-id=someLocalID --prefix=D- --from-string=false	Account ID = someAccountID & Local ID length = 26
#16	Application must create a universal ID from the string representation of account and local IDs	common.packet.UniversalId.createFromString()	--account-id=someAccountID --local-id=someLocalID --prefix= --from-string=true	Account ID = someAccountID & Local ID = someLocalID
#17	Application must create a universal ID from account and local IDs	common.packet.UniversalId.createFromAccountAndLocalId()	--account-id=someAccountID --local-id=someLocalID --prefix= --from-string=false	Account ID = someAccountID & Local ID = someLocalID
#18	Application must create a universal ID from account ID and a prefix	common.packet.UniversalId.createFromAccountAndPrefix()	--account-id=someAccountID --local-id=someLocalID --prefix=D- --from-string=false	Account ID = someAccountID & Local ID length = 26
#19	Application must create a universal ID from the string representation of account and local IDs	common.packet.UniversalId.createFromString()	--account-id=someAccountID --local-id=someLocalID --prefix= --from-string=true	Account ID = someAccountID & Local ID = someLocalID
#20	Passwords must be at least 15 characters with at least 2 non alphanumeric characters to be considered "strong"	client.core.MartusUserNameAndPassword.isWeakPassword()	--password=notStrong&^	Weak password
#21	Passwords must be at least 15 characters with at least 2 non alphanumeric characters to be considered "strong"	client.core.MartusUserNameAndPassword.isWeakPassword()	--password=longButNotStrong&	Weak password
#22	Passwords must be at least 15 characters with at least 2 non alphanumeric characters to be considered "strong"	client.core.MartusUserNameAndPassword.isWeakPassword()	--password=longAndStrong&^	Strong password
#23	once a variable of type FieldSpec is given attributes, MartusField must inherit them (.getTag() field checked here)	common.fieldspec.MessageField.getFieldSpec()	tag	true
#24	once a variable of type FieldSpec is given attributes, MartusField must inherit them (.getLabel() checked here)	common.fieldspec.MessageField.getFieldSpec()	label	true
#25	once a variable of type FieldSpec is given attributes, MartusField must inherit them (FieldSpec and MartusField are compared to each other here)	common.fieldspec.MessageField.getFieldSpec()	compare	true