Document title	**QosCosGrid 2.6.1 middleware final security audit results**
Organizational unit	Wrocław Centre for Networking and Supercomputing (WCSS)
Organization	Wrocław University of Technology, Poland
Date of creation	21.02.2012
Last modified	18.04.2012
Testers	Bartłomiej Balcerek, Maciej Kotowicz

1. **Preface**

The QosCosGrid (QCG) middleware is an integrated system offering advanced job and resource management capabilities developed in **Poznań Supercumputing and Networking Center.** By connecting many distributed computing resources together, QCG offers highly efficient mapping, execution and monitoring capabilities for variety of applications, such as parameter sweep, workflows, MPI or hybrid MPI-OpenMP. Thanks to QosCosGrid, large-scale applications, multi-scale or complex computing models written in Fortran, C, C++ or Java can be automatically distributed over a network of computing resources with guaranteed QoS. The middleware provides also a set of unique features, such as advance reservation and co-allocation of distributed computing resources.

Wrocław Centre for Networking and Supercomputing (WCSS) acts as a Computer Security Incident Response Team (CSIRT) for National Grid Initiative. WCSS security team has identified a number of serious vulnerabilities in widely used Open Source and commercial software, in particular in grid middleware, e.g: CVE-2011-2193, CVE-2011-2907. A number of next CVE requests are pending at this moment (i.e. PBSPro remote root vulnerability and Maui Scheduler authorization bypass).

WCSS conducted a thorough security test procedure (security audit) of QCG suite**.** The procedure was iterating, i.e. after full security check a list of security flaws was sent to authors, which issue next software version as a input to next security tests. This report concerns final version of QCG middleware which is **2.6.1**.

Although during the intermediate test steps a number of security flaws was found, the final versions of QCG is considered to be secure and could by recommended to use in EGI infrastructure.

2. Testbed

For testing purposed two identical testbeds were prepared. Each of the systems was a virtual 64 bit machine of hardware configuration:

- 2 x Intel(R) Xeon(R) CPU X3220 @ 2.40GHz
- 1 GB RAM
- 4GB disk space

Each machine was operating on Scientific Linux SL release 5.6 (Boron). Kernel and essential packages version follows:

- Linux 2.6.18-274.12.1.el5
- globus-gsi-proxy-ssl-2.3-3
- globus-gsi-openssl-error-0.14-8
- globus-gsi-callback-2.8-2
- globus-callout-0.7-8
- globus-openssl-5.1-2
- globus-common-11.6-5
- globus-gsi-sysconfig-3.2-1
- globus-gsi-cert-utils-6.7-2

- globus-gsi-credential-3.5-3
- globus-gssapi-gsi-7.8-1
- globus-gss-assist-5.10-1
- globus-libtool-1.2-4
- globus-openssl-module-1.3-3
- globus-gsi-proxy-core-4.7-2
- globus-proxy-utils-3.10-1
- openssl-0.9.8e-12
- **qcg-comp-client-2.6.1-8.x86_64**
- **qcg-comp-schemas-2.6.1-8.x86_64**
- **qcg-comp-2.6.1-8.x86_64**
- **qcg-openmpi-1.3.1-2.x86_64**
- **qcg-ntf-2.6.1-1.x86_64**
- **qcg-core-2.6.1-3.x86_64**
- **qcg-dep-1.0.1-2.x86_64**

3. Tools and methodology

A two different approaches to find security flaws were applied. One approach was static code analysis, the second consisted of number of dynamic tests e.g. tests on running applications.

The manual review was oriented to develop a detailed data flow within the system, especially investigating the flow of data passed from the client, which must not be trusted. Several additional tests were performed using the following tools:

- RATS (Rough Auditing Tool for Security)
- Flawfinder
- Splint
- cppcheck

For dynamic tests a publicly available tools were used, and in addition some specific tools were developed. As this tests relied primarily on fuzzing methods, following fuzzing tools were engaged:

- zzuff
- Radamsa
- proxyfuzz
- XmlFuzz[1]
- MemFuzz[2]

During dynamic testing gdb (Gnu Debugger) and strace (System Trace) tools were intensely used.

1 Collection of private fuzzers, designed to generate malformed input suitable for QCG.

2 Private in memory fuzzer combined with XmlFuzz.

4. Test areas and results

1. Memory management (proper use of memory allocation and release)

No memory corruption or control flaw abuse (leading to use-after-free or double free errors) were found.

2. Buffer operations (proper use of functions operating on stack and heap buffers)

All buffers are handled in secure manner.

3. Counter operations (proper use of integer variables, preventing possible overflows, underflows and improper casting)

No flaws has been found.

4. Command injection (proper data validation, character whitelisting)

Shell meta-characters are escaped when passed to system(3)-like functions.

5. SQL injection (proper data validation, character whitelisting preserving types across data)

Proper prevention techniques are engaged. No flaws has been identified.

6. XML External Entity Attack (proper handling of XML features)

Parser does not allow to execute external entities.

7. Format string manipulations

No format strings are allowed from untrusted source.

8. Integer types casting

No improper casting found (signedness or size).

9. File manipulation (safe file handling, Time of Test Time of Use, rigorous file permissions, temporary files management)

Files operations are conducted on file descriptors - not on paths. When file is created a least premissive permissions are assigned.

10. Signal handling

No flaws has been found.

11. XPath/XQuery injection

No flaws has been found.

12. Transfer encryption

All QCG components use secure protocols to exchange data, i.e.: SSL3, TLS1 or GSI. Only secure cipher suites are allowed by the server.

13. Proxy certificate authorization and authentication

Valid proxy is needed (time-valid and non-revoked) to run a job. QCG-compd does not allow to run jobs with limited, independent or restricted proxy.

14. Server-client mutual authentication

Both client and server authenticate each other proper way.

15. Binary level hardening

Efficient hardening has been provided at binary level: stack overwrite and Fortify Source protections.

5. Summary

There were not any really significant security flaws in tested version of QosCosGrid software suite found. Very most of the code was written with security-in-mind and security well practices were applied. Software has been hardened at binary level, therefore even if any part of the software suffers overlooked implementation errors, exploitation will be extremely difficult. Authors also provide a good understanding of grid specific security protocols and methods. We approve **QosCosGrid 2.6.1** for production use.

Bartlomiej Balcerek

[illegible]

Maciej Kotowicz

[illegible]

6. Contact data

Postal Address:
Wroclawskie Centrum Sieciowo-Superkomputerowe
Wybrzeze Wyspianskiego 27
50-370 Wroclaw POLAND

Courier Office:
Plac Grunwaldzki 9 building D-2 PWr, room 101
50-377 Wroclaw

phone (+48 71) 320 20 79
fax (+48 71) 322 57 97

WWW: http://www.wcss.wroc.pl/english/

Email: bartol@pwr.wroc.pl, maciej.kotowicz@pwr.wroc.pl

7. References

[1]. Seacord, R. "Secure Coding in C and C++". Upper Saddle River, NJ: Addison-Wesley, 2006 (ISBN 0321335724).

[2]. M. Dowd, J McDonald, J Schuh. The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities. Upper Saddle River, NJ: Addison-Wesley, 2006 (ISBN 0321444426).

[3]. M. Sutton, A. Greene, P. Ammini. Fuzzing: Brute Force Vulnerability Discovery. Addison-Wesley Professional; 1 edition (July 9, 2007) (ISBN 0321446119)

[4]. A. Taken, J. DeMott, C. Miller. Fuzzing for Software Security Testing and Quality Assurance. Artech House Print on Demand; 1 edition (June 30, 2008)(ISBN 1596932147)

[5]. David A. Wheeler "Secure Programming for Linux and Unix HOWTO", http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/index.html

[6]. Bartłomiej Balcerek, Gerard Frankowski, Agnieszka Kwiecień, Adam Smutnicki, Marcin Teodorczyk „Security best practices: applying defense-in-depth strategy to protect the NGI". Building a National Distributed e-Infrastructure–PL-Grid Lecture Notes in Computer Science, 2012

[7]. S.Herzog .XML External Entity Attacks (XXE). OWASP 20.10.2010, https://www.owasp.org/images/5/5d/XML_Exteral_Entity_Attack.pdf

[8]. The METASM assembly manipulation suite. http://metasm.cr0.org/

[9]. zzuf - multi-purpose fuzzer. http://caca.zoy.org/wiki/zzuf

[10]. Radamsa. https://www.ee.oulu.fi/research/ouspg/Radamsa