Re-designing the PhEDEx Security Model

C-H.Huang[1], T.Wildish[2], X.Zhang[3]
[1]Fermi National Accelerator Laboratory, [2]Princeton University, [3]Institute of High Energy Physics, Beijing

PhEDEx. the data-placement tool used by the CMS experiment at the LHC, was conceived in a more trusting time. The security model was designed to provide a safe working environment for site agents and operators, but provided little more protection than that. CMS data was not sufficiently protected against accidental loss caused by operator error or software bugs or from loss of data caused by deliberate manipulation of the database. Operations staff were given high levels of access to the database, beyond what should have been 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 the new security model, and we show how the resulting enhancements have improved security on several fronts simultaneously.

What are we protecting?

People are our most important asset. We trust a great many people to do their jobs, often under pressure, and grant them high levels of access to systems. In doing so, we expose them to risk. In the event of an accident or a successful break-in, it's natural to ask who could have been responsible. Because many people more access than they need, they become potential suspects in such circumstances.

Reputation or **credibility** of the experiment is our next most important asset. To protect it, we need to be able to respond rapidly and effectively to security threats.

Data is often considered our most valuable asset, but compared with our credibility, it's not. As we acquire more data, the value of any chunk of it goes down, but a lost reputation is lost forever.

Infrastructure is also important. We rely on many services, with servers of all sorts, batch nodes, databases and networks which are tightly coupled. Loss of access to any of these, even for a short while, can inconvenience a lot of people.

Risks, Threats, and Vulnerabilities

Risk: A possibility that a threat exploits a vulnerability in an asset that causes damage or loss to that asset

For PhEDEx, the risks are:

- Loss of data, either of a single copy or all copies of data. Of course, loss of raw experiment data is much more damaging than loss of monte-carlo, but even loss of monte-carlo can cause inefficiencies.
- Loss of access to data. Even if data is not destroyed, if we lose access to it for some period of time, this can delay analyses and cause deadlines to be missed.
- Loss of ability to move data. Data is often consumed in places other than where it is produced. If we cannot move data around CMS, this is also very bad. This could happen because of network problems, site problems, or problems in the PhEDEx database.

Threats: Something that can potentially cause damage to CMS, or to our ability to do work

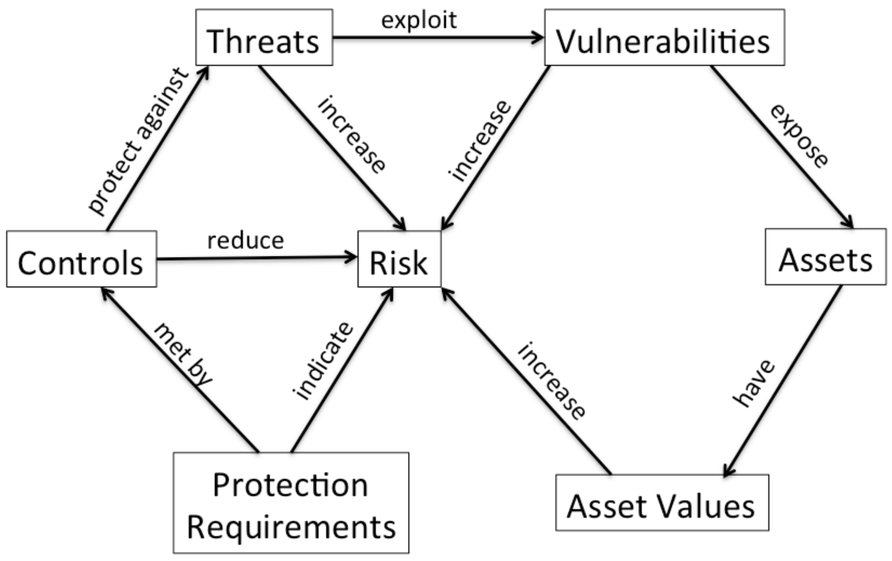
The main threats to PhEDEx can be classified as:

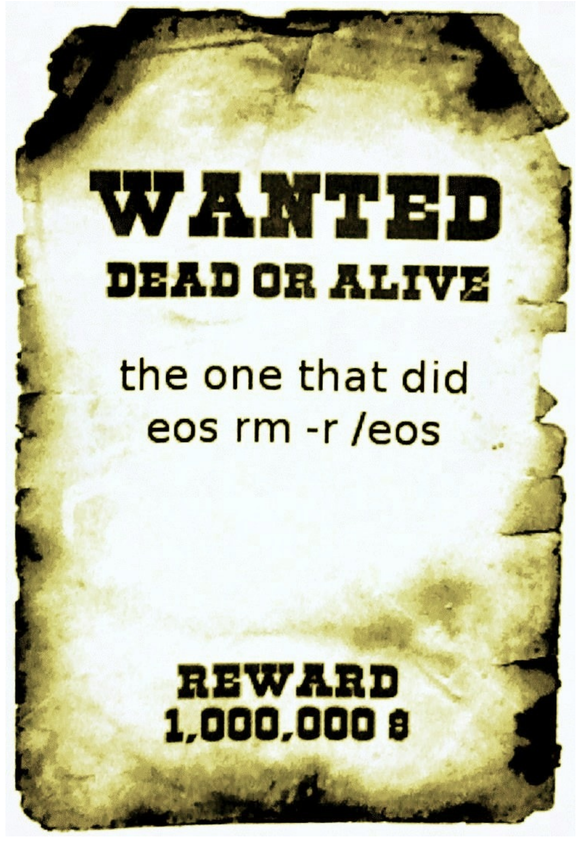
- Anyone with write access to the PhEDEx database. Many CMS members could harm the PhEDEx database or infrastructure if they chose to, by mis-using the access they need for their daily work.
- CMS-internal threats, such as mistakes by inexperienced operators, PhEDEx administrators at CMS sites, or developers.
- CMS-external, but still WLCG-internal. There are many people who have legitimate access to our infrastructure who are not CMS members. These include CERN IT service administrators and WLCG site operations personnel, among others.
- External to WLCG. This includes ex-employees who still retain access to our systems, as well as random hackers and systematic attacks by people or malware.

Vulnerabilities: Weaknesses that can be exploited by threats

PhEDEx, has many vulnerabilities:

- Access-rights to the PhEDEx database are too coarse-grained. People who need only basic access get more access than they really need.
- Shared accounts, where many people know the password. This can lead to leaking of other information that should be kept private, such as an operators' private database connection parameters.
- Shared voboxes or computers, where many people use the same machine, possibly for different purposes. This can also lead to the leaking of information that should be kept private.
- Operational procedures rely on vigilance from a great many people to be carried out safely. Data-transfer or deletion requests need to be scrutinised by different sets of people, depending on the nature of the data. Entry and exit procedures for people joining or leaving CMS are complicated by the extended nature of our computing environment. That same extended nature makes incident-response a challenge.
- The PhEDEx website, which contains security-related code mapping users certificates to their allowed roles. Someone with physical access to the webserver could in principle subvert these checks, allowing users to perform actions they should not be allowed to perform.





"If you think technology can solve your security problems, then you don't understand the problems, and you don't understand the technology"

Bruce Schneier

Protecting the website

The PhEDEx website is a key resource. It performs many actions on behalf of all users, so has higher privileges than any normal user would otherwise have.

The webservers are installed and maintained by the CMS HTTP group, who otherwise have nothing to do with PhEDEx or its daily operations. Vulnerabilities in the website therefore expose these people to greater risk than necessary, since they have access to the installed code.

The best way to protect the website (and the HTTP group), therefore, is to move as much of the security-related activity as we can out of the website code and into the database, where far fewer people have access.

This can be done by providing extra data-service APIs that call database functions that perform actions on behalf of the user, and granting the website access to those functions, instead of to the underlying database tables.

Protecting the database

The database can be better protected in many ways:

- Fine-grained database permissions, allowing write access to only those columns that a user should be able to update. This is done with standard Oracle grants.
- Controlling access to table rows, so that site X does not accidentally (or deliberately!) update rows belonging to site Y. This can be done with database triggers, though the performance overhead has yet to be evaluated.
- Use of PL/SQL procedures to perform sensitive tasks, with access granted to those procedures instead of the underlying tables. This is easy to do, and allows thorough checking of input arguments and logging of actions in a way that is impossible for someone to circumvent unless they have administrative access to the database.
- Fine-grained user-roles, allowing an individual user to have exactly the access-rights they need. This needs a more rigorous mapping tool than we have used in the past, where users were mapped to a single role which had all the rights (and more) than they actually needed. Instead, we need a tool that maps individual tables and their columns to specific CMS-functionality (e.g. adding a new node), and a way to map an arbitrary set of that functionality to a given user.

Residual risk

No distributed computing system can ever be 100% secure, there is always some residual risk that must be accepted. By examining our threats and vulnerabilities, we can reduce risk to acceptable proportions. A systematic approach to deciding what to protect, and what to protect it from, are essential.

By applying these principles we have been able to greatly reduce the risk of damage to CMS or its members arising from the use or misuse of PhEDEx.

References

- Request for All - Generalized Request Framework for PhEDEx (CHEP 2013)
- The CMS Data Management System (CHEP 2013)
- Integration and validation testing for PhEDEx, DBS and DAS with the PhEDEx LifeCycle agent (CHEP 2013)