PROCEEDINGS OF SCIENCE

Importance of User Deprovisioning from Services

Slávek Licehammer[1]
Masaryk University
Žerotínovo náměstí 9, Brno, Czech Republic
E-mail: slavek@ics.muni.cz

Michal Procházka
Masaryk University
Žerotínovo náměstí 9, Brno, Czech Republic
E-mail: michalp@ics.muni.cz

Every service uses an authorization process to determine the access rights of individuals. Lots of services make authorization decisions only during the authentication process and though the process the information about access rights is valid for the whole session. The other common approach is to run the authorization process for single each request from the user.

Both of the these approaches are commonly used and they are sufficient for most services. However there are services that enable users to work with persistent resources. An example of such services are cloud infrastructures which enable users to start virtual machines or use data storages for storing large amounts of data. Apart from the aforementioned authorization done whilst user is interacting with the service, there is a need to know that the user is still authorized to use the resources, even though the user is not interacting with the service. Such knowledge enables services to free the persistent resources which were occupied by the user who is no longer authorized.

Deprovisioning is the process which enables service to know about users who are no longer authorized. It is the opposite of the well-known provisioning process, which is used in cases where the services need to know the users in advance of their first usage of the service.

In this paper we describe the importance of the deprovisioning process based on real use-cases and services. Moreover we will focus on possible options to implement deprovisioning in existing infrastructures. Last but not least, we will describe similarities between a standard deprovisioning process and the suspension of users on services due to security incidents. Based on those similarities, we will demonstrate on a real system how to utilize the deprovisioning process to automate mitigation of security incidents.

International Symposium on Grids and Clouds 2016
13-18 March 2016
Academia Sinica, Taipei, Taiwan

[1]Speaker

PoS(ISGC 2016)016

1. Introduction to Authentication and Authorization

All e-infrastructures have to deal with authentication and authorization issues. Current trend are leaning towards providing authentication and authorization to the service externally. It can be achieved either by deploying some central solution or by utilizing trust model like X.509 [1] certificates provided by chains of trustworthy authorities or identity federations like SAML2 [2] based eduGAIN [2] inter-federation.

These solutions usually make authorization decision together with authentication at the moment when the user accessing the service. Authorization is typically based on additional user information in form of attributes which are obtained either from external attribute authority or they are passed as a part of authentication request. SAML assertions in SAML request are very good example of this, another example is attributes obtained from X.509 certificate. The process for creating a user account in a system based on the information obtained is called provisioning. Alternative to this just-in-time provisioning, where attributes are obtained during user login to the service, is active provisioning from an external authoritative source (e.g. an attribute authority). It can be achieved either using a push model or a pull model. The push model require an external authoritative source that monitors changes within itself and actively propagates these changes to relevant services. On the contrary, in a pull model each service needs to query the external authoritative source for changes.

Just-in-time provisioning is much easier to implement because it can be fully handled as part of authorization on the service side. For that reason it is commonly used on most services today. Moreover, it enables you to use data from multiple attribute authorities (e.g. SAML2 based authentication) without any major obstacles. Active provisioning either requires that only a single attribute authority is used, or that a well-defined process exists to deal with duplicate user records in multiple attribute authorities.

The aforementioned considerations notwithstanding, the active provisioning has several advantages over just-in-time provisioning. It enables the service to know about the user in advance, before his/her first login to the service, which can be used to enhance user experience for some services. For example data sharing service enables you to easily find your colleagues to whom you want to share the data in internal user list. Moreover, active provisioning enables the service to update data about users even though they are not currently using the service. For example to keep the user's e-mail address stored within the service up-to-date. Last but not least, the service can discover that some user is no longer authorized to use the service and react appropriately – this process is called “deprovisioning” as an opposite to the provisioning process.

[2] http://edugain.org

2. Deprovisioning

Deprovisioning is fundamental for services where user can consume or reserve resources for himself. In other words, all services that have permanent data of any kind related to the user should implement a deprovisioning process in some way. These services have to know that the user lost his/her rights to use the resources and, based on that fact, the service can take appropriate action to free this resources and by that enables other legitimate user to utilize them.

Another reason for deprovisioning is security hardening. Without deprovisioning it is possible for a user to start some long term job (for example in a computational grid environment) or start a virtual machine which will remain active even when user access rights expires. This could be evaluated as a security risk. In the case of virtual machines or other similarly complex services, it is possible for the user to open a back-channel connection directly to the resource that uses a different type of authentication (e.g. SSH keys). This means the user can fully utilize the resources even if he/she is no longer authorized to.

Deprovisioning can also be successfully used as a tool for security incident mitigation. When an incident is discovered, the security team should disable accounts of affected users on all services, therefore the attacker will not be able to escalate stolen privileges even further. This principle is the same as the principle of deprovisioning. In a matter of fact, the deprovisioning process can be used to address this use-case. The process stays the same, only the services will get notification about suspended user in addition to expired or deleted, which it would get in a standard provisioning process. The push model is more suitable for security related deprovisioning because you can have assurance that changes will be delivered as soon as possible. The use of a pull model causes delay between information about suspended users being published and the time that services will pull the data and apply them internally.

Although a deprovisioning process is necessary, current services often do not implement it directly, but they rely on some administrative process as a substitution for it. For pay-per-use services it is very simple because the user is charged for the use until the user decides to stops using the service. A more difficult situation is related to services which are not paid for directly, and where the access to the service is provided based on other conditions like the affiliation of the user. Such services are typical for academic environment. This services are in practice often dependent on manual or partially automated processes which will be executed when a specific condition is met and then should find the users that should be deprovisioned. A trigger for such process can be running out of space on data storage or not having enough resources to run new virtual machines on cloud infrastructure. Being dependent on a concrete use-case, these conditions may never happen in practice, so it is common that deprovisioning is not solved at all for some services.

An other reason not to implement deprovisioning may be hidden in requirements for underlying authentication and authorization infrastructure. Specifically, a deprovisioning process depends on a well-defined user life-cycle. To be able to inform the services that a user is authorized to use them, you have to have this information available in the identity management

system. A defined user life-cycle enables it to clearly communicate to the services the status of the user together with a list of resources which are available for the user on each service. A well-defined and documented user life-cycle helps services administrators to understand how the users are managed, and thereby the trustworthiness of the system as a whole is increased.

3. User life-cycle

The user life-cycle starts by creating the user in the identity management system. It can be done either by self-registration of the user or by importing him directly from other source of identities. Then, the user is assigned roles or placed in some groups. Based on the group or roles the user obtains access rights to use some resources on the services. That should trigger the provisioning process which will propagate required data to services. Sussequently, the user related data can be changed. User can obtain new roles or become also a new member of a different group or some of his/her attributes can be changed. The provisioning process ensures that all the changes will be communicated to services. The next step in the user life-cycle is one of expiration, of renewal of expired account or of deletion of a non-renewed account. All such changes need to be provisioned to the services so the service can react in accordance with its policies. The aim of provisioning and deprovisioning is to notify the service about changes in the account used, not to define how the service should react.

To support these features in an existing infrastructure, the user life-cycle must be implemented by the infrastructure. Alternatively, additional component can be incorporated into the infrastructure to address aforementioned requirements that were not hitherto addressed. We want to introduce Perun [3] system as universal tool for identity and user management. It has been designed to cover the management of the whole ecosystem around the users' identities, groups, resources and services. It has been also designed to be easily incorporated into existing infrastructure and workflows, without any modification on existing systems interfaces, supposing the existing systems are able to export relevant data. Perun can be used as a central component that merges identities, user attributes or groups from several existing systems into one place and by that provide a uniform interface as a consolidated way to work with them.

4. Perun system

Perun provides several options to deal with provisioning and deprovisioning processes. User data can be provided using the API of the Perun or alternatively using a SAML2 attribute authority or LDAP [4] interface (fig. 1.), for which Perun have built-in support. All these methods rely on active querying by the end-services, which may not be suitable especially for larger infrastructures, hence it probably needs to spend effort on service level to have these endpoints up-and-running all the time. Furthermore, at larger scales it might bring performance issues, because all services will periodically query the endpoint even when there is no change in data, because the service does not know that in advance.

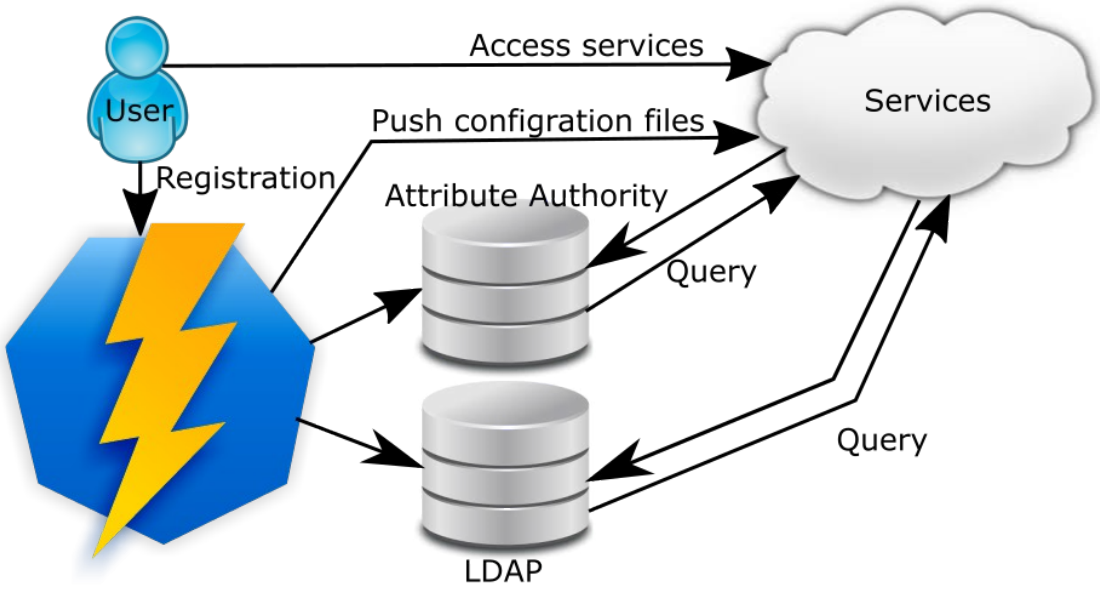

Fig. 1. Provisioning options using Perun

As a solution to this issue, Perun provides possibility of active provisioning to services only on data change. Perun is able to monitor data flows within itself and to decide which data affects which service. Based on that the fact, new configuration for the service is prepared in a format suitable for the particular service and then the configuration is sent to the service using a standardized channel like ssh (fig 2.). Alternatively it is possible to write a simple connector to send the data using a proprietary protocol.

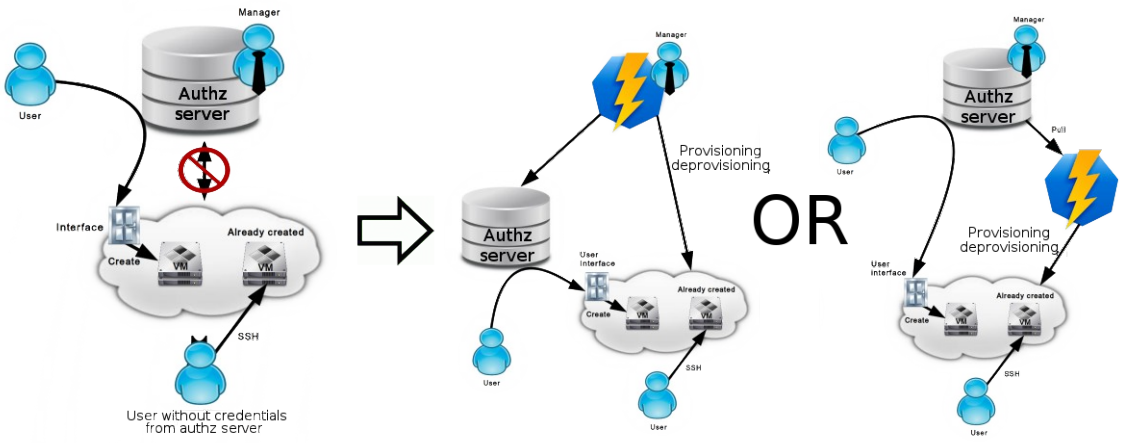

Fig. 2. Implementing provisioning and deprovisioning using Perun

The advantage of such solution is that there are virtually no requirements on the service level. Each service has a way to get the configuration from outside, therefore writing but a simple connector is usually sufficient to manage authorization rights for the service from the outside. Moreover, this way of provisioning scales very well, because data are generated and sent only when there is really some change within them. Data are stored directly into service configuration file or internal database, that means the service can run and authorize users

independently of the central identity management component. Thanks to that, we do not have to worry about outage of the identity management component: in that case all services will still be fully operational, they merely will not get updates until the outage is over.

5. Summary

Deprovisioning is necessity for the services that store permanent data or enable users to run permanent or semi-permanent jobs. Current services often do not deal with deprovisioning systematically, but rather depends on an administrative solution to this issue. To implement deprovisioning within the infrastructure you have to have a well-defined user life-cycle. Alternatively one may add a new system to the infrastructure to help implement user life-cycle adn based on that build a provisioning and deprovisioning processes. The Perun system was presented as an example of such system.

References

[1] ITU-T Recommendation X.509 – Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks, 2005, http://www.itu.int/rec/T-REC-X.509/e.

[2] S. Cantor, J. Kemp, R. Philpott, E. Maler, et al. – *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS, 2005, http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf*.

[3] M. Procházka, S. Licehammer, L. Matyska, Perun – *Modern Approach for User and Service Management*, Mauritius: IIMC International Information Management Corporation Ltd, 2014. 11 s. ISBN 978-1-905824-44-1.

[4] Zeilenga, Kurt. *Lightweight directory access protocol (ldap): Technical specification road map*.,2006, https://tools.ietf.org/html/rfc4510.