

A user-friendly approach to computational grid security

B. Beckles

*University of Cambridge Computing Service, New Museums Site, Pembroke Street,
Cambridge CB2 3QH*

P. V. Coveney

*Centre for Computational Science, Department of Chemistry, University College London,
Christopher Ingold Laboratories, 20 Gordon Street, London WC1H 0AJ*

P. Y. A. Ryan

School of Computing Science, University of Newcastle, Newcastle upon Tyne NE1 7RU

A. E. Abdallah

*Institute for Computing Research, Faculty of Business, Computing, and Information
Management, London South Bank University, 103 Borough Road, London SE1 0AA*

S. M. Pickles, J. M. Brooke, and M. McKeown

*Manchester Computing, Kilburn Building, The University of Manchester, Oxford Road,
Manchester M13 9PL*

Abstract

Many of the existing security components and frameworks for computational grid environments either suffer from significant usability issues for end-users and/or administrators, or their administration and deployment is extremely complex and resource-intensive. This has led to a situation where using such environments securely is so difficult that end-users either refuse to use them or else deliberately use them in an insecure fashion. In this paper we describe work underway to provide more user-friendly security mechanisms for computational grid environments.

1 Introduction

Although computational grid environments are being increasingly considered for use for scientific computing, there is still disagreement about the precise definition of “computational grid” and related terms ([1]). For our purposes, we define “computational grid environment” to be a distributed computing environment which is transparent across multiple administrative domains (following [2]). While the apparent benefits of working in such environments have been widely publicised (e.g. [3]), it is still the case that there is relatively little use of “grid technology” for serious computing projects ([4], [5]). In particular, routine use of computational grid environments by the academic community is still far from widespread, with many researchers being reluctant to use the existing environments (as noted in [6], [7]).

One class of problem faced by end-users of these computational grid environments is to do with the usability of the security mechanisms/frameworks usually deployed in these environments ([8]). This is particularly a problem regarding authentication in these environments ([8]). In fact, reports from end-users have revealed that some users will knowingly use the environments in a manner they know to be insecure because using it in the mandated “secure”

fashion is too difficult ([8]). Some potential users have even refused to use these environments at all if they are forced to use the “usual” authentication mechanism (digital certificates) used by the existing environments ([9]). Thus the usability problems of the existing grid security solutions are so serious as to be a major barrier to the adoption of grid technology by the wider community.

In this paper we outline some of the major usability and other related problems with the existing security solutions in use in computational grid environments and then discuss our proposed methods of addressing these problems. The key aspect of our development methodology is the adoption of a “user-friendly” approach to grid security, which combines aspects of usability engineering with formal methods in security engineering. This ensures that not only are our security components easy for our end-users to use, but also that they are theoretically sound.

2 Grid security: the problems

Before discussing the problems we have encountered with the existing grid security solutions we define a few terms that we shall use frequently in our discussion of these problems. We define “usability” to be the extent to which the user (subjectively) feels that the software has

successfully fulfilled their requirements for that software. Depending on the context, the user in question may be the end-user of the software, the system administrator installing, maintaining or administering the software, etc. We consider software to be “user-friendly” if its usability is high and if the user is able to use it appropriately in such a manner that they are either unaware that they are using it or that its impact on their user experience is positive.

In the context of computational grid infrastructure, we use “heavyweight” to mean software that is some combination of the following:

- complex to understand, configure or use;
- has a system “footprint” (disk space used, system resources used, etc.) that is disproportionately large when considered as a function of the frequency and extent to which the user uses the software;
- has extensive dependencies, particularly where those dependencies are unlikely to already be satisfied in the operating environment, or where the dependencies are themselves “heavyweight”;
- difficult or resource-intensive to install or deploy;
- difficult or resource-intensive to administer; and/or
- not very scalable.

We use the term “lightweight” to refer to software that is not “heavyweight”, i.e. it possesses none of the features listed above, or only possesses a very small number of them to a very limited extent. Thus lightweight software would be characterised by being:

- not too complex for end-users, developers and administrators to use and understand in their normal work context,
- easy to deploy in a scalable manner, and
- easy to administer and maintain.

See [6] and [10] for a fuller discussion of lightweight and heavyweight grid middleware.

The problems we have encountered with the current grid security mechanisms and frameworks tend to fall into two main categories. There are those problems that seem to be common to most of the existing grid security solutions, e.g. excessive complexity, poor scalability, high deployment and maintenance overheads, etc. We observe that all these solutions have a common characteristic that we believe is responsible for these problems, namely they are all very heavyweight solutions to the security problems they address.

Then there are those problems that are specific to the particular aspect(s) of security (e.g. authentication, authorisation, etc) with which the

solution is concerned, e.g. credential management. First we shall outline those problems that we have encountered that we believe are due to the heavyweight nature of the existing solutions, and then we shall discuss problems specific to particular aspects of computational grid security.

2.1 Heavyweight security solutions

Many of the problems inherent in heavyweight grid middleware have been discussed in detail elsewhere ([6], [2], [10]), and so we shall not discuss them here. Instead we outline the specific problems that the heavyweight nature of the existing grid security solutions causes in the context of the security of computational grid environments. (It is important to note, however, that, security considerations aside, heavyweight grid middleware is, by its very nature, a significant barrier to the wider uptake of grid technology ([6], [11], [2]).) The specific problems that such grid middleware causes in a security context include the following:

- *Complexity:*

Amongst security professionals it is well known that complexity is the enemy of security, indeed, some security professionals have described complexity as the *worst* enemy of security ([12]). The existing grid security solutions are extremely complex, and this makes them very difficult to configure or use, and, most crucially, to understand ([13], [8]). Thus it is very difficult even for experienced system administrators to correctly install and configure these security solutions ([14]).

- *Extensive dependencies:*

The security of a piece of software is – at a minimum – dependent on the security of its components and dependencies. It is also dependent on many other factors, but if the security of any of its components and dependencies is breached then the likelihood is high that the security of the software as a whole will also be violated (although there are techniques, such as privilege separation ([15]), which can help mitigate this). The software’s components and dependencies thus form a “chain of trust”, which is only as strong as its weakest link ([12]). It therefore follows that, all else being equal, software with extensive dependencies is both more likely to be vulnerable to security exploits, and also more difficult to secure in the first place, than software with few dependencies.

It is also worth mentioning in this context that software which installs its own versions of system libraries (or is statically compiled with respect to those system libraries) is also likely to be more insecure. This is because its versions of these

libraries are unlikely to be as readily upgraded in response to security alerts as the system libraries, in part because system administrators may well not be aware that the software uses its own versions of these libraries.

- *Scalability:*

Many of the current grid security solutions are not particularly scalable ([16], [17]). Given that, at least in principle, computational grids can consist of thousands of individual users and machines and hundreds or thousands of administrative domains, it is clear that grid middleware whose scalability is poor will not be suitable for medium to large grids. One aspect of this lack of scalability that has serious security implications is the requirement for the security configuration of each node and/or client in a grid environment to be individually configured ([8], [14]). As one might expect, this leads to some nodes/clients being incorrectly configured, and in an environment with a large number of nodes this may remain unnoticed for a considerable period of time.

In addition, where this configuration must be maintained on a regular basis – particularly if the software is designed to stop working if its configuration is not kept up-to-date – it is not unusual for users to have found ways of circumventing this. For example, where up-to-date certificate revocation lists (CRLs) are required for successful operation, often no CRLs will be installed to avoid the software failing when the CRL is no longer current ([8]).

- *Usability:*

From the definition of “heavyweight software” given above it should be clear that it is inherently difficult for such software to have high usability or to be user-friendly ([10]). In addition, an examination of the software development process for most of the existing grid security solutions reveals very little evidence of usability considerations playing a significant role in their design. Given the wealth of usability-related problems ([8]) with the Grid Security Infrastructure (GSI) ([18]) – the principal authentication mechanism used in existing grid environments – it seems unlikely that its design was informed by many usability concerns. The authors are only aware of two grid security components/solutions in active use – MyProxy ([19]) and PERMIS ([20]) – where usability concerns have been explicitly considered at some point in their development/deployment ([19], [21]), and even with those products it is not clear whether usability was a core concern of their designers.

Unfortunately, since security mechanisms whose usability is poor in the environment in

which they are deployed are inherently insecure ([22], [23]), it is therefore likely that most of the existing grid security solutions will be deployed or used in an insecure manner in practice. Further, usability is not a “feature” that can be bolted on at the end of the development process; usable systems are only possible if usability is a central concern of the designers at all stages of the system’s design and development ([24], [25], [26]). It is thus unfortunately the case that most of the existing grid security solutions are unlikely to be usable or secure in practice.

2.2 Authentication

As mentioned above, the principal authentication mechanism in current computational grid environments is GSI, and there are a number of serious problems with this mechanism; these are discussed in some detail in [27], [8] and [14]. Perhaps the most significant point that has emerged from these examinations of the problems with GSI is that the digital certificates upon which it relies are “cognitively difficult objects” for its users ([8]). This is borne out by the fact that some potential users of grid computing environments have refused to use those environments if they required a digital certificate in order to do so ([9]). It would thus seem probable that any better authentication solution for grid environments must not rely on users making active use of digital certificates ([8]). Ideally, users should not have to have such certificates at all ([28], [8], [14]).

2.3 Authorisation

The current computational grid environments provide very limited authorisation mechanisms ([16], [17]); most of them rely on simple “allow” lists in manually maintained text files. Although more sophisticated mechanisms are available (e.g., CAS ([29]), VOMS ([30]), PERMIS ([20]), Shibboleth ([31])), few of these are currently in active use in the existing environments. Some of these mechanisms (e.g. CAS, VOMS) are based on GSI and thus inherit its usability and security problems. In addition, they are all, to a greater or lesser degree, heavyweight (in the sense defined above). This leaves system administrators with two choices, neither of which is satisfactory: they can either rely on an overly simple mechanism that does not scale well (“allow” lists in text files), or they can attempt to use one of the more sophisticated, heavyweight, solutions, with all of the associated problems inherent in such solutions.

2.4 Auditing

The auditing features of current computational grid environments are, at best, rudimentary. GSI, the

principal authentication mechanism, relies wholly on the integrity of users' credentials being maintained – the only auditing information provided is the distinguished name (DN) of the digital certificate and the IP address from which the authentication request appears to originate. As discussed in [8], the likelihood of the integrity of the digital certificate being compromised is high. Also, in the general case, an entity's apparent IP addresses cannot be relied upon to belong to it (due to IP address "spoofing"). Thus, the audit information provided is often of little value when attempting to detect or clean up after a security incident.

Very little work has currently been done in the area of the auditing requirements of computational grid environments, and so it is not clear what information needs to be collected, or how it should be stored, processed, etc ([32]). However, it is clear that most of the existing security solutions do not address these problems in any detail. Indeed, many of them (e.g. GSI), do not follow accepted best practice of storing the audit data at a location remote to where that data was gathered. This means that, in the event of the security of a machine in the grid environment being breached, the attacker will be able to modify the audit data that might otherwise reveal how they obtained access to the machine.

3 Solution Requirements

From the discussion above, and other work in the areas of grid middleware design ([6], [10], [9]), grid security ([27], [28], [8], [14]), and usable security ([22], [23], [33], [34], [21]), we abstract some very general requirements that any grid security solution should possess. We then consider some of the specific security requirements in the areas of authentication, authorisation and auditing.

3.1 General requirements

As should be clear from the discussion in Section 2.1, heavyweight grid security solutions are unlikely to improve the current situation. Indeed, the seriousness of the problems inherent in heavyweight grid middleware is arguably one of the most significant factors in the relatively low use of current computational grid environments ([7], [10]). Thus a core requirement for any grid security solution is that it is *lightweight* (in the sense defined above).

As discussed above, security solutions whose usability is poor are inherently insecure, and usable security solutions require usability to be a central concern of the developers at all stages of the solution's design and development. Thus grid

security solutions must have *high usability* in their deployed environments, and usability must be a core concern of their designers. As discussed in [10], this requires *continuous user involvement* in the design and development processes from their earliest stages.

Given the large investment (in terms of resources allocated) in the existing computational grid environments, it is clear that, despite their relatively low usability and the high likelihood of their security being violated, they cannot simply be abandoned. Thus our solution also needs to be *interoperable* with the existing computational grid environments.

As security professional Bruce Schneier notes in [12], "In cryptography, security comes from following the crowd", and this is true of secure software in general. From a security standpoint it is generally undesirable to re-invent security primitives (especially cryptographic primitives), since these will, of necessity, have received less testing and formal security verification than existing primitives that are widely used. We therefore intend to *use appropriate existing cryptographic techniques* rather than developing any new ones. In addition, we intend to *use existing cryptographic toolkits and security components*, where we can do so without negatively impacting on our other design goals (such as usability).

In summary, our solution must be:

- (a) lightweight, i.e.
 - of low complexity,
 - easy to deploy,
 - easy to administer,
 - minimal in its external dependencies, and
 - scalable.
- (b) highly usable by its intended users,
- (c) developed in conjunction with its intended users,
- (d) interoperable with the existing computational grid environments (or at least with the most common ones),
- (e) implemented using existing cryptographic techniques, and
- (f) maximal in its use of appropriate existing cryptographic toolkits and other security components.

See [22], [33], [35], [6], [10] and [21] for more complete discussions on appropriate design and development methodologies and guidelines for meeting requirements (a) to (c).

3.2 Authentication

As discussed in Section 2.2, it is vital that *end-user interaction with digital certificates be minimised*, ideally to the point where end-users have no interaction with digital certificates. A number of

methods have been proposed for doing this ([28], [8], [14]). We have chosen to adopt the method described in [28] as it is designed to be interoperable with the existing grid authentication mechanisms, highly scalable, extensible and represents a significant improvement in its auditing facilities over the existing mechanisms. Of equal importance, its development methodology fully embraces the need for usability to be a core concern of any security solution and for user involvement in the design and development processes.

3.3 Authorisation

It seems intuitively obvious that a truly general authorisation framework that would be applicable to any possible scenario in a computational grid environment is likely to be extremely complex and its implementations would quite probably be very heavyweight. The general nature of authorisation frameworks such as PERMIS and Shibboleth is probably precisely what causes them to be such heavyweight security solutions and so unsuitable for our purposes. As discussed in [10], it is probably impossible to develop usable grid middleware that can be used in all the problem domains that might be of interest. Instead, smaller domain-specific middleware is needed.

Thus we do not attempt to re-invent a general authorisation framework. Instead, we propose to concentrate on some specific problem areas of interest and develop a *lightweight* authorisation framework appropriate for those areas. A design goal for this framework is that it is *extensible*, so that others working in similar problem areas are likely to be able to make use of it. As noted in Section 3.1, *interoperability* is a crucial requirement, and so our initial investigation will be into wrapping the flat file authorisation tables used in most existing computational grid environments, in a manner analogous to that proposed in [28] for the existing authentication mechanisms.

3.4 Auditing

As discussed in Section 2.4, the auditing requirements for computational grid environments are not yet well understood, and the existing facilities are quite basic. Further research is necessary before the requirements in this area can be definitively stated. However, it seems likely that *conforming with accepted best practice for network auditing* (e.g. storing audit data at a location remote from where it is collected, not relying on easily falsifiable identifiers such as IP addresses, etc.) would be a sensible place to start. The mechanism proposed in [28] attempts to

improve the reliability of the data collected for audit purposes and we propose to use this as a starting point for further development.

4 Development Methodology

Our development methodology is a combination of user-centred security methodologies (such as [22], [35]), drawing heavily on the user-centred design methodologies (e.g. [25], [36], [37]), in conjunction with formal methods drawn from security analysis (e.g. [38], [39]). There are two principal strands to our development strategy: the development and security verification of working software solutions, and research into, and formal analysis and modelling of, the security space (computational grid environments) in which that software is to be deployed.

Both usability and security requirements are situated requirements, and, as such, are crucially dependent on context. As mentioned above (and discussed in more detail in [6] and [10]), the attempts to produce truly general solutions has led to heavyweight grid middleware whose usability is low and this has directly contributed to the low adoption of grid technology by the academic community. To avoid repeating these mistakes, we are situating our project in a specific context, namely the RealityGrid Project ([40]) and its chosen middleware, WEDS ([2]) and the AHE ([7]). A design goal of our software is thus tight integration with the middleware used by the RealityGrid Project.

As discussed in [41] and [10], a prerequisite for effective development of any software solution is a detailed knowledge of the requirements of the stakeholders for whom that solution is being developed. Thus the first stage in our software development plan is a requirements capture exercise using techniques from the user-centred design methodologies. The results of this exercise are fed into both the formal modelling and analysis of the security problem space under consideration, and the software design process. Low fidelity prototyping (e.g. [42]) is used to ensure that the gathered requirements accurately reflect the users' needs. Our development cycle thus looks something like this:

- 1) Requirements capture and analysis;
- 2) Formal modelling and analysis;
- 3) Software design;
- 4) Formal verification of security of design;
- 5) Usability testing using low fidelity prototypes.

The above cycle is iterative (note that the different phases may partially overlap) with each phase dependent on the successful completion of the previous phase. If a phase cannot be completed successfully, earlier phases are repeated

with the current design and suggested changes from the failed phase as input. This cycle iterates until a stable design is produced.

Note that no program code (even as a prototype) is written until a stable design has been produced, and this design has passed usability trials and security verification. This helps to ensure that software development is user-driven rather than programmer-driven, and that the software design is responsive to user concerns. After the stable design is produced, a software prototype of this design is then developed, which is integrated with our target middleware in the following cycle:

- 1) Development of software prototype;
- 2) Integration with target middleware;
- 3) Usability testing;
- 4) Formal verification of security of integrated prototype.

The above cycle is iterative in a similar manner to the previous cycle. This cycle iterates until a working software prototype has been integrated with the target middleware and passed its usability trials and security verification.

We have adopted a component based approach to the authentication and authorisation mechanisms we are trying to develop, so these are developed as separate components, the development of each following the development cycle described above.

5 Security Improvements

In this section we discuss how our proposed solutions and our software development methodology will improve the security of computational grid environments. First we discuss the impact of our development methodology on the security of our solutions, and then we discuss how our solutions address some of the security issues present in the current computational grid environments.

5.1 Development Methodology

Our development methodology incorporates software development principles and practice (such as [22], [33], [35], [34]) proven to increase the security of systems by addressing usability concerns. In addition, we undertake a continuous programme of usability testing to ensure that our usability goals have actually been achieved. Thus not only will our software have been designed in accordance with proven usability design principles, but we will have tested the software in usability trials to ensure that it genuinely possesses a high degree of usability for its intended user community. Together, these measures guarantee that – at least in usability terms – our solutions

should be major improvements over the current grid security solutions.

In addition, our development methodology incorporates formal security analysis and security verification of both our software design and its implementation. By using formal security methods we can ensure that our software's security model is at least theoretically sound. Whilst this does not, of course, guarantee that the software will be used securely in practice, if any software deployment is to be secure, a prerequisite is that its security model is theoretically sound. We are unaware of any formal security analysis or verification of the existing grid security solutions. Once the final implementation of our software has undergone formal security analysis and verification, we believe it would be reasonable to have greater confidence in our software being secure than in the existing solutions.

5.2 Authentication

From the discussion above (and see [8] and [14] for further details) it is apparent that, in current computational grid environments, there is a significant risk of users' credentials being obtained by unauthorised individuals, and this is principally due to usability issues with the current security mechanisms. Obtaining user credentials would allow attackers to successfully exploit these environments *without* needing to discover and exploit security vulnerabilities in the program code of the security mechanisms. We intend to address the usability issues with the current mechanisms (see Section 3.2), and, as discussed in Section 5.1, will undertake extensive usability testing to *ensure* we have addressed these issues. This means that our software should significantly reduce – or possibly even eliminate – the risk of attackers obtaining users' credentials through the usability deficiencies of the environment's security mechanisms.

5.3 Authorisation

Environments that currently use the existing heavyweight authentication solutions are likely to have inappropriate authorisation policies in place, i.e. the complexity of the solution is such that the actual authorisation policy implemented may not match the intended authorisation policy in the administrator's mind. If the implemented policy is too lax, then unauthorised individuals may be able to access the environment, or users may be able to perform unauthorised actions. If the implemented policy is too strict, this is likely to be quickly discovered by users, who will complain about being unable to perform legitimate actions. This will lead to a relaxation of the policy, and may

well lead to the policy being relaxed to an inappropriately high degree. This may happen because of the dissonance between the administrator's understanding of the authorisation mechanisms and the actual workings of these mechanisms (which led to the mistaken policy being implemented in the first place). Alternatively, fear of further angering the user community may lead the administrator to err on the side of "caution", i.e. to implement an extremely permissive policy so as to be certain that the users are able to do what they should be allowed to do.

As described in Section 5.1, our development methodology actively addresses the usability issues that cause such dissonance. Consequently, our authorisation component will represent significant improvements for administrators (and so for users) over the current situation, i.e. it will be much more likely that administrators will be able to correctly set appropriate authorisation policies.

5.4 Auditing

As discussed in Section 2.4, the auditing facilities of the existing computational grid environments are not very sophisticated, and the auditing requirements are not very well understood. This means that it is likely to be difficult to diagnose attacks against these environments. We intend to actively investigate this area as a better understanding would allow us to implement better auditing facilities. However, as discussed in Section 3.4, merely by following acknowledged best practice for network auditing we will have improved the current situation significantly.

6 Preliminary Work

As described in [28] and [8], our proposed replacement authentication mechanism is nearing the end of its design phase – see [8] for details of usability concerns addressed in the design process. We anticipate soon having some software prototypes for integration with our target lightweight grid middleware ([2], [7]). Once this is done we will begin usability testing of these prototypes. As further development of the other security mechanisms proceeds, we will gradually incorporate prototypes of these mechanisms in accordance with the software development methodology outlined in Section 4.

7 Conclusion

The existing grid security solutions suffer from a number of deficiencies that are serious barriers to wider adoption of grid technologies by the

academic community. Many of these deficiencies stem from the heavyweight nature of the solutions in question, and from their low usability. The low usability of these solutions is probably a consequence of the fact that usability considerations were not central to their design.

In this paper we have described how our "user-friendly" approach to grid security seeks to mitigate the existing deficiencies by closely involving the user in the design and development processes. By combining this with formal methods in computer security we aim to produce lightweight security solutions that are easy to use for our target user population and which are theoretically sound.

Acknowledgements

This work is supported by EPSRC through the *User-Friendly Authentication and Authorisation for Grid Environments* project (EP/D051754/1).

References

- [1] Hurley, J. "THE GRID IDENTITY CRISIS: DEFINING IT AND ITS REALITY", GRIDtoday, Vol. 1, No. 10 (19 August, 2002): <http://www.gridtoday.com/02/0819/100248.html>
- [2] Coveney, P., Vicary, J., Chin, J. and Harvey, M. Introducing WEDS: a WSRF-based Environment for Distributed Simulation (2004). UK e-Science Technical Report UKeS-2004-07: http://www.nesc.ac.uk/technical_papers/UKeS-2004-07.pdf
- [3] Evans, B. "Grid Computing: Too Big To Be Ignored". *InformationWeek*, 10 November 2003: <http://www.informationweek.com/story/showArticle.jhtml?articleID=16000717>
- [4] McBride, S. and Gedda, R. "Grid computing uptake slow in search for relevance". *Computerworld Today*, 12 November 2004: <http://www.computerworld.com.au/index.php?id=138181333>
- [5] Ricadela, A. "Slow Going On The Global Grid". *InformationWeek*, 25 February 2005: <http://www.informationweek.com/story/showArticle.jhtml?articleID=60402106>
- [6] Chin, J. and Coveney, P.V. Towards tractable toolkits for the Grid: a plea for lightweight, usable middleware. (2004). UK e-Science Technical Report UKeS-2004-01: http://www.nesc.ac.uk/technical_papers/UKeS-2004-01.pdf
- [7] Coveney, P.V., Harvey, M.J., Pedesseau, L., Mason, D., Sutton, A., McKeown, M. and Pickles, S. Development and deployment of an application hosting environment for grid based computational science (2005). *Proceedings of the UK e-Science All Hands Meeting 2005*, Nottingham, UK, 19-22 September 2005: <http://www.allhands.org.uk/2005/proceedings/papers/366.pdf>
- [8] Beckles, B., Welch, V. and Basney, J. Mechanisms for increasing the usability of grid security (2005). *Int. J. Human-Computer Studies* **63** (1-2) (July 2005), pp. 74-101: <http://dx.doi.org/10.1016/j.ijhcs.2005.04.017>
- [9] Sinnott, R. Development of Usable Grid Services for the Biomedical Community (2006). *Designing for e-Science: Interrogating new scientific practice for usability, in the lab and beyond*, Edinburgh, UK, 26-27 January 2006.
- [10] Beckles, B. Re-factoring grid computing for usability (2005). *Proceedings of the UK e-Science All Hands Meeting*

- 2005, Nottingham, UK, 19-22 September 2005: <http://www.allhands.org.uk/2005/proceedings/papers/565.pdf>
- [11] Pickles, S.M., Blake, R.J., Boghosian, B.M., Brooke, J.M., Chin, J., Clarke, P.E.L., Coveney, P.V., González-Segredo, N., Haines, R., Harting, J., Harvey, M., Jones, M.A.S., McKeown, M., Pinning, R.L., Porter, A.R., Roy, K. and Riding, M. The TeraGyroid Experiment (2004). *Proceedings of GGF10*, Berlin, Germany, 2004: <http://www.cs.vu.nl/ggf/apps-rg/meetings/ggf10/TeraGyroid-Case-Study-GGF10-final.pdf>
- [12] Schneier, B. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, 2000.
- [13] Beckles, B., Brostoff, S., and Ballard, B. A first attempt: initial steps toward determining scientific users' requirements and appropriate security paradigms for computational grids (2004). *Proceedings of the Workshop on Requirements Capture for Collaboration in e-Science*, Edinburgh, UK, 14-15 January 2004, pp. 17-43: http://www.escience.cam.ac.uk/papers/req_analysis/first_attempt.pdf
- [14] Beckles, B. Kerberos: a usable grid authentication protocol (2006). *Designing for e-Science: Interrogating new scientific practice for usability, in the lab and beyond*, Edinburgh, UK, 26-27 January 2006.
- [15] Provos, N., Friedl, M. and Honeyman, P. Preventing Privilege Escalation (2003). *12th USENIX Security Symposium*, Washington, DC, USA, 2003: http://www.usenix.org/publications/library/proceedings/sec03/tech/provos_et_al.html
- [16] Lock, R., and Sommerville, I. Grid Security and its use of X.509 Certificates. *DIRC internal Conference* submission 2002. Lancaster DIRC, Lancaster University, 2002: <http://www.comp.lancs.ac.uk/computing/research/cseg/projects/dirc/papers/gridpaper.pdf>
- [17] Broadfoot, P. J. and Martin, A. P. A critical survey of grid security requirements and technologies (2003). Programming Research Group Research Report PRG-RR-03-15, Programming Research Group, Oxford University Computing Laboratory, August 2003: <http://web.comlab.ox.ac.uk/oucl/publications/tr/rr-03-15.html>
- [18] Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S. A security architecture for computational grids (1998). *Proceedings of the 5th ACM conference on Computer and communications security*, San Francisco, CA, 1998, pp.83-92: <http://portal.acm.org/citation.cfm?id=288111>
- [19] Basney, J., Humphrey, M. and Welch, V. The MyProxy online credential repository (2005). *Software: Practice and Experience* **35** (9):801-816, 25 July 2005: <http://dx.doi.org/10.1002/spe.688>
- [20] Chadwick, D.W. and Otenko, O. The PERMIS X.509 role based privilege management infrastructure (2002). *Proceedings of the 7th ACM symposium on Access control models and technologies*, Monterey, CA, 2002, pp. 135-140: <http://portal.acm.org/citation.cfm?id=507711.507732>
- [21] Brostoff, S., Sasse, M.A., Chadwick, D., Cunningham, J., Mbanaso, U. and Otenko, S. *R-What?* Development of a role-based access control policy-writing tool for e-Scientists. *Software: Practice and Experience* **35** (9):835-856, 25 July 2005: <http://dx.doi.org/10.1002/spe.691>
- [22] Zurko, M.E. and Simon, R.T., User-centered security (1996). *Proceedings of the 1996 workshop on New security paradigms*, Lake Arrowhead, CA, USA, 1996, pp. 27-33: <http://portal.acm.org/citation.cfm?id=304859>
- [23] Adams, A. and Sasse, M.A. Users are not the enemy: Why users compromise security mechanisms and how to take remedial measures (1999). *Communications of the ACM*, Volume 42, Issue 12 (December 1999), pp. 40-46: <http://portal.acm.org/citation.cfm?id=322806>
- [24] Gould, J.D. and Lewis, C. *Designing for Usability: Key Principles and What Designers Think* (1985). *Communications of the ACM*, Volume 28, Issue 3 (March 1985), pp. 300-311: <http://portal.acm.org/citation.cfm?id=3170>
- [25] Beyer, H. and Holtzblatt, K. *Contextual Design: Defining Customer-centered Systems*. Morgan Kaufmann Publishers, 1998.
- [26] Cooper, A. *The Inmates Are Running the Asylum: Why High-tech Products Drive Us Crazy and How to Restore the Sanity*. Sams, 1999.
- [27] Snelling, D.F., van den Berghe, S. and Li, V. Q. Explicit Trust Delegation: Security for Dynamic Grids. *FUJITSU Sci. Tech. J.* **40** (2) (December 2004), pp. 282-294: <http://www.unigrids.org/papers/explicittrust.pdf>
- [28] Beckles, B. Removing digital certificates from the end-user's experience of grid environments (2004). *Proceedings of the UK e-Science All Hands Meeting 2004*, Nottingham, UK, 31 August - 3 September 2004: <http://www.allhands.org.uk/2004/proceedings/papers/250.pdf>
- [29] Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S. A Community Authorization Service for Group Collaboration (2002). *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, Monterey, CA, 2002, pp. 50-59: http://www.globus.org/alliance/publications/papers/CAS_2002_Revised.pdf
- [30] Alfieri, R., Cecchini, R., Ciaschini, V., dell'Agnello, L., Frohner, A., Gianoli, A., Lorente, K.L., Spataro, F., VOMS, an Authorization System for Virtual Organizations (2003). 1st European Across Grids Conference, Santiago de Compostela, Spain, 13-14 February 2003: <http://grid-auth.infn.it/docs/VOMS-Santiago.pdf>
- [31] Shibboleth Project: <http://shibboleth.internet2.edu/>
- [32] Van, T. "Grid Stack: Security debrief". *Grid Stack*, 17 May 2005: <http://www-128.ibm.com/developerworks/grid/library/gr-gridstack1/index.html>
- [33] Yee, K-P. User interaction design for secure systems (2002). *Proceedings of the 4th International Conference on Information and Communications Security*, Singapore, 9-12 December 2002. Expanded version available at: <http://zesty.ca/sid>
- [34] Balfanz, D., Durfee, G., Grinter, R.E, Smetters, D.K. In search of usable security: Five lessons from the field (2004). *IEEE Security and Privacy* **2** (5) (September-October 2004), pp. 19-24: <http://doi.ieeecomputersociety.org/10.1109/MSP.2004.71>
- [35] Fléchaux, I., Sasse, M.A. and Hailes, S.M.V. Bringing Security Home: A process for developing secure and usable systems (2003). *Proceedings of the 2003 workshop on New security paradigms*, Switzerland, 2003, pp. 49-57: <http://portal.acm.org/citation.cfm?id=986664>
- [36] Cooper, A., and Reimann, R. *About Face 2.0: The Essentials of Interaction Design*. John Wiley and Sons, New York, 2003.
- [37] Sommerville, I. *An Integrated Approach to Dependability Requirements Engineering* (2003). *Proceedings of the 11th Safety-Critical Systems Symposium*, Bristol, UK, 2003.
- [38] Ryan, P.Y.A., Schneider, S.A., Goldsmith, M.H., Lowe, G., and Roscoe, A.W. *Modelling and Analysis of Security Protocols*. Addison-Wesley Professional, 2000.
- [39] Abdallah, A.E., Ryan, P.Y.A. and Schneider, S. (eds.). *Formal Aspects of Security*. Proceedings of the First International Conference, FASec 2002, London, December 2002. LNCS 2629, Springer-Verlag, 2003.
- [40] RealityGrid: <http://www.realitygrid.org/>
- [41] Beckles, B. User requirements for UK e-Science grid environments (2004). *Proceedings of the UK e-Science All Hands Meeting 2004*, Nottingham, UK, 31 August - 3 September 2004: <http://www.allhands.org.uk/2004/proceedings/papers/251.pdf>
- [42] Snyder, C. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann Publishers, London, 2003.