Security Risks in Clouds and Grids

Elisa Heymann
Computer Architecture and Operating Systems Department
Universitat Autònoma de Barcelona
Elisa.Heymann@uab.es

Barton P. Miller
James A. Kupsch
Computer Sciences Department
University of Wisconsin
bart@cs.wisc.edu

Condor Week
May 5, 2011
Efficient execution of SPMD Applications on Multicore Environments

Problem

Hierarchical communication architecture

SPMD Application Message Passing Interface

Objective

- Maximum Speedup
- Efficiency over a defined threshold

How?

Methodology

- Maximum Speedup
- Efficiency over a defined threshold

Ideal Size of Supertile

Ideal Number of Core

SuperTile allows to overlap the internal computation with edge communication
Possible Threat?

- Clouds and Grids have databases with management and operational information
- Denial of Service:
  - Prevent updates in the database
Possible Threat?

> Hijack machines
  - Process escapes Cloud/Grid/control: Keeps forking and exiting to escape detection.

![Diagram showing process forking and exiting](image-url)
Possible Threat?

› Cloud/Grid Accounting System
  - Maintains a Grid-wide view of resource utilization.
    • Job Submission (Priority in the batch queue, CPU time, Memory usage)
    • Storage (Disk usage, Tape storage)
  - Accounting Information easily available to people (web interface) and to applications (Web Services)

› Use the Accounting System for bad purposes.
Possible Threat?

```
[rohit@localhost ~]$ su 'r.TimeDuration(' 
sh-3.2# 
sh-3.2# 
sh-3.2# chfn 
Changing finger information for root. 
Name [root]: 
```
Real Threat!

GRATIA-CONDOR-2010-0003

Summary:
Any user that can submit Condor jobs on the host running Gratia Condor probe, can execute arbitrary code as the root user.

<table>
<thead>
<tr>
<th>Component</th>
<th>Vulnerable Versions</th>
<th>Platform</th>
<th>Availability</th>
<th>Fix Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>condor_meter</td>
<td>1.04.4d-1</td>
<td>all</td>
<td>not known to be publicly available</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>1.06.13b-1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status</th>
<th>Access Required</th>
<th>Host Type Required</th>
<th>Effort Required</th>
<th>Impact/Consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verified</td>
<td>local ordinary user with Condor submission privilege</td>
<td>any</td>
<td>low</td>
<td>high</td>
</tr>
</tbody>
</table>

Fixed Date
Credit
Rohit Koul

Access Required:
local ordinary user with Condor submission privilege

The vulnerability requires local access to the machine and the ability to submit Condor jobs.

Effort Required:
low

Exploiting this vulnerability requires the attacker to submit a Condor job with unusual job attributes in the submit script.
What the bad guys can do

- Gain root access
- Privilege escalation
  - Gain higher privilege access (admin, condor)
- Hijack machines
  - Attack the process running there
What the bad guys can do

- Injections
  - Command
  - SQL

1. String user = request.getParameter("user");
2. String password = request.getParameter("password");
3. String sql = "select * from user where username=' " + user + " ' and password=' " + password + " ' " + ' or '1'='1'--";
What the bad guys can do

> Injections
  - Command
  - SQL

---

1. Hi, this is your son’s school.
   We’re having some computer trouble.

2. Oh, dear – did he break something?
   In a way...

3. Did you really name your son
   Robert’); DROP TABLE Students; -- ?

4. Oh, yes. Little Bobby Tables,
   we call him.

5. Well, we’ve lost this year’s student records.
   I hope you’re happy.

And I hope you’ve learned to sanitize your database inputs.
What the bad guys can do

› Injections
  - Command
  - SQL
  - Directory traversal
  - Log

› Denial of Service (DoS)
Why do we care

- Machines belonging to a cloud/grid site are accessible from the Internet
- Hundred of thousands of machines are appealing
- Those machines are continuously probed:
  - Attackers trying to brute-force passwords
  - Attackers trying to break Web applications
  - Attackers trying to break into servers and obtain administrator rights
Why do we do it

- SW has **vulnerabilities**
- Cloud and Grid SW is **complex and large**
- Vulnerabilities can be exploited by legal users or by others
Why do we do it

- **Attacker** chooses the time, place, method, ...

- **Defender** needs to protect against all possible attacks (currently known, and those yet to be discovered)
Key Issues for Security

› Need independent assessment
  - Software engineers have long known that testing groups must be independent of development groups

› Need an assessment process that is NOT based solely on known vulnerabilities
  - Such approaches will not find new types and variations of attacks
Our Piece of the Solution Space

First Principles Vulnerability Assessment:

› An analyst-centric (manual) assessment process.
› You can’t look carefully at every line of code so:

  Don’t start with known threats ...
  ... instead, identify high value assets in the code and work outward to derive threats.

• Start with architectural analysis, then identify key resources and privilege levels, component interactions and trust delegation, then focused component analysis.
First Principles Vulnerability Assessment
Understanding the System

Step 1: Architectural Analysis

- Functionality and structure of the system, major components (modules, threads, processes), communication channels
- Interactions among components and with users
Architectural Analysis: Condor

1. fork
2. machine ClassAd
3. submit job ClassAd
4. job ClassAd
5. Negotiator cycle
6. Report match
7. claim host
8. fork
9. establish channel
10. start job

Master

Stork server host

OS privileges
- condor & root
- user
First Principles Vulnerability Assessment
Understanding the System

Step 2: Resource Identification
- Key resources accessed by each component
- Operations allowed on those resources

Step 3: Trust & Privilege Analysis
- How components are protected and who can access them
- Privilege level at which each component runs
- Trust delegation
Resource Analysis: Condor

(a) Common Resources on All Condor Hosts

- generic Condor daemon
- Condor Binaries & Libraries
- Condor Config
- Operational Data & Run-time Config Files
- Operational Log Files

(b) Unique Condor Checkpoint Server Resources

- OS privileges
  - condor
  - root
  - user

- ckpt_server
  - Checkpoint Directory
  - Send and Receive Checkpoints (with Standard Universe Jobs)

(c) Unique Condor Execute Resources

- User Job
- starter
  - Job Execution Directories

(d) Unique Condor Submit Resources

- shadow
  - User’s Files

System Call Forwarding and Remove I/O (with Standard Universe Jobs)
First Principles Vulnerability Assessment
Search for Vulnerabilities

Step 4: Component Evaluation
- Examine critical components in depth
- Guide search using:
  - Diagrams from steps 1-3
  - Knowledge of vulnerabilities
- Helped by Automated scanning tools
First Principles Vulnerability Assessment
Taking Actions

Step 5: Dissemination of Results
- Report vulnerabilities
- Interaction with developers
- Disclosure of vulnerabilities
Our Experience

**Condor**, University of Wisconsin
Batch queuing workload management system
15 vulnerabilities
600 KLOC of C and C++

**SRB**, SDSC
Storage Resource Broker - data grid
5 vulnerabilities
280 KLOC of C

**MyProxy**, NCSA
Credential Management System
5 vulnerabilities
25 KLOC of C

**glExec**, Nikhef
Identity mapping service
5 vulnerabilities
48 KLOC of C

**Gratia Condor Probe**, FNAL and Open Science Grid
Feeds Condor Usage into Gratia Accounting System
3 vulnerabilities
1.7 KLOC of Perl and Bash

**Condor Quill**, University of Wisconsin
DBMS Storage of Condor Operational and Historical Data
6 vulnerabilities
7.9 KLOC of C and C++
Our Experience

**Wireshark**, wireshark.org
Network Protocol Analyzer in progress 2400 KLOC of C

**Condor Privilege Separation**, Univ. of Wisconsin
Restricted Identity Switching Module 21 KLOC of C and C++

**VOMS Admin, INFN**
Web management interface to VOMS data 35 KLOC of Java and PHP

**CrossBroker**, Universitat Autònoma de Barcelona
Resource Mgr for Parallel & Interactive Applications 97 KLOC of C++
Our Experience

ARGUS 1.2, HIP, INFN, NIKHEF, SWITCH
gLite Authorization Service
in progress

glExec 0.8, Nikhef
Identity mapping service
What do we do

› Make cloud/grid software more secure
› Make in-depth assessments more automated
› Teach tutorials for users, developers, admin, managers:
  - Security risks
  - Vulnerability assessment
  - Secure programming
Who we are

Bart Miller
Jim Kupsch
Karl Mazurak
Rohit Koul
Daniel Crowell
Wenbin Fang

Elisa Heymann
Eduardo Cesar
Jairo Serrano
Guifrè Ruiz
Manuel Brugnoli
Security Risks in Clouds and Grids

Elisa Heymann
Elisa.Heymann@uab.es

Barton P. Miller
James A. Kupsch
bart@cs.wisc.edu

http://www.cs.wisc.edu/mist/