University College London
Department of Computer Science
M.Sc Data Communications, Network and Distributed Systems

Extensible Secure Event and Report Toolkit

Executive Summary

Group Members
Shaohua Yan
Yang Wang
Xiaoyue She
Ping Liang

Supervisor
Dr. Saleem Bhatti

30 Aug 2005

Extensible Secure Event and Report Toolkit (ExSERT), a LINX networking project started last year, aimed to provide a secure distributed network monitoring and network data analysis system. The main purpose of this project is to resolve the problems that IXP meet now - Internet Exchange Point (IXP), a physical infrastructure, allows different Internet Service Providers to exchange Internet traffic between IXPs. Since many IXPs have similar network monitoring requirements and all have semantically similar tools that are developed in house, the tools often differ in the presentation of the information. The IXPs also use different scripts for processing these logs. As a result, they retrieve, store, and analyse the traffic data in a variety of different ways. These differences make it difficult for the IXPs to share information directly, to use common information for troubleshooting, to make comparisons of multi-site data, or to perform analysis using this multi-site data. Most of IXPs in Europe recognise that this is a growing problem.

Last year's project group has already tackled the problems outlined above. Data representation and format has been standardised by defining a set of protocols for the information exchanged between the IXPs. This year's project is still based on the architecture designed last year – system is the combination of three modular components: The Glue component retrieves the non-standard data created by the network and parse into standard format; The PoD analyses the data and transmit the required data to clients; Client component communicates with the PoD which has the associated output. In this year's project, re-engineering and refinement work is carrying out and mainly focuses on the auto-generation of the system components and formalise information model in XML. The project use completely different programming language from last year, so a complete project lifecycle from design and implementation to testing and final report was carried out comprehensively.

In the project report, functional requirement and non-functional requirement were discussed separately. Then Design and implementation were followed to describe how to meet the requirements. The first functional requirement introduced is representing similar data in a common format. XML becomes the preferred syntax since XML is a standardised protocol to exchange data and this resolves the data heterogeneity generated by different back-end network tools effectively. The second functional requirement is to allow generation of code from schema. Compiler was developed to generate the main components such as Glue and PoD, and produced common API for data analysis. The auto-generation of code greatly saves the time and resources while new PoD with different requirements need to be created. The last functional requirement is secure communication between clients and servers. SSL was implemented to provide data encryption. It also provides the secure access by managing the certificate and key files.

There are many non-functional requirements, such as easy to use, easy to deploy and use of open-available, free components, allow integration of existing data sources such as the standard documents like SNMP and proprietary files, easy to extend and add new report and event types. One of our implementation is to use Python, a scripting language, to develop toolkits. The major reason to choose Python is that Python is very easy to use and implement, saving the time of programming. So many IXP prefer this scripting language for data processing. Python also provide excellent libraries to work with XML. Some other implementations are also the novel parts of this year's project, including common data representation XML, XML validation, binary encoding, compiler, database for certificate management and security between client and server. Each part was introduced clearly in the report. Among these parts, compiler can meet most of the requirements such as easy to extend and to integrate new functions.

Furthermore, Several API functions were added to make different kinds of PoD, which can satisfy different clients with disparate requirements. Each function can build one specific toolkit, such as RTTMeasure, LinkStatus, ThresholdRTT, Utilization, and StatisticRTT.

As an integrated project, the project lifecycle from design and implementation to test plan and final report was completely under the control of project management. In the final report, we also described how our project management works. Extreme Programming was used as a base to manage our project.

Lastly, the report includes a section on future work that describes improvements and plans to the system in order to deploy the monitoring more widely and powerful. Because of the time and resource constraint, some of valuable functionalities and deployments have to be investigated in the future, such as error prediction, looking into routing information, design of super PoD etc. Finally, the report contains a user's manual that describes the system requirements, how to install the toolkit and how to run it.

Despite the many challenges we meet along the way, the project has been completed with a great success. All team members showed a great passion to this project and contributed various skills while working together. The accomplishment of this project not only means the soon deployment in the customer LINX and many other potential users such as other member IXPs of Euro-IX, but also laid a great foundation for the future development of toolkit for IXPs.