Cybersecurity Strategies: The QuERIES Methodology, pp. 20-26
Lawrence Carin, George Cybenko, and Jeff Hughes

The Quantitative Evaluation of Risk for Investment Efficient Strategies (QuERIES) methodology offers a novel computational approach to quantitative cybersecurity risk assessment. The authors based this approach on rigorous quantitative techniques drawn from computer science, game theory, control theory, and economics.

Preliminary experiments have corroborated the QuERIES methodology, suggesting that it provides a broadly applicable alternative to red teaming (which involves attackers who have little or no knowledge of a systems’ internal protection), blackhat analysis, and other decision-support methodologies previously tried in cybersecurity-related risk assessment.

CROWN-C: A High-Assurance Service-Oriented Grid Middleware System pp. 30-38
Paul Townend, Jinpeng Huai, Jie Xu, Nik Looker, Dacheng Zhang, Jianxin Li, and Liang Zhong

The grid computing community is creating service-oriented grids to solve the challenge of coordinated resource sharing and problem solving in virtual organizations. SOGs create new challenges regarding dependability and security, however.

The University of Leeds and Beijing University are collaborating to develop a grid middleware system that features integrated tools for the assessment and deployment of high-assurance systems. The result of this collaboration is a high-assurance SOG middleware system known as CROWN-C.

On Testing and Evaluating Service-Oriented Software pp. 40-46
Wei-Tek Tsai, Xinyu Zhou, Yinong Chen, and Xiaoying Bai

The many standards and related techniques that address Web service security issues have produced a level of security customers can trust. Despite this progress, service reliability, testing, and verification techniques aren’t mature enough to support dependable and trustworthy computing.

To address SOA dependability and trustworthiness issues, the authors propose an open framework that implements group testing to enhance test efficiency. The framework identifies and eliminates test cases with overlapping coverage. It also ranks newly added test cases and reranks existing test cases using updated coverage relationships and recent test results. A case study demonstrates the framework’s effectiveness in reducing testing effort while maintaining effectiveness.

QoS-Reconfigurable Web Services and Compositions for High-Assurance Systems pp. 48-55
I-Ling Yen, Hui Ma, Farokh B. Bastani, and Hong Mei

Much research has addressed service selection based on quality of service (QoS) criteria, such as reliability, real-time constraints, accuracy, and so forth. However, many real-world applications have a limited set of alternative services to choose from.

The authors have developed a rule-based parameterization technique to convert components into reconfigurable services. Further, they have developed QoS analysis and decision-making techniques to select reconfigurable as well as regular services to compose the system.

Assuring Timeliness in an e-Science Service-Oriented Architecture pp. 56-62
John C. Sloan, Taghi M. Khoshgoftaar, and Venkat Raghav

Public-resource computing requires services from numerous providers, each offering an identical service to relatively few requesters, usually researchers on a particular project. At the heart of this SOA, a portal links providers and requesters, primarily coordinating interaction between itself and the many providers registered on it. A volunteer supercomputing portal must cope with several routine failures caused by PC unavailability. A portal must know when it should reassign a work unit to another provider.

The authors have modeled this decision process for a simplified version of the Berkeley Open Infrastructure for Network Computing (Boinc) and translated their model into a timed automata in Uppaal, a model-checking and simulation environment for verifying real-time systems.

Specifying High-Assurance Services pp. 64-71
Colin Atkinson, Daniel Brenner, Giovanni Falcone, and Monika Juhasz

Service-oriented architectures’ ease of assembly and alteration also makes their verification more difficult. While software engineers can integrate a traditional system’s components under controlled conditions and test them together before deployment, they do not bring the services together until they configure the system in its runtime environment.

The authors’ approach specifies services intuitively, simplifying the use of dynamic, test-oriented techniques for quality assurance.