Table of Contents

PanDA Security

Overview

Panda services that interact with other Panda components and with external clients use the standard GSI grid security model of authentication and authorization based on X509 grid certificates, implemented via OpenSSL and https. mod_gridsite is used in Panda Apache servers to support this. Clients use curl for http and https support.

Interactions with Panda that involve anything other than passive read actions require secure https. Any user of the system (e.g. job submission via client interface, pilot requesting work) must hold a valid certificate proxy. Panda also checks the VOMS attributes of the proxy to ensure that the user is a member of a VO authorized for Panda use. The DN of the authenticated user is carried as part of the metadata of a Panda job, so user identity is known and tracked throughout Panda operations. Panda-internal accounting and quota/usage management also uses this information.

Production job execution and file management in Panda relies on production certificates, with the pilot carrying a specific 'pilot' VOMS role to control its rights. Analysis jobs also run under a production proxy unless glexec is employed in identity switching mode (see the glexec discussion).

We have also defined a means of securing the job workload specification (transformation) from tampering in the Panda DB (encryption of the transformation using RSA key pair, with decoding/validation in the pilot prior to execution).

Further security measures, existing and planned, are described in the security review answers below.

Answers to questions from the pilot job framework security review, Version 2.5.3, 11/18/2008

0. Describe in a schematic way all the components of the system. If a component needs to use IPC to talk to another component for any reason, describe what kind of authentication, authorization, integrity and/or privacy mechanisms are in place.

Panda's framework is based on the concept of a pilot job. According to it, payload job never gets submitted to the target Grid site directly. Instead, a precursor job (the pilot) is submitted from a select account (or a number of accounts), and after probing the OS and network environment, and optionally performing other settings, contacts the Panda server in order to determine the location of the payload definition. At this point, the job is marked by the server as active. The payload then gets executed (after being downloaded) and the pilot contacts the Panda server to register the job's completion. Note that unless special provisions are made, the identity associated with the payload being executed will be inherited from the pilot job. However, the mechanisms described later in this document provide for a proper identity switch to rectify that.

The code of the pilot job is generic in the sense that it is in no way coupled to a particular submission mechanism. As such, it can be launched using Condor or any batch management system or even interactively.

Specific components of the Panda system are as follows (for more information see the PanDA main wiki):

a) The "Panda Server", which itself includes the following: i) Apache-based Web interface ii) Backend databases which contain job data and various metadata

It performs the following functions:

- accepting job submissions via https from python front-end client (see c)
- managing the system-wide job queue
- performing brokerage on the queue based on job assignment criteria such as data locality, job priority, realtime availability of processing resources
- pre-staging input data to sites or out of mass store prior to release of jobs, via an associated data service
- dispatching of jobs to pilots making job requests, based on brokerage decisions

b) The Panda Monitor, which is loosely coupled to (a) via its network connection to the databases and whose functionality is to display various details on the execution of jobs and data movement

c) A few variants of the client software (eg. the physics analysis interface called *"pathena"*) which connects to the Panda server via https and is secured via a X509 proxy; the function of this component (c) is to register a job request with the server (a).

The request itself does not contain the payload job definition, and instead contains a reference to a URL specified by the user requesting the job, pointing to a Web service from which the job definition should be obtained. Currently, there are no restrictions on the exact location of such definition.

Another variant of the job submission front end, Bamboo, interfaces Panda to the ATLAS production database which records jobs requested by the physics community and operates automatically. Ultimately, its functionality is similar to *pathena*, which is to register a job request (or rather, multiple job requests) with the Panda server. The main difference is that *pathena* is a client application run by each end user doing analysis from their workstation, while Bamboo acts as automated agent driven by entries in the job database, and is essentially a centralized service.

d) The Pilot Scheduler, with the main version being called "Autopilot", which governs submission of pilot jobs to target sites (condor-G is the job submitter in this case; another scheduler being developed uses gLite job submission tools)

e) The Pilot Job, originated on the target site by (d), which after launching on a worker node connects to (a) via https to request a payload job, manages site-local data handling and job execution, and reports job status and results to (a)

f) MyProxy server which is used to cache users' X509 credentials when necessary. Entities who contact the server include the pathena client (c) and the Pilot Job (e), with the former depositing the user's proxy and the latter retrieving it as necessary (more on that in item "1" below). The server relies on TCP/IP with the Transport Layer Security (TLS) protocol for its communication needs.

As is explained in the item below, there are additional enhancements of the Pilot Job security (some already implemented and others work in progress), which involve use of tokens cached in the database of the Panda server. In terms of IPC, this implies communication of both Pilot Scheduler and Pilot Job with the Panda Service, accomplished via secure HTTP.

1. Describe how user proxies are handled from the moment a user submits a task to the central task queue to the moment that the user task runs on a WN, through any intermediate storage.

As explained in item 2 (below), on sites specifically equipped for such procedure, the Pilot will change the identity under which the payload job is run, from that of the pilot submitter (who carries the "role=pilot" annotation) to UID of the end user, via the *glexec* privileged executable.

Assuming this is the case, the user proxies are handled in the following manner: in the distributed analysis scenario, when the user submits a job to the Panda system, the client software (pathena) will check if the user already has a proxy deposited on a dedicated MyProxy server, which is owned and managed by the Atlas organization, and that this instance has sufficiently long remaining period of validity (that criterion is configurable). If that's not the case, the client will deposit a new user proxy onto the MyProxy server. On the other hand, the Pilot Job (and the Pilot Job only) has the credentials to extract the concrete user's proxy from the MyProxy server, a process which happens once the Pilot connects to the Panda server and is ready to execute the Transformation (the script defining the actual payload). Working under the assumption that we limit the lifetime of the user's proxy, the Pilot Job is responsible for its periodic renewal during the job's execution.

Possible security concerns related to misuse of a compromised pilot proxy are being addressed in the following way: an instance of the proxy being stored the MyProxy server is assigned a unique key that will be required upon future retrieval. The key is stored on the Panda server. This is in addition to requiring X509 credentials from the client for authentication. The software component that does that has been implemented. This key is generated as a uuid string.

In addition, we use single (or few) use tokens that will be used by the Pilot Job in order to get a payload job from the Server. The key is to be generated by the pilot scheduler, which will store it in a database (together with the scheduler's own validation token) as well as send it along with the submitted pilot. The pilot in turn presents the token to the Panda server in the job request process. The server retrieves the token (and associated scheduler token) from the DB, so validating the pilot, and immediately deletes the token (or decrements a maximum usage count).

Note that the proxy itself is never cached in Panda. In this scheme, the ways for an intruder to take possession of a user's proxy are:

(a) to gain access to the Panda database, which is protected - in addition to the pre-requisite of having a valid pilot proxy. We consider this level of security breach highly unlikely as it involves intrusion of a few loosely coupled components, each with its own set of credentials, and would require that all of these be simultaneously compromised.

(b) to directly penetrate the Worker Node and impersonate pilot jobs. In case a Worker Node is compromised during a limited period (say a few hours or days), the attacker might try and imitate the pilot(s), requesting a series of jobs to obtain their associated user proxies, but the constraints on the token reuse ensure that for a given pilot any such abuse can only affect a small number of users. The attacker might kill the pilot itself, so that another job with another proxy (and possibly a token) gets started, but such an attack in itself is not specific to pilot-based workload management systems.

(c) to gain direct access to the pilot job scheduler. This kind of threat is not unique in that there are already components in the grid infrastructure which, if compromised, could provide the intruder with credentials of multiple users. Such a risk is minimized by having a very small number of machines performing this task, together with establishing a strict control over access to these. Indeed, this is exactly the current practice.

One additional concern about the security of the proxy stems from the fact that in case of an abnormal job termination (such as when killed by site administrator) the proxy cached on the worker node is not immediately destroyed. Note, however, that to gain access to such proxy one needs to possess the identity of the pilot job. This effectively reduces such threat to case "b" in the list above. In addition,the lifetime of the proxy is limited (see item 7).

2. What happens around the identity change on the WN, e.g. how is each task sandboxed and to what extent?

This identity change via glexec is an optional feature of Panda, activated only at sites which request its use. Such condition is reflected in the site metadata recorded in the Panda server's database. In cases where the use of glexec is not required, the corresponding part of logic in the pilot job code is bypassed, and glexec is not actuated. The security implications of such scenario are as follows: in case of a security incident, it makes it harder to immediately attribute suspect jobs already running on a grid site, to their end user. Note however, that it is still possible to accomplish that by using logs kept on the Panda server itself.

In the following, we concentrate on the case when the identity change does take place.

The Pilot Job is started by a privileged user (who has the *role=pilot* VOMS role annotation), whose rights in this case are limited to being able to utilize glexec. This is achieved by mapping the user (based on the aforementioned VOMS annotation contained in the proxy) to an account for which glexec is explicitly configured and limiting other types of access for this account as necessary (cf the difference between *pilot* and *production* roles).

The glexec utility is activated and uses the user's proxy obtained from the MyProxy server as explained in item 1. Effectively, each task is sandboxed by being executed under the end user account to which the user's proxy is mapped, with privileges limited to those of that account and proxy.

3. How can running processes be accounted to the correct user?

As explained in the previous item, the identity switch via glexec leads to each user's processes running under specific UIDs traceable to their respective identities. At the Panda level, the DN of the job submitter is recorded permanently for each Panda job, such that Panda can trace and account usage.

4. How is a task spawned on the WN and how is it destroyed?

Pilot Jobs download the so-called "transformation" script (based on the job definition they obtain from the Panda server) which is the payload job script. The transformation script will be executed just like any other script submitted to the local batch queue, and its spawning and destruction would be identical to that. A process can either complete its execution normally, exit with errors or be terminated by the site administrator in case of a suspected security breach. Panda also has a mechanism by which Panda users and operators can request job termination: the Panda server messages the pilot managing the payload job to terminate the job.

5. How can a site be blocked?

There are two different kinds of site blocking.

(a) blocking the delivery and execution of the payload by the Panda server, on the suspect site. This is achieved by using database tables which support the token-based access control described above (in the second half of item 1). The schema for the table storing the tokens will also contain the DN of the pilot scheduler operator and pilot destination site. By disabling database entries corresponding to suspect site(s), we will be able to deny the delivery of jobs to suspect pilots on one or any number of participating sites. This functionality is work in progress.

(b) stopping submission of legitimate Panda Pilots to suspect sites. This is achieved by Panda maintaining a database of grid sites, which are kept in logical groups, that can be reconfigured at any time. Based on information stored in the database, Pilot submission is always targeting only sites chosen and approved for the specific set of tasks. It is trivial, therefore, to exclude a suspect site from Panda usage by modifying the relevant database entry. This functionality already exists in Panda.

6. Security of services

6.1 What site security processes are applied to the machine(s) running the WMS?

The machines running the Panda WMS at BNL are operated by personnel of "RHIC and ATLAS Computing Facility" (RACF) at Brookhaven National Laboratory and are subject to policies and processes specific to BNL, which comply with regulations set forth by the US Department of Energy. Likewise, instances of the Panda service deployed at CERN are subject to its security policies and processes.

6.2 Who is allowed access to the machine(s) on which the service(s) run, and how do they obtain access? How are authorized individuals authenticated on the machine(s)?

Only a limited number of qualified IT personnel are granted access to this machines, as defined by the management.

6.3 What is the process for keeping the service(s) and OS patched and up-to-date, especially with respect to security patches?

See 6.1

6.4 Do you have an identified security contact?

RACF does have the security contact as per 6.1

6.5 Describe the incident response plan to deal with security incidents and reports of unauthorized use?

See 6.1 - the Panda services are subject to a wider set of security policy implementations, and watched over by BNL cybersecurity specialists

6.6 What services (in general) run on the machine(s) that offer the WMS service?

Only Panda services run on the machines (apart from associated middleware, such as *Condor* for machines supporting pilot schedulers). Panda database services are hosted on machines that run no services other than the DB services (which currently use MySQL and will be migrated to Oracle in future).

6.7 What processes exist to maintain audit logs (e.g. for use during an incident)?

All crucial components of Panda WMS maintain extensive log files, which are periodically rotated and archived for possible future use in case of an incident.

6.8 What monitoring exists on the machine(s) to aid detection of security incidents or unauthorized use?

There are a few layers of monitoring:

- a web interface displaying the workload submitted by each user, including job summaries over specified time periods or job types, with drill-down to complete job definition and status details. Panda usage patterns are thereby transparent to both developers and the shift crews looking after production.
- as mentioned in 6.7, there are extensive logs that can be consulted to aid in detection of a possible security incident

7. Can you limit the users that can submit jobs to the VO WMS? How?

There are a few layers of user access control in Panda

- a DB table of Panda user attributes can be used to ban specific users from having their jobs executed by Panda

- job submitters are checked (via their DN) for affiliation with a VO as defined in the VOMS system (VOMS data cached periodically), and jobs are rejected if there is no valid affiliation with a VO approved for Panda use
- X509 proxies used in Panda have limited lifetimes (typically of 12 hour duration), so in case of a certificate revocation a user would eventually lose job submission rights. Note however that this process is not instantaneous.

Major updates:
-- MaximPotekhin - 18 Nov 2008
-- TorreWenaus, MaximPotekhin - 06 Aug 2008
-- TorreWenaus - 06 Oct 2006

Responsible: TorreWenaus

This topic: PanDA > PandaSecurity
Topic revision: r5 - 2009-04-27 - JoseCaballero