User Manual

If you're running a Linux/x86 environment, the provided binaries should work. Otherwise, you will have to build the sources to test my system. First and foremost, you must have a C++ compiler and I recommend a UNIX environment for building and testing. Once you have a compiler, you will need to download, build, and install libxml2 and xmlwrapp. I believe libxml2 is already on Bengal, but xmlwrapp is not. You can obtain libxml2 and xmlwrapp from the following links:

libxml2: http://www.xmlsoft.org/
xmlwrapp: http://pmade.org/software/xmlwrapp/index.html

From what I understand these install on most Unix systems without any hassle. My environment is Linux (kernel 2.4.22) with gcc 3.3.1, and installation took about 5 minutes. On Bengal it took even less since it already had libxml, and xmlwrapp took about a minute or less to build.

After you install these dependencies, you can run 'make xmlhandle'. This will produce a binary called xmlhandle that uses the following command line arguments:

```
xmlhandle <xmlfile> <username> <groupname> <domain_name> <ip_address> [-s style_sheet] [-a]
```

xmlfile: name of the xml file
username: username of requester
groupname: groupname of requester
domain_name: network domain connection is coming from
ip_address: ip address of the request
[-s style_sheet] apply optional stylesheet
[-a] don't remove access control attributes from document transformation

By default, the transformation will not include the access control attributes, so the '-a' switch is useful for testing since it overrides that. I don't recommend using the '-s' option since transformations will usually not validate since so much content has been filtered out. This is included for experimental purposes, but please don't even test it if you plan to let it affect my grade! Along with the source I have included the sample documents that I used for testing. Here is a description of these documents:

plato.xml: the original xml document (used for comparison)
plato.acml: the well-formed xml document with access control attributes (use this for testing the program)
plato.xsl: the stylesheet (again, the stylesheet feature is experimental and not meant for transformed xml!!!)

This is an example of how you might execute the program:

```
./xmlhandle plato.acml mike Everyone missouri.edu 128.206.180.54 –a
```

Of course, you will want to modify the plato.acml file and the arguments to test it more thoroughly.

I generated an API reference with Doxygen, check it out! Or if you prefer, here is an extremely simple API reference:

class AccessControl: compares the elements access control attributes with the requester's information to grant access or deny it
public methods:

AccessControl();
standard constructor
~AccessControl();
standard destructor
void reset();
clear the access control sets
void setSubject(string user, string group, string ip, string domain);
set the requester's information
void allowUser(string user);
add a user to the set of allowed users
void denyUser(string user);
add a user to the set of denied users
void allowGroup(string group);
add a group to the set of allowed groups
void denyGroup(string group);
add a group to the set of denied groups

void allowIP(string ip);
add an ip address to the set of allowed ip addresses
void denyIP(string ip);
add an ip address to the set of denied ip addresses
void allowDomain(string domain);
add a network domain to the set of allowed network domains
void denyDomain(string domain);
add a network domain to the set of denied network domains
bool grantAccess();
find out if the requester has access to the current element (node)

class AccessRestrictionParser: inherits from xml::tree_parser; performs the document transformation by filtering out data recursively based on the access controls
public methods:
AccessRestrictionParser(const char *filename, bool allow_exceptions);
constructor; filename is the name of the xml file
~AccessRestrictionParser();
standard destructor
void setShowAttributes(bool show) { m_show_attributes = show; }
this gets called if the '-a' switch is set
void setStyleSheet(StyleSheet *ss) { m_style_sheet = ss; }
set the stylesheet (stylesheet support is experimental)
void parseAccessTree(string user, string group, string domain, string ip);
parse the xml tree and filter out data based on user, group, domain, and ip
void printTree();
print the xml document
const xml::document& getAccessDocument() const;
get a reference to the document root
friend std::ostream& operator<< (std::ostream &stream, const AccessRestrictionParser &arp);
add support for the << operator

class StyleSheet: apply a stylesheet (this is experimental!!!)
public methods:

StyleSheet(std::string name);

constructor

~StyleSheet();

standard destructor

xml::document& transform(xml::document *doc);

apply the stylesheet to the xml document

For any questions, please do not hesitate to contact me.