Contributing to XML-Security

1. Introduction

The XML Security Project is an Open Source volunteer project released under a very open license. This means there are many ways to contribute to the project - either with direct participation (coding, documenting, answering questions, proposing ideas, reporting bugs, suggesting bug-fixes, etc..) or by resource donations (money, time, publicity, hardware, software, conference presentations, speeches, etc...).

To begin with, we suggest you to subscribe to the XML-Security mailing list (follow the link for information on how to subscribe and to access the mail list archives). Listen-in for a while, to hear how others make contributions.

You can get your local working copy of the latest and greatest code (which you find in the xml-security module in the CVS code repository. Review the todo list, choose a task (or perhaps you have noticed something that needs patching). Make the changes, do the testing, generate a patch, and post to the dev mailing list. (Do not worry - the process is easy and explained below.)

2. Help Wanted Here

The rest of this document is mainly about contributing new or improved code and/or documentation, but we would also be glad to have extra help in any of the following areas:

- Answering questions on the mailing list - there is often a problem of having too many questioners and not enough experts to respond to all the questions.
- Testing the package (especially its less-frequently-used features) on various configurations and reporting back.
- Debugging - producing reproduceable test cases and/or finding causes of bugs. Some known bugs are informally listed on To Do, and some are recorded in Bugzilla (see explanation below).
- Specifying/analysing/designing new features - and beyond. (If you wish to get involved with this, please join the mailing list, install and try out xml-security and read some of the mail archives. You should have a strong "fluency" in XML technologies especially XMLDSig and XML Encryption, Java or C++ and a basic understanding of the architecture of this package.

3. CVS Usage Precis

An overview of how to use CVS to participate in the XML Security development. Do not be afraid - you cannot accidently destroy the actual code repository, because you are working with a local copy as an anonymous user. Therefore, you do not have the system permissions to change anything. You can only update your local repository and compare your revisions with the real repository.

(Further general CVS usage information is at www.cvshome.org and your local `info cvs` pages or `man cvs` pages or user documentation.)

Have a look at the Java or C++ installation pages to see how to get a local copy of the source.

4. CVS Committer with Secure Shell access

After a developer has consistently provided contributions (code, documentation and discussion), then the rest of the dev community may vote to grant this developer commit access to CVS.

You will need secure access to the repository to be able to commit patches. Here are some resources that help to get your machine configured to use the repository over SSH.

- The CVS Book
- www.cvshome.org

5. Procedure for Raising Development Issues

There are two methods for discussing development and submitting patches. So that everyone can be productive, it is important to know which method is appropriate for a certain situation and how to go about it without confusion. This section explains when to use the `developer` mailing list the bug database.

Research your topic thoroughly before beginning to discuss a new development issue. Search and browse through the email archives - your issue may have been discussed before. Prepare your post clearly and concisely.

Most issues will be discovered, resolved, and then patched quickly via the `developer` mailing list. Larger issues, and ones that are not yet fully understood or are hard to solve, are destined for Bugzilla.

Experienced developers use Bugzilla directly, as they are very sure when they have found a bug and when not. However, less experienced users should first discuss it on the user or developer mailing list (as appropriate). Impatient people always enter everything into

Bugzilla without caring if it is a bug of our package or their own installation/configuration mistake - please do not do this.

As a rule-of-thumb, discuss an issue on the `developers` mailing list first to work out any details. After it is confirmed to be worthwhile, and you are clear about it, then submit the bug description or patch via Bug Tracking.

Perhaps you do not get any answer on your first reply, so just post it again until you get one. (But please not every hour - allow a few days for the list to deal with it.) Do not be impatient - remember that the whole world is busy, not just you. Bear in mind that other countries will have holidays at different times to your country and that they are in different time zones. You might also consider rewriting your initial posting - perhaps it was not clear enough and the readers eyes glazed over.

6. Contribution Notes and Tips

This is a collection of tips for contributing to the project in a manner that is productive for all parties.

- Every contribution is worthwhile. Even if the ensuing discussion proves it to be off-beam, then it may jog ideas for other people.
- Use sensible and concise email subject headings. Search engines, and humans trying to browse a voluminous list, will respond favourably to a descriptive title.
- Start new threads with new Subject for new topics, rather than reusing the previous Subject line.
- Keep each topic focused. If some new topic arises then start a new discussion. This leaves the original topic to continue uncluttered.
- Whenever you decide to start a new topic, then start with a fresh new email message window. Do not use the "Reply to" button, because threaded mail-readers get confused (they utilise the `In-reply-to` header). If so, then your new topic will get lost in the previous thread and go unanswered.
- Prepend your email subject line with a marker when that is appropriate, e.g. `[Patch]`, `[Proposal]`, `[RT]` (Random Thought which quickly blossom into research topics :-), `[STATUS]` (development status of a certain facility).
- When making changes to XML documentation, or any XML document for that matter, use a validating parser (one that is tried and true is OpenSP/onsgmls). This procedure will detect errors without having to go through the whole `build docs` process to find them. Do not expect Forrest or the build system to detect the validation errors for you - they can do it, but that is not their purpose. (Anyway, nsgmls validation error messages are more informative.)
- Remember that most people are participating in development on a volunteer basis and in their "spare time". These enthusiasts will attempt to respond to issues. It may take a little

while to get your answers.

- Research your topic thoroughly before beginning to discuss a new development issue. Search and browse through the email archives - your issue may have been discussed before. Do not just perceive a problem and then rush out with a question - instead, delve.
- Try to at least offer a partial solution and not just a problem statement.
- Take the time to clearly explain your issue and write a concise email message. Less confusion facilitates fast and complete resolution.
- Do not bother to send an email reply that simply says "thanks". When the issue is resolved, that is the finish - end of thread. Reduce clutter.
- You would usually do any development work against the HEAD branch of CVS.
- When sending a patch, you usually do not need to worry about which CVS branch it should be applied to. The maintainers of the repository will decide.
- If an issue starts to get bogged down in list discussion, then it may be appropriate to go into private off-list discussion with a few interested other people. Spare the list from the gory details. Report a summary back to the list to finalise the thread.
- Become familiar with the mailing lists. As you browse and search, you will see the way other people do things. Follow the leading examples.