VomsSnooper - a tool for managing VOMS records

S Jones
University of Liverpool, Department of Physics,
Oliver Lodge Laboratory, Oxford Street, Liverpool, L69 7ZE, UK
E-mail: sjones@hep.ph.liv.ac.uk

Abstract. VomsSnooper is a tool to keep documents and sites up to date with the newest VOMS records, removing the need for manual edits to security configuration files. The input data is administered centrally at the Operations Portal, and is made available in XML format. Prior to VomsSnooper, it was necessary at each site to manually convert the XML data. VomsSnooper automates this process by checking and creating new VOMS records directly from the portal, providing a bridge between the Operations Portal and the site configuration. The tool is also used to publish online the VOMS data of thirty Approved VOs in the GridPP wiki.

1. Introduction
In this paper, I explain how a troublesome gap in the security functionality of the grid middleware was exposed. The gap caused duplication of effort and loss of service. I show how cheap, common open source software was employed locally to solve the problem at Liverpool. The outcome of this initiative was a useful tool that can be used through the UK or globally by any grid site to maintain its security records relating to VOMS servers. I also comment on some of the technical choices and I report on the uptake of the tool at sites and possible future developments.

2. Previous ways of disseminating VOMS records
The motivation to automate the update of VOMS[1] records arose from a previous process that was somewhat ad hoc. Prior to VomsSnooper[2], a centralised database, the Operations Portal[3], had been set up containing the configuration details of most VOs[4]. But the process for disseminating the records from there had not been well developed and a set of ad hoc procedures had been adopted. Any grid site had several options for maintaining its configuration. It could use the Operations Portal interface to view the records of any particular VO and scrape the settings from that, or the site could access and download the entire set of public VOMS records in XML[9] format and pick from those. In either case, the records would have to be manually transcribed or edited to make them suitable for Yaim[5], which is the canonical configuration tool for grid sites. As an alternative to these methods, a document containing the records for all “Approved VOs”[6] had been written on the GridPP wiki to hold pre-formatted records, but this had been done on a one-off basis. It had fallen into disorder because no systematic approach had been adopted to maintain it. To further confuse matters, some VOs had not registered their records with the Operations Portal. This led to unofficial registries potentially containing non-canonical and untested records.

1 To whom any correspondence should be addressed.
Such problems led to difficulties at the sites. First, each site had to develop its own way to pull the records in, duplicating effort. Secondly, as there were various unsynchronised sources of data, there could be no standard across the sites. Hence there was extra work and loss of service.

3. Proposal for change

The main functional requirement for any replacement was to automate the acquisition of VOMS records from the Operations Portal and the translation into suitable formats such as Yaim[5] and LSC[8]. This would systematically address the defects in the ad hoc process. The new procedure should be compatible with existing measures if possible, to reduce the learning impact. I proposed to eliminate unofficial registries altogether and I proposed to retain the Approved VOs document but bring it up to operational standards and give it a regular maintenance schedule. I also proposed to give functions to allow sites to bypass the Approved VOs document and directly update their configurations if they chose. A small survey of some members of the user community uncovered typical use cases for VOMS records life-cycle management, and the following list comprises a working subset of those.

- newVomsRecsForMySite - allow the site admin to download a new set of records for inclusion in the local configuration.
- checkSite - get the newest records from the Operations Portal and make sure the site configuration tallies.
- fixApprovedVos - maintain the Approved VOs[6] document.
- makeTarball - create a tar[7] ball of all the records for all approved VOs.
- getLSCRecords - generate new LSC[8] records automatically from XML.
- checkMyLSCRecords - compare the LSC records at a site to the ones that exist in the Operations Portal.

4. Implementation

A snapshot of the entire set of public VOMS records can be obtained in XML format as a download from the Operations Portal website. This format is suitable for parsing by a SAX[10] XML parser. Many languages have good SAX parser frameworks but I chose Java for ease of development - it has the particularly powerful Eclipse[11] development editor. Furthermore, Java offers good hardware/software portability[12], and support for component based design[13]. The basic functionality for parsing and storing sets of VOMS records is contained in a java library[14] file, with which individual use cases are realised by tailoring specific interfaces. Ultimately, the package contains the following discrete tools:

- VomsSnooper parses the Operations Portal XML and writes out formatted data records in a standard, sorted way suitable for Yaim.
- SidFormatter parses existing Yaim files and prints them out in a standard, sorted manner.
- SiteChecker parses existing Yaim files and the Operations Portal XML using a SAX parser. It compares both data sources and prints out a report on the differences.
- CicToLsc also uses the SAX parser, but only includes functionality for making LSC files. It is for the convenience of people who wish to bypass the Yaim configuration process.

The development and maintenance is done in an evolutionary, spiral[15] process model, and the work bears an Open Source license. Git[16] and Github are used for version control, and the work is packaged and distributed as an RPM[18] in a central Yum[17] repository. Yum and/or RPM are the main tools for installation. Some manual configuration is required to set up the use cases for the requirements of any particular site.

5. Results
The effects on documentation were immediate. The Approved VOs document could be maintained with approaching 100% accuracy. Comparisons showed that the only discrepancies compared to the Operations Portal were due to latency with respect to the rate at which the Operations Portal changes. To minimise this, the Approved VOs document is updated on a weekly basis, and also whenever notice is given that a change has occurred in the Operations Portal.

Broadly positive qualitative feedback was received from sites that adopted VomsSnooper. One user reported in an email “VomsSnooper worked really well, all done in an hour or so”, and “one particularly neat thing is that we have a local VO and it spotted it fine”. Some criticism was levelled, but it was mostly stylistic.

Take up at individual sites is hard to measure. Informal feedback might suggest that a handful of sites use one or more of the use cases provided. Those that have used the tool are mostly positive about its usefulness. Even without a large take up of the tool, it is still influential as the tool maintains the canonical document, the Approved VOs. Many UK sites are at least indirectly effected by this development.

6. Discussion
While reviewing this work, it occurred to me that another Object Oriented language may have been suitable. Python[19] has many of the advantages of Java (good SAX parser, easy development, hardware/software portability, support for component based design) but it also has a compact syntax and it requires no compilation phase. Python is not subject to the same degree of change as Java, and Python programs conveniently execute directly as shell scripts[20], while Java programs require shell script wrappers that are a maintenance burden[21].

7. Conclusions and future plans
The Approved VOs document is now kept semi-automatically up to date in accordance with any changes at the operations portal. Alerts are sent to inform sites wherever significant changes are made. Thus, sites that rely on the Approved VOs document are now assured to get reminders of any changes that may be necessary.

Furthermore, sites have a tool at their disposal to generate their own security records independently of the Approved VOs document. Ultimately we have the option to choose whichever of these two maintenance processes is superior, but for the time being both will be maintained.

In the longer term, if many small VOs were to emerge and changes became much more frequent, even this partial automation may not be sufficient to keep pace. In this case, it would be desirable to research ways to convert this process into a service, using some distributed quasi-real-time architecture.
8. References

[1] Alfieri R, Cecchini R, Ciaschini V, dell’Agnello L, Frohner Á, Gianoli A, Lőrentey K and Spataro F 2004 Grid Computing (Berlin: Springer) pp 33-40

[2] Jones S October 2013 (accessed) Tools for managing VOMs records http://hep.ph.liv.ac.uk/~sjones/VomsSnooper_Tools.html

[3] Cordier H, L’Orphelin C, Reynaud S, Lequeux O, Loikkanen S, & Veyre P 2011 From EGEE Operations Portal towards EGI Operations Portal In Data Driven e-Science (Springer: New York) pp 129-140 http://link.springer.com/content/pdf/10.1007%2F978-1-4419-8014-4.pdf

[4] Laure E, Edlund A, Pacini F, Buncic P, Barroso M, Di Meglio A, ... & Fisher M 2006 Programming the Grid with gLite http://cds.cern.ch/record/936685/files/egee-tr-2006-001.pdf

[5] Yaim https://twiki.cern.ch/twiki/bin/view/EGEE/YAIM

[6] Gronbech P Oct 2013 (accessed) GridPP approved VOs https://www.gridpp.ac.uk/wiki/GridPP_approved_VOs

[7] Wikipedia Oct 2013 (accessed) Tar (computing) http://en.wikipedia.org/wiki/Tar_%28computing%29

[8] LSC Files https://twiki.cern.ch/twiki/bin/view/LCG/VOMSLSCfileConfiguration

[9] Bray T, Paoli J, Sperberg-McQueen C M, Maler E, & Yergeau F 1997 Extensible markup language (XML) World Wide Web Journal, 2(4), pp 27-66 www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf

[10] Wikipedia Oct 2013 (accessed) Simple API for XML http://en.wikipedia.org/wiki/Simple_API_for_XML

[11] Binstock A Oct 2013 (accessed) Top Java programming tools http://www.infoworld.com/d/developer-world/infoworld-review-top-java-programming-tools-191

[12] Roulo M Oct 2013 (accessed) Java’s three types of portability http://www.javaworld.com/javaworld/jw-05-1997/jw-05-portability.html

[13] Kopetz, H 1998 Component-based design of large distributed real-time systems Control Engineering Practice, Volume 6, Issue 1, pp 53-60

[14] Pugh, W 1999 Compressing Java class files In ACM SIGPLAN Notices (Vol. 34, No. 5, pp 247-258) ACM http://pdf.aminer.org/000/542/799/compressing_java_class_files.pdf

[15] Boehm, B 1988 A spiral model of software development and enhancement Computer, 21(5), pp 61-72

[16] Dabbish, L, Stuart, C, Tsay, J, & Herbsleb, J 2012 Social coding in github: transparency and collaboration in an open software repository In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work (pp. 1277-1286). ACM.

[17] Yum http://yum.baseurl.org/

[18] RPM http://www.rpm.org

[19] Cunningham D, Subrahmanian E, & Westerberg A 1997 User-Centered Evolutionary Software Development Using Python and Java In Proceedings of the 6th International Python Conference http://www.python.org/workshops/1997-10/proceedings/cunningham.html

[20] Delaney R Oct 2013 (accessed) Python Scripts as a Replacement for Bash Utility Scripts http://www.linuxjournal.com/content/python-scripts-replacement-bash-utility-scripts

[21] Deng Z Oct 2013 (accessed) A better java shell script wrapper http://saltnlight5.blogspot.co.uk/2012/08/a-better-java-shell-script-wrapper.html