Policy Component Reference Manual

Generated by Doxygen 1.4.7

Thu Sep 16 11:29:41 2010

Contents

1 Policy Component Class Index

1.1 Policy Component Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

2 Policy Component File Index

2.1 Policy Component File List

Here is a list of all documented files with brief descriptions:

3 Policy Component Class Documentation

3.1 CDigiSig Class Reference

`#include <digiSig.h>`

Public Member Functions

- int **Sign** (const std::string &sPrivateKey, const std::string &sCommand, uint8_t *sig_buf, unsigned int &iSigLen)
- bool **Verify** (const std::string &sPublicKey, const std::string &sCommand, const uint8_t *sig_buf, unsigned int iSigLen)
- bool GetIssuerName (const std::string &sPublicKey, std::string &sIssuerName)
- bool VerifyByCA (const std::string &sCert, const std::string &sFile)
- bool GetSubjectName (const std::string &sPublicKey, std::string &sSubjectName)
- int SignSMIME (const std::string &sCert, const std::string &sPrivateKey, const std::string &sCommandFile, const std::string &sOutFile, const std::string &sTo, const std::string &sFrom, const std::string &sSubject, bool bAddNoCert=false)
- int VerifySMIME (const std::string &sFileToVerify, const std::string &sCAChain, const std::string &sDataFile, const std::string &sSignerCert=std::string())
- int ParseMIME (const std::string &sFile, ReqType req, std::string &sOut)
- int **GetAllIssuer** (const std::string &sCert, std::set< std::string > &sIssuerSet)
- int **GetSenderCertificate** (const std::string &sMsg, const std::string &sFileToSave)
- int **save_certs** (const char *signerfile, STACK_OF(X509)*signers)
- void **print_file** (const std::string &sFileName)

Private Member Functions

- bool **GetParam** (const std::string &sPublicKey, std::string &sParamName, const ReqType eReq)
- int **GetParamsFromPKCS7** (const PKCS7 *p7, ReqType req, std::string &sOut)

3.1.1 Detailed Description

Main interface class between OpenSSL library and the Policy component. Provides functionality for SMIME authentication, certificate parsing, CA chain verification etc.

3.1.2 Member Function Documentation

3.1.2.1 bool CDigiSig::GetIssuerName (const std::string & *sPublicKey*, std::string & *sIssuerName*)

Extracts the issuer name from X509 certificate.

3.1.2.2 bool CDigiSig::GetSubjectName (const std::string & *sPublicKey*, std::string & *sSubjectName*)

Extracts the Subject name from the SMIME file.

3.1.2.3 int CDigiSig::ParseMIME (const std::string & *sFile*, ReqType *req*, std::string & *sOut*)

Parse a SMIME file for different field such as TO, FROM, SUBJECT etc.

3.1.2.4 int CDigiSig::SignSMIME (const std::string & *sCert*, const std::string & *sPrivateKey*, const std::string & *sCommandFile*, const std::string & *sOutFile*, const std::string & *sTo*, const std::string & *sFrom*, const std::string & *sSubject*, bool *bAddNoCert* = `false`)

Signs a given file sCommandFile with private key using SMIME format and output the signed file in sOutFile. Put sTo, sFrom and sSubject in the SMIME message. If bAddNoCert is false, then attach the sender's certificate sCert in the SMIME message.

3.1.2.5 bool CDigiSig::VerifyByCA (const std::string & *sCert*, const std::string & *sFile*)

Verifies a given certificate sFile by a CA chain hierarchy as specified in sCert.

3.1.2.6 int CDigiSig::VerifySMIME (const std::string & *sFileToVerify*, const std::string & *sCAChain*, const std::string & *sDataFile*, const std::string & *sSignerCert* = `std::string()`)

Verifies a given file sFileToVerify in SMIME format with the CA chain sCAChain. Extracts the data and write in sData file. Extracts the sender's certificate and write in sSignerCert file.

The documentation for this class was generated from the following files:

- digiSig.h
- digiSig.cpp

3.2 CGossipComm Class Reference

`#include <gossip_comm.h>`

Public Member Functions

- CGossipComm (std::string sFileName)
- int Run ()
- void StartNetworkThread ()
- int SendCommand (const std::string &sCommand)
- int SendCommand ()
- int FindFile (const std::string &sFile)
- bool LoadSubscription ()
- const int * GetSocketPair ()

Private Member Functions

- int **GetSockFd** ()
- void **OnCommandReceived** (byte *buffer, size_t length)
- void **CheckEvent** (int iFd)
- int SendToGossip (const std::string &sFileName, uint8_t module)
- int CreateSocket ()
- void FindBootStrapNode ()
- int ParseConfig ()
- std::string **GetString** (std::vector< std::string > &vConf, const std::string &sLine)
- std::string **GetTwoSidedString** (std::vector< std::string > &vConf, const std::string &sLine, std::string &sLeft)
- void **StrTrim** (std::string &str) const

- bool VerifySignature (const std::string &sFileName)
- int PublishCert (const std::string &sCert)
- std::string CopyFileToKeyDir (const std::string &sFile)
- std::string CopyFileToCommandDir (const std::string &sFile)
- std::string **MakeAbsPath** (const std::string &sPath)
- void PublishCACerts ()
- std::string **GetCAName** (const std::string &sCert)
- bool **FileExistsInKeyDir** (const std::string &sFileName)
- bool VerifyCA ()
- bool CollectCerts (std::string &sFileName)
- bool ReverseVerification (bool bVerifyCA=true)
- std::string **GetPath** (const std::string &sFileName)
- std::string **GetCertificateName** (const std::string &sCert)
- bool IsSignedByHigherCA (const std::string &sIssuer)
- void **DeleteFileFromCache** (const std::string &sPath)
- bool IsSubscribed (const std::string &sFrom)
- void GossipAndExecute (const std::string &sFileName, const std::string &sFileToExecute)
- bool **IsFirstTimeReceived** (const std::string &sFile)
- int **InitSocketPair** ()
- void **ShutdownNetworkThread** ()
- void BootStrapKadReq (int ip, int port)
- uint8_t **CheckFileType** (const std::string &sFile)
- int **GetKadBootIPandPort** (const std::string &sFile, int &ip, int &port) const
- void **GetAddrPortFromString** (const std::string &sLine, int &ip, int &port) const
- bool **CheckSelfReceived** (const std::string &sFrom) const
- int ExtractRunLevel (const std::string &sFrom) const
- bool CheckSenderIntigrity (const std::string &sFile, const std::string &sCert)
- int LoadPolicyProfile ()
- uint8_t **CheckModuleType** (const std::string &sFile)
- int ReportFeedback (const std::string &sFileName, bool bSuccess)

Static Private Member Functions

- static void * ThreadCallback (void *arg)

Private Attributes

- std::string m_sConfigFile
- pthread_t **m_Thread**
- int **m_iSockFd**
- uint32_t m_uGossipIP
- uint16_t m_uGossipPort
- uint32_t **m_uClientIP**
- uint16_t **m_uClientPort**
- std::string **m_sTestCertificate**
- std::string m_sKeyDir
- bool **m_bSignEnable**
- boost::uuids::uuid m_ClientName
- std::string **m_sClientName**
- std::map< std::string, std::string > m_packet

- std::vector< std::string > m_vCAs
- std::vector< std::string > **m_vSubscribed**
- std::string **m_sSubscriptionFileName**
- bool **m_bSubscribedAll**
- std::string m_sTrustedCAName
- std::string m_sTrustedCAPath
- std::string **m_sCommandRepository**
- std::string **m_sLogFileName**
- std::string **m_sCommandFileName**
- std::map< std::string, std::string > **m_cert**
- CDigiSig m_digisig
- std::map< std::string, int > **m_ca_in_file**
- std::set< std::string > **m_issuer_set**
- int m_iTransportProtocol
- std::map< std::string, int > **m_MessageIDMap**
- int **m_iSockPairFd** [2]
- bool **m_bInit**
- int **m_iBootServerIP**
- int **m_iBootServerPort**
- std::map< std::string, uint8_t > **m_recv_file_types**
- bool **m_bBootstrapEnable**
- std::map< std::string, std::string > **m_UserId**
- std::string **m_sDefaultUserId**
- std::string **m_sClientCertDir**
- std::string **m_sPolicyProfile**
- std::map< int, std::string > m_ProfileMap
- std::map< std::string, uint8_t > **m_recv_module_types**
- bool **m_bValidateCert**
- std::deque< std::string > **m_vCert**

3.2.1 Detailed Description

Placement of Policy component in TCP/IP stack

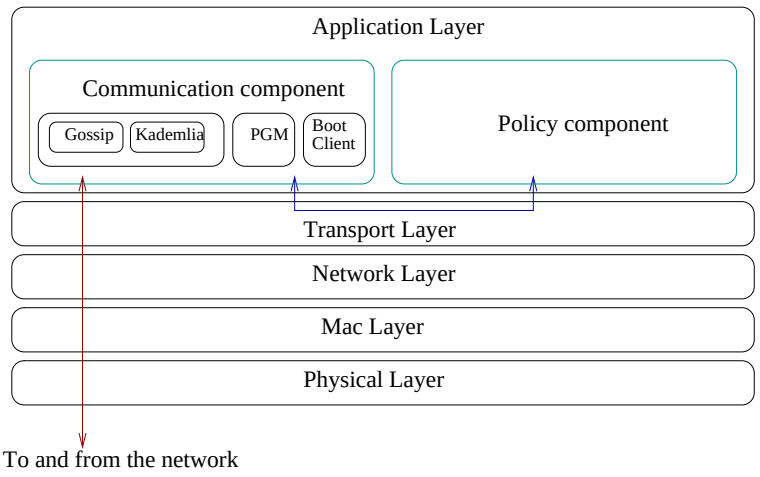

Figure 1: C2 System Architecture

3.2.2 Constructor & Destructor Documentation

3.2.2.1 CGossipComm::CGossipComm (std::string *sFileName*)

Constructor of the CGossipComm class; it takes configuration file name as its argument An example of such configuration file is:

```
#IP address on which the Communication module is running on.
gossip_ip = localhost
#Port number on which the Communication module is running on.
gossip_port = 7772

#ID of the node; it is a UUID.
client_name = e211d574-48ee-492a-b199-ccd337d61101

# Node's IP and port; Not important, leav as they are.
client_ip = localhost
client_port = 0

#Is digital signature enables? It should be 'yes' always.
digital_sign = yes

#path of the certificate cache directory. In this path, Policy component saves
#all certificates it collects from the p2p network.

certificate_cache_dir = shared_dir

#The node's trusted CA file location.
trusted_ca=keys/76005f60-0fd9-4b08-a6ee-70fdad47a840.pem

#The name of the trusted CA.
trusted_ca_name = 76005f60-0fd9-4b08-a6ee-70fdad47a840

#Gossip can be done with TCP and UDP transport protocol. It should be TCP
#always.

tcp_gossip = yes

#The directory where all gossip messages are stored.
command_store = download

#Log file name where the policy component logs everything.
log_file = log.txt

#Name of the subscription file. If "All" string is found in this file, then
#commands from any sender node will be executed. If any node ID is specified
#in this file, then only command from that node ID will be executed.
#The file can contain multiple Node ID.

subscription_file = subscription.txt

#The name of the command file which will be sent for sending command.
command_file_name = command.sh

#Should the node collect bootstrap node's ip:port from bootstrap server?
bootstrap_enable = yes

#bootstrap server's ip address.
bootstrap_server_ip = 130.55.115.201

#bootstrap server's port.
bootstrap_server_port = 75670

#Policy component supports multiple user identity.
# here, list all the user identity.
# user identity is formatted with the name+sensitivity level.
user_id_1 = e211d574-48ee-492a-b199-ccd337d61101+4
```

```
user_id_2 = e211d574-48ee-492a-b199-ccd337d61101+7

#Which id should be the default for sending command?
default_user = user_id_1

#Noed's certificate directory where the private key of the node
#and the trusted certificates are stored.
client_certificate_dir = keys

#Profile file name. In this file name, the sensitivity level and corresponding
#parameters are stored.
policy_profile = profile.txt
```

3.2.3 Member Function Documentation

3.2.3.1 void CGossipComm::BootStrapKadReq (int *ip*, int *port*) [private]

Once received authenticated IP and port from bootstraping server, policy component asks the Communication module for doing Kademlia Bootstrapping.

3.2.3.2 bool CGossipComm::CheckSenderIntigrity (const std::string & *sFile*, const std::string & *sCert*) [private]

Check whether the sender's name in his certificate and his anounced name in sFrom are same.

3.2.3.3 bool CGossipComm::CollectCerts (std::string & *sFileName*) [private]

From a certificate, if the policy component tries to find the issuer certificate. If the issuer certificate is not found in the certificate cache directore, then it calls FindFile(const std::string&) function.

3.2.3.4 std::string CGossipComm::CopyFileToCommandDir (const std::string & *sFile*) [private]

Copies 'sFile' to the command directory. Generally all command files are stored in this directory.

3.2.3.5 std::string CGossipComm::CopyFileToKeyDir (const std::string & *sFile*) [private]

Copies 'sFile' to the Key directory and returns the path of the copied file. The name of the key directory is specified in the configuration file. Key directory generally means a certificate cache directory.

3.2.3.6 int CGossipComm::CreateSocket () [private]

A TCP socket is created to talk to the Communication module.

3.2.3.7 int CGossipComm::ExtractRunLevel (const std::string & *sFrom*) const [private]

Find the sensitivity level from the sender's name.

3.2.3.8 void CGossipComm::FindBootStrapNode () [private]

Bootstrap node's IP and port can be known from the central bootstrap server. To find such a bootstrap IP and port, the function asks the Communication module to communicate with the bootstrap server and receive ip and port.

3.2.3.9 int CGossipComm::FindFile (const std::string & *sFile*)

Asks Communication module for finding sFile using Kademlia protocol.

3.2.3.10 const int* CGossipComm::GetSocketPair () [inline]

A socket pair is created to communicate between the main thread and the network thread. If the main thread exists, it writes shutdown on this socket pair so that network thread shutdowns and thus a graceful shutdown takes place.

3.2.3.11 void CGossipComm::GossipAndExecute (const std::string & *sFileName*, const std::string & *sFileToExecute*) [private]

If verification is successfull, then policy component gossips the command by sending the signed file to the Communication component and Executes the command if the command has arrived for the first time.

3.2.3.12 bool CGossipComm::IsSignedByHigherCA (const std::string & *sIssuer*) [private]

Checks if the Issuer name is the trusted higher CA or not.

3.2.3.13 bool CGossipComm::IsSubscribed (const std::string & *sFrom*) [private]

Checks whether the policy component has subscribed to the sFrom.

3.2.3.14 int CGossipComm::LoadPolicyProfile () [private]

Load the policy profile which gives a map between the sensitivity level and the corresponding parameter.

3.2.3.15 bool CGossipComm::LoadSubscription ()

Loads the subscription file in the run time. The name of the subscription file is specified in the configuration file. The module will execute commands which are from the nodes listed in the subscription file. If "All" is specified in the subscription file, then all commands will be executed regardless of the senders.

3.2.3.16 int CGossipComm::ParseConfig () [private]

Parses the configuration file. Returns -1 if there is an error in the configuration file.

3.2.3.17 void CGossipComm::PublishCACerts () [private]

Asks the Communication module to publish CA certificates. It is usefull to publish any CA certificates.

3.2.3.18 int CGossipComm::PublishCert (const std::string & *sCert*) [private]

Asks Communication module to publish the file sCert in the p2p network.

3.2.3.19 int CGossipComm::ReportFeedback (const std::string & *sFileName*, bool *bSuccess*) [private]

If a certificate is received, it is also sent back to the communication component. this is necessary only for pollution attack robustness technique.

3.2.3.20 bool CGossipComm::ReverseVerification (bool *bVerifyCA* = `true`) `[private]`

After receiving all relevent certificates, it verifies the whole certificate chain. Returns true if verification becomes successfull.

3.2.3.21 int CGossipComm::Run ()

Main method for running this policy module. Returns -1 if it is unable to run for configuration error in the configuration file.

3.2.3.22 int CGossipComm::SendCommand ()

It calls int SendCommand(const std::string& sCommand) by putting m_sCommandFileName as its argument. Returns -1 if fails.

3.2.3.23 int CGossipComm::SendCommand (const std::string & *sCommand*)

Sends the command written the file sCommand to the Communication module. Before sending the command, it signs the command file using SMIME. For signing, private key and the certificate are required which are searched to the location specified in the configuration file. Returns -1 if fails.

3.2.3.24 int CGossipComm::SendToGossip (const std::string & *sFileName*, uint8_t *module*) `[private]`

sFileName file is sent to the Communication module for gossiping. module refers to the corresponding module such as MULTICAST or GOSSIP etc.

3.2.3.25 void CGossipComm::StartNetworkThread ()

Runs the network thread; The thread waits for any communication from the Communication module.

3.2.3.26 void * CGossipComm::ThreadCallback (void * *arg*) `[static, private]`

Main function that is run by the network thread.

3.2.3.27 bool CGossipComm::VerifyCA () `[private]`

Verifies a certificate in the certificate chain. Returns false if varification fails.

3.2.3.28 bool CGossipComm::VerifySignature (const std::string & *sFileName*) `[private]`

Verifies SMIME signature in the sFileName file.

3.2.4 Member Data Documentation

3.2.4.1 boost::uuids::uuid CGossipComm::m_ClientName `[private]`

The node's Identity

3.2.4.2 CDigiSig CGossipComm::m_digisig `[private]`

An object for Openssl interface for SMIME authentication.

3.2.4.3 int CGossipComm::m_iTransportProtocol [private]

Supports both TCP and UDP gossip. If UDP gossip, then does not attach the senders certificate in the SMIME message.

3.2.4.4 std::map<std::string, std::string> CGossipComm::m_packet [private]

A Map for certificate name and its issuer name relationship.

3.2.4.5 std::map<int,std::string> CGossipComm::m_ProfileMap [private]

Maps the sensitivity level and corresponding parameter in the profile.

3.2.4.6 std::string CGossipComm::m_sConfigFile [private]

Configuration file name

3.2.4.7 std::string CGossipComm::m_sKeyDir [private]

Certificate cache directory.

3.2.4.8 std::string CGossipComm::m_sTrustedCAName [private]

Name of the trusted CA.

3.2.4.9 std::string CGossipComm::m_sTrustedCAPath [private]

Path of the trusted CA file.

3.2.4.10 uint32_t CGossipComm::m_uGossipIP [private]

The IP address on which Communication module is listening.

3.2.4.11 uint16_t CGossipComm::m_uGossipPort [private]

The port on which Communication module is listening.

3.2.4.12 std::vector<std::string> CGossipComm::m_vCAs [private]

Vector for CA chain hierarchy.

The documentation for this class was generated from the following files:

- gossip_comm.h
- gossip_comm.cpp

3.3 EventMessage Class Reference

```
#include <EventMessage.h>
```

Public Member Functions

- void SetStatus (int i)
- void SetReasonCode (int i)
- void SetReasonString (const std::string &s)
- void SetFileType (uint8_t t)
- void SetFileSize (int t)
- void SetFileName (const std::string &s)
- void SetFilePath (const std::string &s)
- void SetSearchID (int i)
- int GetStatus ()
- int GetReasonCode ()
- std::string GetReasonString ()
- uint8_t GetFileType ()
- int GetFileSize ()
- std::string GetFileName ()
- std::string GetFilePath ()
- int GetSearchID ()
- void SetRelatedModule (uint8_t t)
- uint8_t GetRelatedModule ()
- virtual void serialize (std::string &sBuf) const
- virtual int deserialize (const std::string &sBuf)

Private Attributes

- int **m_iStatus**
- int **m_iReasonCode**
- int **m_iReasonStringLen**
- std::string **m_sReasonString**
- uint8_t **m_8FileType**
- int **m_iFileSize**
- int **m_iFileNameLen**
- std::string **m_sFileName**
- int **m_iFilePathLen**
- std::string **m_sFilePath**
- int **m_iSearchID**
- uint8_t **m_8RelatedModule**

3.3.1 Detailed Description

An event message class. This class is used to send events in: a socket pair between main thread and the network thread. to communicate between policy component and the communication component. For serialization/deserialization, it inherits ISerializable class.

3.3.2 Member Function Documentation

3.3.2.1 int EventMessage::deserialize (const std::string & *sBuf*) [virtual]

Deserialize the message after receiving from the network.

3.3.2.2 std::string EventMessage::GetFileName () [inline]

Gets the file name.

3.3.2.3 std::string EventMessage::GetFilePath () [inline]

Gets the file path.

3.3.2.4 int EventMessage::GetFileSize () [inline]

Gets the file size.

3.3.2.5 uint8_t EventMessage::GetFileType () [inline]

Gets the file type.

3.3.2.6 int EventMessage::GetReasonCode () [inline]

Gets the reason code of the message.

3.3.2.7 std::string EventMessage::GetReasonString () [inline]

Gets the reason string of the message.

3.3.2.8 uint8_t EventMessage::GetRelatedModule () [inline]

Gets the source module name.

3.3.2.9 int EventMessage::GetSearchID () [inline]

Gets the search id. Only useful for KAD file transfer.

3.3.2.10 int EventMessage::GetStatus () [inline]

Gets the status of the message.

3.3.2.11 void EventMessage::serialize (std::string & *sBuf*) const [virtual]

Serialize the message before sending in the network.

3.3.2.12 void EventMessage::SetFileName (const std::string & *s*) [inline]

Sets the File name.

3.3.2.13 void EventMessage::SetFilePath (const std::string & *s*) [inline]

Sets the File path.

3.3.2.14 void EventMessage::SetFileSize (int *t*) [inline]

Sets the File size.

3.3.2.15 void EventMessage::SetFileType (uint8_t *t*) [inline]

Sets the File type.

3.3.2.16 void EventMessage::SetReasonCode (int *i*) [inline]

Sets the reason of the message.

3.3.2.17 void EventMessage::SetReasonString (const std::string & *s*) [inline]

Sets the reason string of the message for human readibility.

3.3.2.18 void EventMessage::SetRelatedModule (uint8_t *t*) [inline]

Sets the source module name.

3.3.2.19 void EventMessage::SetSearchID (int *i*) [inline]

Sets the search id. Only required for KAD file transfer.

3.3.2.20 void EventMessage::SetStatus (int *i*) [inline]

Sets the status of the message.

The documentation for this class was generated from the following files:

- EventMessage.h
- EventMessage.cpp

4 Policy Component File Documentation

4.1 util.h File Reference

```
#include <vector>
```

Typedefs

- typedef std::pair< uint32_t, uint16_t > **AddressPortPair**

Functions

- int GetFileSize (const std::string &sFile)
- std::string GetDotFormatIP (uint32_t ip)
- void reuse (int fd)
- void nonblock (int fd)
- void nodelay (int fd)
- void **ByteOrderTesting** ()
- int RunAsDaemon (const char *stdoutname)
- int **write_pid** (const char *sFileName)
- AddressPortPair GetSockName (uint32_t sock)
- int StringTokenizer (const std::string sToParse, const char *cToken, std::vector< std::string > &v)

4.1.1 Detailed Description

Utility functions.

4.1.2 Function Documentation

4.1.2.1 std::string GetDotFormatIP (uint32_t *ip*)

Gets the dot formatted IP address as a string from int IP.

4.1.2.2 int GetFileSize (const std::string & *sFile*)

Gets the file size in bytes of a given file.

4.1.2.3 AddressPortPair GetSockName (uint32_t *sock*)

Gets the ip and port from a socket. Useful for debugging.

4.1.2.4 void nodelay (int *fd*)

Disables nagle's algorithm.

4.1.2.5 void nonblock (int *fd*)

Set the socket for as non blocking.

4.1.2.6 void reuse (int *fd*)

Set the socket for reusing.

4.1.2.7 int RunAsDaemon (const char * *stdoutname*)

Makes the current process as a daemon.

4.1.2.8 int StringTokenizer (const std::string *sToParse*, const char * *cToken*, std::vector< std::string > & *v*)

Special string tokenizer; not generic.

Index