

Re-designing the PhEDEx Security Model

This content has been downloaded from IOPscience. Please scroll down to see the full text.

2014 J. Phys.: Conf. Ser. 513 042051

(<http://iopscience.iop.org/1742-6596/513/4/042051>)

View [the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 131.225.23.169

This content was downloaded on 26/11/2014 at 20:35

Please note that [terms and conditions apply](#).

Re-designing the PhEDEx Security Model

Huang C-H¹, T Wildish² and Zhang X³

¹ Fermi National Accelerator Laboratory, Batavia, IL, USA

² Princeton University, Princeton, NJ, USA

³ Institute of High Energy Physics, Beijing, China

E-mail: awildish@princeton.edu

Abstract. PhEDEx, the data-placement tool used by the CMS experiment at the LHC, was conceived in a more trusting time. The security model provided a safe environment for site agents and operators, but offered little more protection than that. Data was not sufficiently protected against loss caused by operator error or software bugs or by deliberate manipulation of the database. Operators were given high levels of access to the database, beyond what was actually needed to accomplish their tasks. This exposed them to the risk of suspicion should an incident occur. Multiple implementations of the security model led to difficulties maintaining code, which can lead to degradation of security over time.

In order to meet the simultaneous goals of protecting CMS data, protecting the operators from undue exposure to risk, increasing monitoring capabilities and improving maintainability of the security model, the PhEDEx security model was redesigned and re-implemented. Security was moved from the application layer into the database itself, fine-grained access roles were established, and tools and procedures created to control the evolution of the security model over time. In this paper we describe this work, we describe the deployment of the new security model, and we show how these enhancements improve security on several fronts simultaneously.

1. Introduction

PhEDEx [1] is the data-placement tool used by CMS, it is the definitive source of where the CMS experiments data resides. It manages the flow of raw experiment data from the Tier-0 to storage at the Tier-1 sites, it moves Monte Carlo data from the Tier-2s where it is produced to permanent storage at a Tier-1, and then on from there for analysis. It checks the consistency of data at sites, and it deletes obsolete data.

PhEDEx was designed in a more trusting time, security considerations were based on the value of the data. Operators and site administrators were given high-level access to the database, and within the scope of those capabilities they were fully trusted. Whenever someone had difficulty performing a task this was normally resolved by giving them greater access to the database, so they could do what they needed directly in SQL.

This is unsuitable for sustained operations. Even where correct authorisation and accounting are used, granting more access than needed is a bad idea. The risk of a hacked account being used to compromise the database, or of an operator error that could be prevented by proper protection, is higher than necessary. We are more vulnerable than we need be.

To address this, we have re-designed the way security is handled within PhEDEx. This paper describes the situation as it was, and how it is after the re-design. The methods used to design and implement the new security model are presented.



2. What are we protecting?

Before you can decide *how* to protect something, you must decide *what* to protect. The obvious thing for a high energy physics experiment is its raw data, which is expensive and irreplaceable. Changing running conditions (detector configurations, beam energy etc) mean that even in a long-lived experiment like CMS, there are periods for which the data has unique characteristics.

2.1. Raw data

CMS protects its raw data by maintaining two copies on tape at separate sites. The Tier-0, at CERN, maintains one complete copy of the raw data, while the 7 Tier-1s between them maintain a second copy. A disaster such as a fire at one centre may destroy one copy, but not both.

One can argue that the even losing raw data is not a disaster. If the raw data were lost only after it has been reconstructed, physics analysis could still continue using the reconstructed data. Any analysis which required full reprocessing of the raw data would clearly no longer be possible, but most of the physics program of CMS would continue unimpeded. Nonetheless, protection of raw data is seen as the ‘gold standard’ for security within an experiment.

2.2. People, process, reputation

However, there are other things that need protecting too. In the recent past, a member of CMS accidentally typed the wrong command and deleted a large number of files from the EOS storage system at CERN. Fortunately, the damage was minimal, few files were permanently lost. People were able to respond in a light-hearted manner, as illustrated by the ‘wanted’ poster seen on CMS office doors at CERN (figure 1).

There are other things that could have been lost in this incident, apart from the files themselves. One is the reputation of the person who issued the command. If their name had become publicly known, they would have experienced considerable embarrassment.

Another expensive loss could be the time it takes to recover from the incident. The loss of Monte Carlo data, analysis ntuples or ROOT files can cause significant delays to ongoing analyses. Publication deadlines could be missed because of such an incident.

Another question here is to ask who is responsible for this incident? Any modern unix filesystem provides protection mechanisms (file mode or access-control lists) which can prevent a normal user from being able to do any damage with such a command. Most storage element implementations offer similar protection.

So is it reasonable to share the responsibility among other people, such as the people who installed or configured the software to leave the file-access permissions wide open?



Figure 1.

3. Risks, threats and vulnerabilities

3.1. Modelling security

Formalising this idea, we can model the security of a system in terms of its risks, threats, and vulnerabilities. Such a model is shown in figure 2, which is self-explanatory. Assets expose vulnerabilities which can be exploited by threats. These all contribute to increasing risk. Controls protect against threats to reduce the risk, according to the protection requirements you choose.

To make use of such a model you need to know what your assets are, how they are vulnerable and how that vulnerability can be exploited. Then you can formulate the protection

requirements. For example, the PhEDEx database is clearly an asset. The need for connection parameters (username and password) to connect to the database expose it to vulnerabilities, such as the threat of these parameters being left unprotected on disk somewhere. This can lead to requirements to reduce the lifetime of passwords, or to limit the access for any given database user to only the minimum necessary, which in turn leads to the implementation of controls to reduce the risk.

Note that the term ‘threats’ here does not necessarily imply malicious behaviour. Any activity that can cause an asset to be damaged is a threat. The use of a wrong command, as evidenced earlier, is a real threat, regardless of the intent of the person who issues it.

PhEDEx interacts with a number of different components of the CMS computing system [2]. As such, there are many assets that are owned by PhEDEx, managed by PhEDEx, or which interact with PhEDEx in a way that can expose them to risk:

- The data, which PhEDEx is capable of moving or deleting
- The PhEDEx database, an Oracle database maintained by CERN IT division
- The PhEDEx codebase, which consists of three main components; the data-transfer agents, a website [3], and a web-based data-service [4]
- The operators who manage the daily activity of making transfer or deletion requests, approving or denying them, and monitoring and debugging problems [5]
- The HTTP group, who install and maintain the servers which run the PhEDEx website and data-service, among other things. These people have access to the internal configuration files of the data-service, and are responsible for ensuring they are correctly deployed.
- Site managers who install and run PhEDEx agents for their site

This list is far from exhaustive. A complete list would have to include the services that interact with PhEDEx, the hardware the agents run on at the CMS sites, the people with access to those machines (and therefore to the agent code at that site), the administrators of the database in CERN IT division, and more.

Of these assets, only the code and database are directly controlled by PhEDEx. However, the data can clearly be put at risk by misuse of PhEDEx. Similarly, everyone who has to use or install the software is also exposed to risk, as they must ensure that only authorised users have access to the machine the software is installed on.

In the rest of this paper we consider our security primarily from the viewpoint of users of the system. Other standard security techniques, such as keeping software up-to-date, activity monitoring, integrity checks and protection against external attackers are still needed, but these are not discussed here.

3.2. The old security model

In the original security model, the data, the database, the website and the data-service were considered to be the items at risk. The PhEDEx code and the people who interacted with it were not considered to be sources of threat, they were implicitly trusted.

The database was secured using standard Oracle techniques, database roles which give specific access-rights to specific groups of tables. Access was granted to update or insert per table, which is not sufficiently fine-grained. Where several site agents write to the same table it is possible that a bug or mistake with an SQL command could cause one site to update rows that contain data for to another site, or to update columns which they should not alter.

Initially, very few people (only the project developers) had access to the database administrative password, so only they could manipulate the schema. With time and experience running PhEDEx, there were occasional interventions that were implemented with SQL scripts, and these required high-level database access. Operators were given the database administration



Figure 2. Modelling security. It is important to know what your assets are, what vulnerabilities they have and that can lead to damage, in order to know how to protect them.

password, and trusted not to expose or abuse it. The correct solution, providing a data-service API to implement the action performed by the SQL script, often took too long to implement, which is why the easier path of granting full access was taken instead.

The website and data-service, for historical reasons, implemented security separately. This can lead to inconsistencies between the two, which we are addressing by re-implementing the website to use the data-service exclusively [6].

3.3. The new security model

In the new security model, the exposure of people to risk is considered a primary problem, on a par with the risk of loss of data. Problems such as the ‘leakage’ of privilege with time (e.g. as described above) are considered directly. CMS has a long future ahead of it, and we need to be able to prevent the decay of security over time.

We are able to improve security partly due to improvements in the code itself. PhEDEx has become more modular [7], so it is easier to write new data-service APIs. We are implementing a ‘generalised request’ framework [8] which will allow us to turn manual procedures into properly managed requests. These will have authentication, authorisation and accounting to ensure that activities are tracked, that operators need no special privileges, and that the right level of oversight is provided by involving all the people with a stake in a particular action.

4. Protecting the database

We are developing new tools for managing database roles. The previous tools for creating roles for sites or central operators were not easy to maintain. Our new approach is to produce a detailed map of the schema and its correspondance with the desired roles. This is verbose, but is easy to verify and update as the database schema evolves.

We use a two-level mapping of roles. In the schema map, roles are named by their low-level functionality, such as ‘topology manager’, for adding and deleting nodes. Table and column access can be specified at the finest granularity, and grouped into the exact set needed to perform a given function. So a topology manager will have write access to the specific tables and columns the role needs, and no other.

These ‘functional roles’ are grouped into sets and granted to a single database role, which is then given to people who need it. The set of functionalities granted to a role can be altered as required, so it is easy to give the exact functionality required to a specific database role.

We also provide PL/SQL procedures to perform certain key functions. These run with administrative rights, and access to the procedures is granted to database roles via the same schema-map mechanism. The PL/SQL procedures validate their input arguments, which prevents typos in SQL commands from wreaking havoc. Granting access to a procedure is also simpler and easier to maintain than granting access to a set of tables. Because users no longer need direct access to a set of tables, the scope for malicious SQL statements is also reduced.

5. Protecting the website

Being publicly visible, it is important that the web servers be secure. The website is maintained by the CMS HTTP group, who work with the developers to ensure the necessary basic security.

For the PhEDEx website, the additional issues are:

- The website and data-service act on behalf of many users, so have high level access to the database. Someone with access to the servers could modify the code for malicious purposes.
- This means the HTTP group are exposed to risk of suspicion if an incident occurs. This is unreasonable, they have as much right to protection as the data itself.

The solution is to limit the capabilities of the website. The way to do this is to establish data-service APIs for sensitive activities, and to use the generalised request framework to turn them into routine operational tasks, like the transfer and deletion requests we use daily.

This moves the focus of security from the application code into the database. Operators and users will need less privilege, and people who install the software (either the website or site agents) will be less exposed to risk.

6. Conclusions

Modeling our risks, threats, and vulnerabilities allows us to take a more holistic approach to security than we have in the past. Considering the risk posed to users and operators of the system, not just to material assets like the experiments’ data, leads us to a more robust design.

We limit the spread of code that implements sensitive features and limit the spread of access to the database beyond that which is really needed. We use the generalised request mechanism, the data-service and stored PL/SQL procedures. This lets us support a wide range of activities in a safe and secure manner, with proper authorisation and accounting.

No-one could claim that PhEDEx will ever be 100% secure, some residual risk will remain. Data can be deleted or lost by means outside PhEDEx’s control. Malicious or accidental misuse of remaining privileges can cause delays that cost time to recover from. Finally, there are people whose exposure cannot be reduced below some pedestal, such as the database administrators in CERN’s IT division who install and maintain the database itself.

Nonetheless, this represents a significant step forward in securing the assets that PhEDEx interacts with or owns, and gives us the means to ensure that security is not eroded with time.

References

- [1] Egeland R, Metson S and Wildish T 2008 Data transfer infrastructure for CMS data taking, *XII Advanced Computing and Analysis Techniques in Physics Research (Erice, Italy: Proceedings of Science)*
- [2] Giffels M, Guo Y, Kuznetsov V, Magini N and Wildish T 2013 The CMS Data Management System *submitted to CHEP 2013*
- [3] The PhEDEx website, <https://cmsweb.cern.ch/phedex/>
- [4] Egeland, R, Huang, C-H and Wildish, T 2010 PhEDEx Data Service. *J. Phys: Conf. Ser.*, **219** 062010
- [5] Gutsche O *et al* 2013 CMS Computing Operations during run 1 *submitted to CHEP 2013*
- [6] Egeland R, Huang C-H, Rossmann P, Sundarrajan P and Wildish T 2012 The PhEDEx next-gen website *J. Phys.: Conf. Ser.* **396** 032117
- [7] Sanchez-Hernandez A, Egeland R, Huang C-H, Ratnikova N, Magini N and Wildish T 2012 From toolkit to framework: The past and future evolution of PhEDEx *J. Phys.: Conf. Ser.* **396** 032118
- [8] Huang C-H, Magini N, Ratnikova N, Sanchez-Hernandez A, Wildish T and Zhang X 2013 Request for All - A Generalized Request Framework for PhEDEx *submitted to CHEP 2013*