

# **Grid software from a security perspective**

**3<sup>rd</sup> TERENA NREN-GRIDS Workshop**

**27./28. April 2006**

**DFN-CERT Services GmbH**

**Klaus Möller**

## Why do we do this ?

- DFN-CERT is the CSIRT for the german academic community
- Investigation of the effects of Grids on our constituency
  - Organisational
  - Technical, i.e. the software
- Work in progress, early stage
- No answers yet but many questions

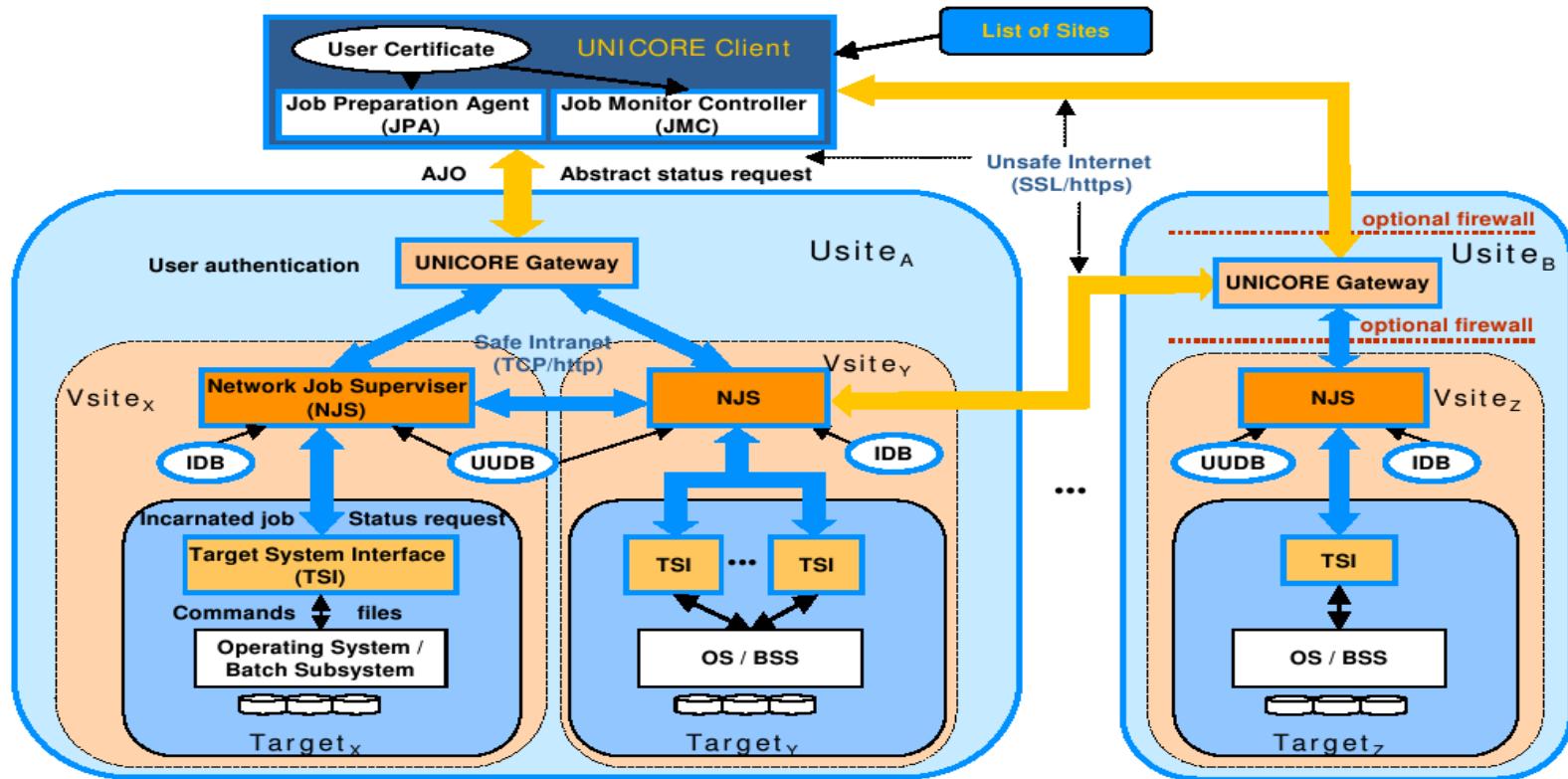
- All the software needed to offer (or use) Grids services
  - Focus on what will be used in D-Grid
- Operating systems
  - Windows (clients only), Linux, other Unix
- Supporting software
  - Condor, GSI OpenSSH , Apache, databases, etc.
- Grid middleware
  - Globus, gLite, UNICORE
- Grid applications
  - Each Grid has its own . . .

## Looking for vulnerabilities

- Three broad categories
  - Design weaknesses
    - Clear text protocols, man-in-the-middle, ...
  - Programming errors
    - Buffer Overflows, Format String Bugs, Race conditions, ...
  - Configuration errors
    - No/weak passwords, unneeded services, ...
- Not every vulnerability may be exploitable
  - Circumstances may prevent exploitation



## UNI CORE Architecture



26.3.2003

Technical Board of the UNICORE Forum e.V.

7

## Possible attacks on the UNICORE client

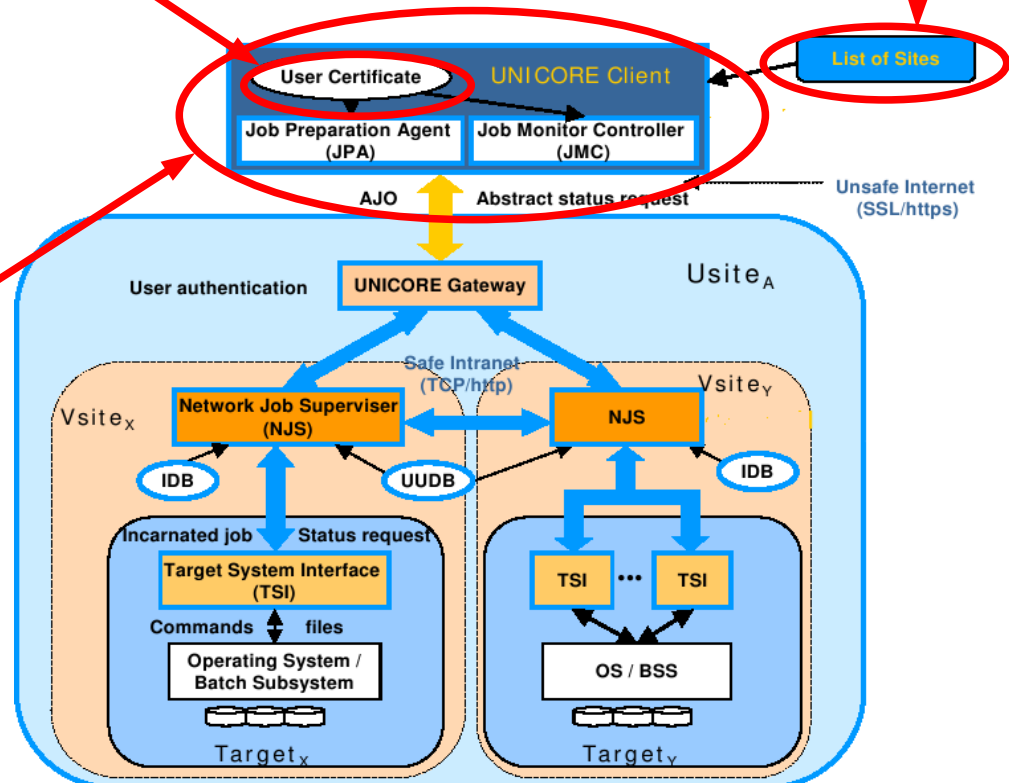
Obtain local private key

- Brute-force passphrase
- Sniff passphrase with keystroke logger

Add Sites to local site DB  
(Phishing style attack)

Manipulate Client

- To submit different job
- Don't show output of manipulated job



## UNICORE Gateway

- UNICORE sites are only accessible through the UNICORE gateway
- Ways to attack a site
  - Use stolen credentials
  - Attack the gateway, then the site from there
    - Exploit vulnerabilities in SSL software (openssl)
    - Will strike before authentication takes place
  - Bypass the gateway
    - Can SQL-injection style attacks be done ?
    - Data injection attack on SSL connections ?

## Possible attacks on the UNICORE site

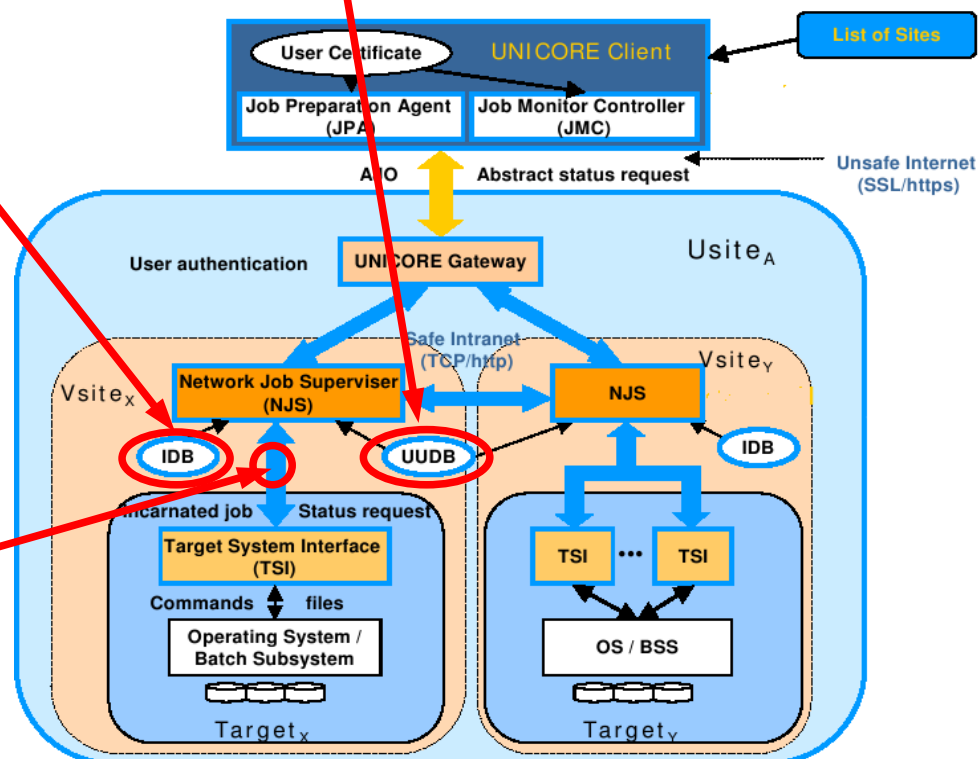
Manipulate User Mapping Database

- Authenticate as different (privileged ?) local user

Manipulate job mapping

- Add Code to legitimate jobs
- Exploit code against other users

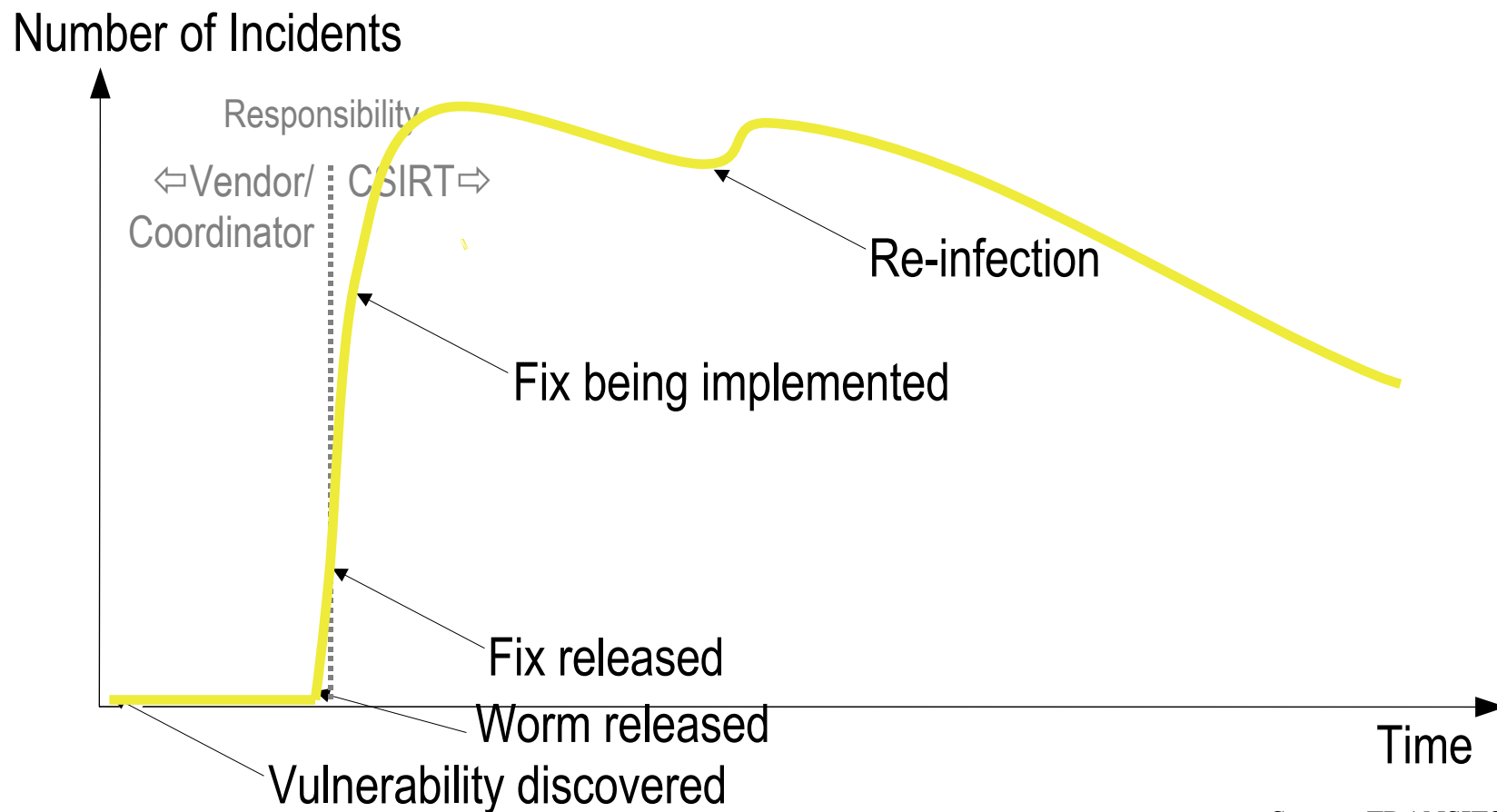
Sniff unencrypted traffic within site



## How many ?

- Attempt to guess a number
  - gLite 1: ~343000 lines of C/C++ code
    - Source: <https://edms.cern.ch/file/581354/1/EGEE-JRA1-PRE-581354-GLITE-180405.pdf>
  - Number of bugs Open Source projects per 1000 lines of code: 0.32 – 0.69
    - Source: <http://scan.coverity.com/>
  - That would be 109 - 236 bugs in code that has been tested and released
- Security Audits ?
  - Amount of required work seems to make this unfeasible

## Influence on incidents over time



Source: TRANSITS

- Information sources
  - Incident reports
  - Hackers / full disclosure community
  - Vendors
  - Commercial services
- For site admins
  - Multiply by number of software packages
  - Multiply by number of hosts / Grids
- Almost impossible to keep track of
  - Scale down to rely on a few trusted sources
  - I.e. security advisores from vendors and CSIRTs

## For Grid software

- Operating systems
  - Standard point of contact for reporting vulnerabilities
  - Public security announcement lists
  - PGP signed advisories
- Middleware
  - Points of contact not commonly known
  - Only Globus has public security announcement list
  - Non signed advisories

## What is needed ?

- Timely information
- Useful information
  - Complete – source and impact of vulnerability
  - Correct
  - Concrete help to fix the vulnerability
- New developments towards joint development of advisories
  - Spreads workload
  - May improve quality

## Standards projects

- Projects for the exchange of advisories
  - CVE: standard reference numbers for vulnerabilities
  - EISPP: advisory format for teams cooperating to write/issue notices
  - CAIF: XML format for structured notices
  - OASIS AVDL: vulnerability description and test project
  - VEDEF: mapping the process and specify requirements

## CVSS

- Common Vulnerability Scoring System
- Rate relative seriousness of vulnerabilities
  - Helps prioritisation at all stages
- Score based on three types of factor
  - Base: unchanging, set by vendor
    - How is vulnerable system reached ? Type of impact ?
  - Temporal: changes in time, set by vendor
    - Availability of exploit; availability of remedy
  - Environmental: set by end user
    - How common is target system ? How severe is potential

## Patching

- Automatic updates ?
  - Standard for OS and some vendor SW
  - Requires additional work for other software
- Grid software may fail due to dependencies
  - Test before deployment – procedures ?
- What if patch breaks Grid software ?
  - Keep old software
    - System stays vulnerable, maybe break later updates
  - Keep copy of old software for Grid
    - And live with the vulnerabilities

## The weakest link ?

- The easiest attack on software ?
  - Compromise the source
    - Malicious developer
    - External attacker
  - Or crack the distribution site
- Has happened in the past
  - OpenSSH/OpenBSD, Linux Kernel, tcpdump, Debian, ...
  - Some got as far as to insert their code into the code base !

## How secure ?

- Mitigation
  - Secure the development/distribution sites
  - Integrity protection
    - MD5/SHA-1 checksums
    - Better: PGP signature, signed RPMs
- Generally not done for Grid middleware
  - Only Globus uses checksums (SHA-1)
  - Protecting the transport channel is not enough, so HTTPS downloads are no substitute

**Thank you !**

**Questions ?**