Multi-mechanism Single Sign-On in Grids (CESNET Technical Report)

Daniel Kouřil, Luděk Matyska, and Michal Procházka

CESNET z. s. p. o., Zikova 4, 160 00 Praha 6,
Masaryk University, Botanická 68a, 602 00 Brno
Czech Republic
{kouril,ludek,michalp}@ics.muni.cz

Abstract. Recently, a lot of effort has been invested in development of various types of authentication mechanisms that offer strong security characteristics for building grid systems. Being based on one of the mechanisms, most grid environments today provide strong authentication protocols, however, they are usually bound with only one, in most cases based on public key infrastructure (PKI). Such an arrangement works pretty well, but unnecessarily limits users since they are required to use only the one mechanism, which may not be flexible or convenient. A better solution would be to offer users a freedom to choose their own authentication mechanism—provided it is strong enough—and provide automatic translations that guarantee that all services and components are securely available regardless of the choice of authentication mechanism. This technical report surveys several authentication mechanisms that are used in contemporary grid and other distributed systems and also discusses possible transitions to translate credentials. The report summarizes recent results in this area, many of which have been achieved by the authors.
This technical report is an extended version of the paper we presented at the CESNET08 conference [**?**]. It also contains information how the described tools have been deployed in the environment of Czech national grid infrastructure METACentrum.

1 Public Key Infrastructures

Asymmetric cryptography allows for efficient key management, especially in loosely-coupled distributed environment. The concept of two separated keys has been proven to provide a scalable solution to parties that need to mutually communicate and do not share any pre-distributed secret. As the simplest implementation of a asymmetric cryptography, a plain key-pair can be used to establish an authentication connection. This approach is employed by the popular `ssh` protocol, where such a simple public-key based authentication is mandatory to implement [1] and widely used. While it is convenient for users, experience shows that having authentication based only on plain keys is not sufficient and is hard to maintain in a large environment. The main drawbacks are a lack of information who a particular public key is bound to and a lack of centralized mechanism that could be used to revoke a public key if the corresponding private key becomes untrusted. These drawbacks have been fully revealed recently, when a vulnerability in some version of the `openssl` toolkit was announced [2], which could cause that weak cryptographic material was generated. When handling the vulnerability by the grid community, an audit has been performed to check all Grid certificates. If a weak key for a certificate has been found, the certificate has been revoked and the certificate owner asked to re-generate their certificate. This allowed to resolve the issue quickly and centrally invalidate all vulnerable certificates. No such precaution can be implemented for `ssh` keys deployed on many machines due to the lack of centralized key management. Moreover, since the lifetime of the keys is not limited, weak keys can still survive on the systems, until the system administrator or the user detect and remove them from each and every system.

Based on these experiences, it is obvious that an additional infrastructure to manage key-pairs is necessary in every large-scale distributed environment. There are several public key infrastructures (PKIs) available today. The most widely used PKI, which is based on X.509 certificates (PKIX) is standardized by ISO/ITU-T specifications [3] and was also profiled for Internet use by IETF [4].

The standards define the format of a public-key certificate, and also other components (e.g., certification and registration authorities) and processes (e.g., certificate revocations) needed to provide the whole infrastructure. X.509 certificates have became very popular over past years and are used in widely used security protocols, such as TLS.

To overcome some issues of PKIX and to cover other areas of computer security, the Simple public key infrastructure (SPKI) has been specified [5, 6]. SPKI concerns rather with authorization issues and binds a public key with a set of authorization attributes with the keyholder name being just one of them.

On the other side of the public key management spectrum, there is Pretty Good Privacy (PGP) [6]. PGP is based on a completely different schema as it does not use the service of centralized authorities. In PGP users build their own *web of trust* in which they act as an independent identity vetting authority. PGP users sign certificates assessing identity of other users and share these certificates globally. The principle of the web of trust resembles reputation systems that are emerging currently.

1.1 PKI in Grids

Even though both SPKI and PGP offer interesting features, to our best knowledge they are not used in any current Grid environment. Contemporary Grids that are based on PKI either use the standard X.509 PKI or they use a special form of digital certificates derived from X.509.

The former approach is employed by the UNICORE grid middleware [7], where jobs are signed by the owner's private key before submission. In the basic version, the UNICORE infrastructure than uses the X.509 certificate to verify the signature during the job processing.

Other grid systems based on PKI use certificates in a different way. Instead of creating long-lived digital signatures, they provide users with mechanism of *single sign-on* (SSO) and credential delegation, which allows users to submit a job to or request an operation at the infrastructure along with appropriate credentials. The credentials are used by the job and middleware component to perform any operations that are necessary to complete the job on the user's behalf. *Proxy certificates* [8] are very often used as the mechanism providing SSO and delegation. A proxy certificate is a special kind of X.509 certificate that is signed by an ordinary user, not by a dedicated CA. The generation of a proxy certificate is performed during the login stage by the user. Lifetime of proxy certificates is very short, usually ten hours. Proxy certificates provide an efficient mechanism for SSO, however they also present new issues. One of the most severe one is a lack of revocation mechanism that could be used to revoke existing proxy certificate. Another issue concerns support of long-running jobs and other operations that last longer than is the proxy lifetime. Despite a renewal mechanism has been designed and implemented [9], it is not an ideal solution since it requires introduction of an additional trusted component.

Proxy certificates are often managed using *MyProxy* [10, 11], which provides a repository where proxy certificates stored and later retrieved using a password or another credential assigned during the storing step. In addition to support of the repository mode, a MyProxy server can also be configured to act as an CA issuing standard X.509 certificates.

Regardless particular type of certificates or PKI, one of the key drawbacks of the PKI is that current tools and producers for certificate management are too complicated for users. This leads either to rejection of the PKI or to insecure private-key management, which dis-empowers all the Grid infrastructure.

2 Kerberos

Kerberos [12, 13] is an authentication protocol using a trusted central authentication service — *Key Distribution Center* (KDC). Each user and service shares a secret key with KDC. KDC issues *tickets* asserting identity of their bearers, which serve similar purpose as public-key certificates. A ticket has a limited lifetime but unlike public-key certificates, it can be only used against a particular end-service, which is specified in the ticket. To support the SSO principle, Kerberos

uses a universal *Ticket Granting Ticket* (TGT) that is used to authenticate against a KDC to retrieve tickets for end-services. Apart from supporting mutual authentication of the peers, the Kerberos protocol also provides means for message encryption or integrity protection.

Since a KDC holds a list of all users and services, every authentication among users and services involves contacting KDC. This feature makes it hard to deploy Kerberos in highly distributed environments since users must be registered with KDC first. In order to make Kerberos more scalable, the users' space can be divided into administrative groups (*realms*), served by independent KDCs. The Kerberos cross-realm authentication mechanism allows seamless interoperability among different realms. The current mechanism of establishing a cross-realm trusted relationship is one the most serious obstacles that prevent from using Kerberos-based infrastructure in Grid systems. In order to create a such a relationship, the KDC administrators must meet to exchange cross-realm keys, which does not scale in large infrastructure. However, Kerberos has become a de-facto standard for local security infrastructures operated by many institutions. There are several Kerberos implementations available today, both open-source and commercial.

The TGT has the same problem of revocation as proxy-certificates—its lifetime is usually limited to one day at most, to lower risk of TGT theft or compromise.

3 SAML and Identity Federations

Federating institutions has become very popular recently, because it allows users to co-operate in a secure manner. The concept of federation can be applied to environment operating both Kerberos and PKI, but in this section we primarily focus on system based on the Shibboleth infrastructure [14].

An identity federation is an infrastructure connecting user management systems from different institutions to provide standardized access to information about users maintained by their systems. Federations provide a standardized platform to which systems for user management and end applications can connect and share authentication and authorization data. Every organization participating in a federation manages its users by a local user management system. An *Identity Provider* (IdP) service is built on the top of each local user management system, providing a standardized interface to access authentication information and other attributes about the users. Any party in the federation can get this information by calling the IdP service using a standardized protocol. End services (*Service Providers*—SP) are able to process the data returned by the user's home IdP and use them to make access control decisions. Before users are allowed to use a service, they have to present a set of *attributes* issued and signed by their home IdP. These attributes are provided to users or to a service working on their behalf upon proper authentication of the user with the IdP.

The major advantage of using the federation model lies in the fact that users use their home institution's credentials to access any service in a federation. Whenever users access a service (SP) and do not have an authenticated session activated, they are redirected to their home IdP to authenticate. After successful authentication they are sent back to the SP along with additional information about their identity added by the IdP. The SP accepts this authentication assessment since it trusts the IdP, and applies appropriate access control methods. Every SP in the federation uses this mechanism transparently from the user's point of view. There is no need to introduce new credentials for every new service or to synchronize existing credentials (like passwords) among different services. Having no additional credentials also means there is no need to distribute them among the users. Such an arrangement not only eases credential management but also makes it more secure, as users are only required to maintain one piece of authentication information and always authenticate at the same (web) interface. Unlike PKI, the federation model is more acceptable to the users, as it is not tied to any particular authentication method and institutions can select the most appropriate method for their users.

Upon proper user authentication, the IdP provides a set of attributes that represent information about the user, its affiliations and roles. They usually include user's name, home institution, roles and position within the institution, group membership, etc. The attributes are very often

encoded using the Security Assertion Markup Language (SAML) [15] in the form of standardized SAML assertions. In this way the home institution provides information that can be used, e.g., for specification of a group of users without explicitly naming them.

Enhanced privacy is a potential side effect of the above-mentioned use of attributes. The IdP could provide only attributes, not a precise user identity (i.e., it could provide the information that a person is enrolled on course identified as CS102 without revealing his or her name or other unique identification). Such attributes can be sufficient for a service to make an authorization decision, however the precision of an audit trace is lost (while the privacy of the user is enhanced). In the event of abuse, the home institution is still able to identify the user from their local logs. This approach is interesting if we do not want to reveal the individual user's identity to a service administrator.

Today's middleware that implements the federation model cannot be directly used in grid environment because it target only at the web environment. Several projects work on integrating federations into the grid and vice versa. Two most known projects are GridShib [16] and ShibGrid [17]. GridShib allows for interoperability between the Globus Toolkit[1] grid middleware software and Shibboleth federation middleware. Globus security is based on proxy certificates used authentication. GridShib consists of several modules that allow users to access the grid by various ways. In general GridShib can work in two modes, pull and push mode. In push mode user, after the successful authentication, obtains proxy certificate from an on-line CA which is part of the federation, therefor response from user's IdP can be included in the proxy certificate as an extension. Afterwards user can access Globus services which can extract the information from the proxy certificate and make authorization decision. In pull mode the Globus service can directly contact IdP and obtain user's attributes. ShibGrid has a similar goal, it integrates British National Grid Service (NGS) into the British academic federation based on Shibboleth middleware, but it does it in different way. NGS uses proxy certificates for authentication as well as Globus. ShibGrid also introduces two modes of operation, one is for users with the UK eScience certificate and second one is for users without it. If user does not have UK eScience certificate she accesses NGS portal as usual SP in the federation. Proxy certificate generation is made behind the portal on the MyProxy server which has built in MyProxy-CA then portal can use user's proxy certificate to access the resources. In second mode when user has the correct certificate it only access special web server which is SP. After successful authentication user obtains from the web server encrypted attributes issued by her IdP. These attributes user's client application puts into the newly generated proxy certificate and upload it to the MyProxy server.

4 One Time Passwords

Passwords are very often used to user's authentication in computer systems. Regardless the popularity of passwords, they can hardly be used for authentication to Grid services since they do not provide SSO. However, passwords are still usually used in Grids to initial login to the infrastructure, i.e., to access private keys in files or MyProxy repositories, obtain a Kerberos tickets, etc.

One interesting implementation of passwords is *one-time passwords* (OTPs) that can be used even from untrusted location since their potential sniffing does not cause any harm. As suggested by the name, an OTP can be used only once and is not accepted for authentication after using. From the user point of view an OTP works in the same way as usual passwords do, except to be always different. If an OTP-based system is to be to be deployed within an infrastructure which already uses passwords, changes required to be applied are minimal. Only the components to validate the passwords have to be substituted, which is usually an acceptable change for deployment. The other change, which is more difficult, is to provide users with tools to manage their OTP and train them in proper usage of the tools.

OTP requires users to manage their lists of generated passwords which are sequentially used for authentication, fortunately there exist hardware devices or applications (so called tokens or soft-

[1] http://www.globus.org

tokens) which facilitate managements the OTPs for the user. OTPs can be managed in several ways by these tokens differing in generating, storing and usage of the password. Application tokens are usually meant for mobile devices such as PDA or mobile phones. Both types of the tokens provide two-factor authentication, where user has to prove that they owns something (i.e., the token) and also knows something (PIN or password for the token). The following chapters describe methods available for managing OTPs.

4.1 Pregenerated sequence of passwords

In this model OTPs are generated before their first use, so the user gets a list of passwords. Passwords are usually printed on paper or maintained by a soft-token that generates the requested password on demand. A user needs to know which password from the list has to be used for authentication to the server, therefor a server has to send the index of the required password at the begining of the authentication process. This system is pretty easy but the user has to updated the list of passwords whenever they have used last one from the list. This principle is used by two systems known as S\Key [18] and OPIE [19], both of them are based on the Lamport's schema [20]. The schema uses an one-way function which eases implementation of password maintance on the server side without requiring the server to store the whole list of OTPs for each user. In this system the user chooses initial password (pass-phrase) and the server generates corresponding random string called seed. User initial password and seed are passed to the hash function which results in a password zero. The hash function is then applied N-times on the password zero, which leads to a list of N one-time passwords. The passwords are used from the end of the list, precisely from the latest generated password, therefor server needs to store only last used password along with the corresponding index from the list. Server can do easily validation of the provided password by applying the hash function on the last used password. If the password is correct then server replaces the stored password with the provided one and decrements the index.

There are also methods like OTPW [21] or HOTP [22] which are not based on Lamport's scheme. They are very promising alternative to methods mentioned above.

Passwords generated by the one way function are not comfortable for use because they are long random numbers. Therefor both specifications OPIE and S\Key define translation dictionaries which encode the long number into six English words (e.g. JIM NOB EARN WIFE RAIN HATH).

Authentication process is based on challenge-response principle, server sends the seed and index of the password to the user. User replies with corresponding OTP password. If the application protocol does not support more than one interation for authentication handshake like HTTP Basic Authentication, where user name and password are send together in one message, it is possible that user provides directly last password from her list and marks it as used.

4.2 Challenge-Response password generating

OTP passwords based on this mechanism are generated by user's PIN code and authentication server's challenge. Typical representative of this category is the CryptoCard RB-1 token[2]. The user enters their PIN code to the payment card-sized token using an embedded keyboard, resulting in an OTP being then displayed on the small display. The OTP is generated by the MD5 hash function, based on the challenge entered by the user and secret key stored on the token as a parameters. The secret key is shared with the authentication server. The token is programmable and various levels of security can be set (length of the PIN code, number of invalid PIN enters) therefor the token can be used indefinitely. As is common for smart cards, the token is disabled if an invalid PIN code is entered several times subsequently.

4.3 Time-based password generating

This type of OTP uses actual time, so it works without any interaction with the authentication server, but on the other side the time has to be precisely synchronized between the authentication

[2] `http://www.cryptocard.com`

server and the token, this could poses a problem. Another disadvantage of this approach is vulnerability against replay attack, because the password is valid for defined period of time (e.g. one minute). This type of OTP is used in RSA SecurID[3] which has only display where every minute is genereated new password. User uses displayed password together with her personal PIN, without that PIN the password from the token is unusable, it is protection against stealing or losing the token. The token has to exchange shared secret, which cannot be then changed on the token, with authentication server during the initialization proces. SecurID tokens have limited lifetime from one to five year depends on the type, after that period new one has to be used. The limited lifetime causes higher cost of the whole system in compare with other types of tokens, however SecurID tokens are to most spread tokens in the world.

4.4 Open source soft-tokens

These tokens are applications which can be uploaded into common mobile devices such as mobile phones which have user by itself for most of the time. These devices are not pernamently connected with the computer, therefor posibility of compromising through the computer is very low. Most of todays mobile phones and PDAs are capable to run that aplications because the application is written in Java Platform Micro Edition (J2ME). Aplications are distributed in standard format with jad or jar extension, that is why they can be uploaded into the device in a standard way. Serial or USB cable, IrDA or Bluetooth can be used for upload the aplication into the mobile device or the aplication can be downloaded directly from the web server over WAP procotol or some other, which is supported by the device. In this section we will introduce some of open source soft-tokens which implement different OTP generating methods.

- *otpgen*[4] It is calculator for S\Key and OPIE methods, it supports MD5, MD4 and SHA1 hash functions. The seed, index of OTP and type of hash function has to be set in the configuration. User do not need to setup these values repeatedly because they are stored in the memory of the device. After usege of password the application decreases OTP index in order to be ready for next generating. The generated password is displayed in human readable word form as well as in hexadecimal format.
- *j2me-otp*[5] This calculator is port of OTP applet to the J2ME. It implements S\Key algorithm with MD5 hash function. In compare to previous calculator it is harder to use, because user has to enter the seed and index of OTP everytime. The input field for seed has caption Challenge which is misleading, also during generating process there is missing information about progress. The input function does not clean up previous results, the new one is added next to the previous therefor it is very hard to distinguish each other.
- *Mobile-OTP*[6] It is used for generating OTP based on the actual time, PIN and shared secret. Last two values have to be shared with the authetication server. The reason why these two values are present is to be able to authenticate the device itself (shared secret) and user (PIN). After the installation the midlet has to be inicialized first that means generate shared secret and transfer it securely to the server.
- *FreeAuth*[7] The project is descendant of Mobile-OTP adding support for time based OTP. Used algorithm does not require user's PIN to be stored on the server. However the midlet is very unstable and is not feasible for production environment.

4.5 OTP support in applications

OTP is widely supported, e.g. by open-souce libraries for S\Key and OPIE mechanisms. S\Key was integrated into the OpenSSH program, which is able to support authentication by one time

[3] `http://rsasecurity.com`
[4] `http://marcin.studio4plus.com/en/otpgen/`
[5] `http://tanso.net/j2me-otp/`
[6] `http://motp.sourceforge.net/`
[7] `http://www.freeauth.org/`

passwords. Variety of PAM modules supports different OTP mechanisms, therefor every application supporting PAM can use OTP. Applications which supports SASL protocol like various SMTP server can also integrate OTP as an authentication mechanism. In WWW environment special modules for Apache web server can be used or PAM modules or special external application installed on the server.

As we can see OTP can be deployed in various applications, nevertheless it is hard to deploy OTP into the large or distributed environment. Most of mentioned types of OTP use local database, which is stored next to the application server, therefor managing this database is not effective and it is problematic to use that data by other services from different places or networks. This is not true for commercial solutions with the hardware tokens, but they are closed source and do not fulfil all requirements.

5 Credential Transitions

Virtually all current grid systems support only a single authentication mechanism and do not provide any coordinated way how a user could smoothly change their credential types. Ideally such transitions would be also supported by the middleware that could obtain a credential type according to the needs of the end service or another middleware component. Such a capability would ease integration of a Grid with other systems that require different authentication mechanisms. In this section we provide an overview of transition mechanisms that are available from current security components. We have tested all the transition described and also contributed to development of several components and mechanisms involved. A schema of transitions can be seen in Fig. 1. The main use-case we have in mind is to ease users' work and allow them to use a broader scale of authentication methods instead of dictating a particular one. According to our experience, if an infrastructure is sufficiently easy to use, it is also safer since users do not pass over barriers using insecure techniques (such as uncontrolled copying of private keys, etc.).

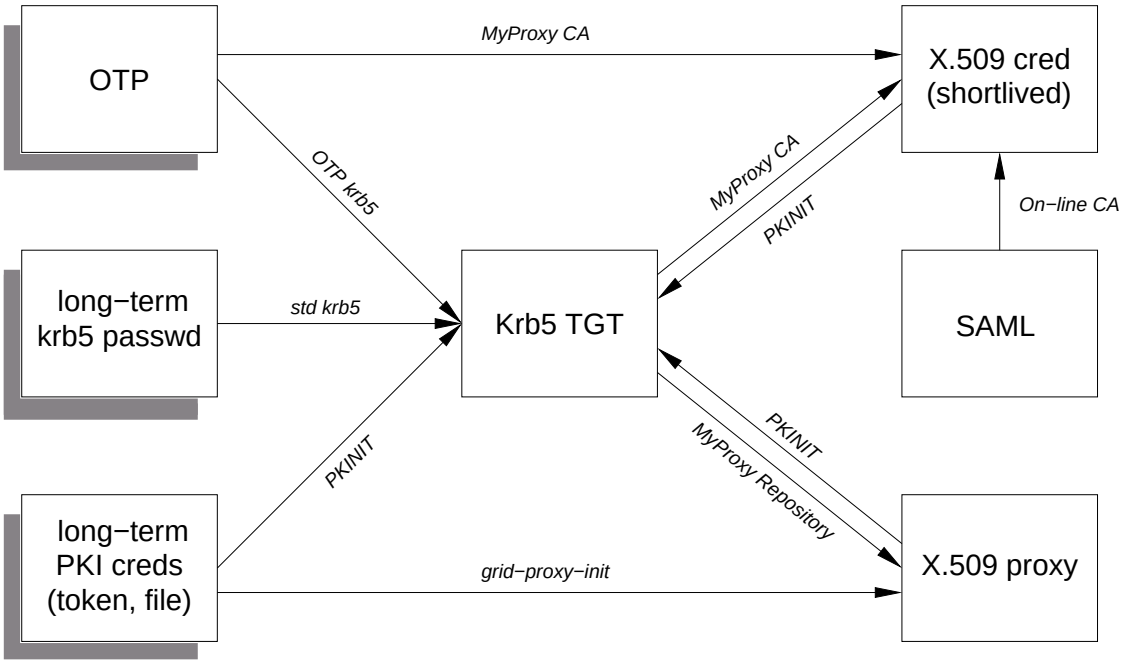

Fig. 1. Schema of transitions

5.1 Converting OTP to X.509 certificate

A MyProxy server operating in both the CA or repository modes uses password based authentication of the clients. Using the pluggable PAM mechanism the server can be configured to support additional verification methods. In our testing environment we successfully configured a MyProxy server using two PAM modules that implement OPIE and MOTP mechanisms which represents two different OTP generating methods. For both these mechanisms there also exist open-source Java client applications that can be loaded to Java-enabled mobile devices. Using these applications users can have secure access to their certificates (either direct or rather via a portal) even from machines that cannot be trusted for input of long-term passwords or other type of credentials.

PAM module for OPIE mechanism is present as a package in common Linux destributions or with other tools for supporting OTP. Either these tools and PAM module works with local database of the users. The database contains all necessary information needed for authentication such as last used OTP password and its index and corresponding seed. The database has to be initialized and list of OTP passwords has to be generated for every user. The PAM module during authentication process checks whether provided password is correct and if so it updates database with new password and index.

For MOTP there is available *mod_mobile_otp* PAM module from the project website in form of source code. Unfortunately provided Makefile needs to be changes in order to be able to produce dynamic libraries. But after that change everything works well. MOTP PAM module also requires database with information about all users.

Mentioned PAM modules can be used together or with other PAM modules for authentication (e.g. Kerberos) which expands MyProxy flexibility of supported authentication mechanisms. There is no need to make any changes on the client side of MyProxy, it can be used as usual. If the user with OTP wants to download the proxy certificate from the MyProxy server or signs the certificate request she uses corresponding command providing password from her OTP generator. This password is then passed to the MyProxy server where is validated by one of the available PAM modules, if validation was successfull then MyProxy will do requested operation. Problem could be with OPIE PAM module, because MyProxy does not have any way to send the seed and index to the the client, therefor the client has to remember last used password and seed. User can use e.g. optgen which manages this information by itself. SASL, general framework for authentication, could be also used because MyProxy supports it.

If the MyProxy works in repository mode, user has to upload the certificate first. During upload process user must identify itself with the username stored in OTP database and also she can set the password which will protect user's data, this option has to be switched off while using the OTP in order to enable PAM functionality. After the user is successfuly authenticated, MyProxy does authorization control whether provided user name correspondes with the subject name of the proxy certificate.

In on-line CA mode, MyProxy has to have in configuration file specified mapping from the user name (stored in OTP database) and newly created subject name of the certificate. MyProxy allows to use localy stored file with mapping from user name to subject name as well as LDAP server.

Mentioned approach, where user can retrieve his proxy certificate or get new certificate from MyProxy using OTP password, implements SSO mechanism for grid environment. MyProxy can be also used as an central authentication server similar to KDC in Kerberos protocol. End services can validete user's passwords against the MyProxy server. Side effect of this approach is obtaing the certificate which can be used by the end service. By this way we can implement OTP gateways for SSO access to the grid environment.

Similar gateway can be implemented for SSH servers, where SSH server lets validation of the password on the MyProxy server. After successful athentication MyProxy returns user's certificate back to the SSH server, where can be used by the user for authentication to the next services. This functionality would require changes in SSH code, but PAM module developed at METACentrum can be used. This module implements communication with the MyProxy server.

Main disadvantage of above approach is lack of administration interface on MyProxy server for the users where they can initialize their OPIE passwords or change PINs for MOTP. Also administrators needs mechanism for imaintaining users entries in OTP database. Nowadays we are discussing two solutions of these administrative problems. First is integration of OTP administration with local identity management system which is generally used on bigger institutions. This will require interface for managing OTP database which can be then used by the identity system. Second solution is to enhance MyProxy protocol with missing administration functionality.

5.2 Converting SAML to standard X.509 certificate

In order for Grids and identity federations to be interoperable it is important that people authenticated through an identity federation can easily access Grid resources. In terms of authentication and single single-on it means that the users must be able to identify themselves using credentials that are acceptable by the grid infrastructure. Since most current Grids uses PKI, some mechanism allowing users to obtain a certificate upon authentication using a SAML assertion issued by an Identity Provider. Such a service can be provided by a special kind of a certification authority integrated into the federation, such as the on-line CA provided by the GridShib project.

The GridShib CA operates as an standard service provider in a federation. Users authenticate against it using standard federation mechanism, with SAML assertions presented to the CA. These attributes are used by the CA not only to decide whether a certificate can be issued, but also to generate a set of X.509 extensions that are embedded into the certificate. The extensions can be later used by services that are able to understand the SAML assertions for additional access control decisions.

The GridShib CA operates entirely in a web world leaving all credential and key operations on the web browser, which is quite convenient for the users. In the standard installation a Java applet is used to perform the cryptographic operations. We also modified the code base to use also the features that the Firefox and Internet explorer browsers provide for cryptographic operations. These modifications simplify the use because no applets are needed any more. We have also developed a GUI for MS Windows systems that simplifies obtaining and management of such certificates [23].

5.3 Converting X.509 to Kerberos TGT

A lot of sites use Kerberos as their primary authentication mechanism and converted their services to support Kerberos. Users on these sites therefore need Kerberos TGT to access local services, including the very basic one such as user accounts or data storage. A mechanism converting X.509 certificates to Kerberos TGT's is a desired feature when such site decides to contribute their resources to a Grid community.

There is an IETF standard PKINIT [24] that specifies how public key cryptography can be integrated with the initial authentication exchange of Kerberos to obtain a TGT. The first open source implementation of the PKINIT was contributed by CESNET developers to the Heimdal implementation of Kerberos. Current Heimdal implementation even supports proxy certificates on the client side.

5.4 Converting Kerberos TGT to X.509

As described above, a lot of institutions use Kerberos as the primary authentication mechanism and their users are accustomed to its use. When the user wants to access a Grid based on PKI, they need to learn about digital certificates and private keys and how to obtain and manage them. It would be much easier if they could get digital certificates transparently, using their Kerberos TGT.

The situation is similar as with the SAML to certificate transition. In this case there are two CAs supporting Kerberos authentication: Kerberos CA (kCA) [25] and MyProxy CA. kCA is a

specialized CA developed to support only this single transformation, while MyProxy CA can be configured to support multiple source credential types. When MyProxy is used as an repository of proxy certificates, the very same configuration can be used to access proxy certificates. The advantage of the latter way is that users can use certificates issued by any CA.

5.5 Converting OTP to Kerberos TGT

For completeness sake we also mention the possibility to obtain a Kerberos TGT using OTP. That possibility has been discussed by the IETF group responsible for Kerberos standardization. An IETF draft describing protocol extension is available but to our best knowledge it is not implemented yet by any Kerberos distribution.

As an alternative solution MyProxy PAM module together with PKINIT PAM module can be used. Both modules were devoloped by the METACentrum. After successful authentication user receives the proxy or common certificate from the MyProxy using OTP authentication this is done by the MyProxy PAM module. The resulting certificate is then used by the PKINIT PAM module which succeeds the MyProxy PAM module. PKINIT PAM module then contacts KDC, sends the certificate and obtains user's ticket. This whole procedure is completely transparent for the user, she only see after successful authentication new proxy or common certificate and kerberos ticket ready for use.

5.6 Transitive translations

In some cases a simple transition may not be sufficient, especially in complex environments with extended workflows encompassing many services. Individual services, through which data flows, may require different authentication mechanisms. In such case, a chain of transitions can be generated, making sure that each service is presented with a credential it expects.

A different possibility is to create several credentials at the moment of user's first contact with a system. User is able to submit appropriate credential to any service he needs; however, high risk of compromise (theft) is associated with this approach.

6 Deployment in METACentrum

A detailed description of our deployment scenario will be described here

7 Conclusion

Authentication is the first step each user performs to access large scale data and computing infrastructures. Also, individual grid components and services must authenticate to other ones, using either their own identity or acting on behalf of a user. Using a single authentication protocol like a PKI in a large heterogeneous grid may not be flexible and convenient—users may be forced to use mechanisms they are not accustomed to and some legacy services may not be able to recognize it. After an overview of basic authentication mechanisms used in nowadays grids, we discussed approaches that can be used to increase the flexibility while keeping the high security standard.

The federated approach offers users to stay with their primary authentication mechanism (usually the one provided by their employer) to access any federation enabled service. Coupled with an on-line CA, the federated approach can be used to open PKI grids to users without long-term certificate. Similarly, using certificates from the on-line CA it is possible to obtain a Kerberos to access local services at an institution, etc.

The credential transition providers allows to build a framework for seamless and transparent translation of user credentials between different authentication protocols. Users initiates their grid access with an authentication method that best suits their actual requirements (e.g., a full certificate from his or her own notebook and OTP when working from an Internet cafe) and credentials required by individual services are created on the fly as needed.

We are working both on further development of the promising approaches and also in their early deployment within the production grid infrastructure of the Czech national infrastructure $\mathcal{META}$Centrum.

In the future we plan to investigate possibilities proposed by the WS Trust standard [**?**], which provides a unified way how credential transitions can be performed.

Acknowledgment

The work has been supported by the research intent "Optical Network of National Research and Its New Applications" (MSM 6383917201) of the Ministry of Education of the Czech Republic.

References

1. Ylonen, T., C. Lonvick, E.: The Secure Shell (SSH) Authentication Protocol. IETF RFC 4252 (2006)
2. CVE-2008-0166: vulnerable random number generator in openssl on Debian-based systems `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-0166`.
3. 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`.
4. Housley, R., Polk, W., Ford, W., Solo, D.: Internet X.509 Public Key Infrastructure—Certificate and Certificate Revocation List (CRL) Profile. IETF RFC 3280 (2002)
5. Ellison, C.: SPKI Requirements. IETF RFC 2692 (1999)
6. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI Certificate Theory. IETF RFC 2692 (1999)
7. Streit, A., Erwin, D., Lippert, T., Mallmann, D., Menday, R., Rambadt, M., Riedel, M., Romberg, M., Schuller, B., Wieder, P.: UNICORE – From Project Results to Production Grids. In: Grid Computing: The New Frontiers of High Performance Processing. Volume 14 of Advances in Parallel Computing. Elsevier (2005)
8. Tuecke, S., Welch, V., Engert, D., Pearlman, L., Thompson, M.: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. IETF RFC 3820 (2004)
9. Kouřil, D., Basney, J.: A Credential Renewal Service for Long-Running Jobs. In: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing (GRID 2005). (November 2005)
10. Novotny, J., Tuecke, S., Welch, V.: An Online Credential Repository for the Grid: MyProxy. In: Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10). (August 2001)
11. Basney, J., Humphrey, M., Welch, V.: The MyProxy Online Credential Repository. Software: Practice and Experience (2005)
12. Neuman, C., Yu, T., Hartman, S., Raeburn, K.: The Kerberos Network Authentication Service (V5). IETF RFC 4120 (July 2005)
13. Neuman, B.C., Ts'o, T.: Kerberos: An Authentication Service for Computer Networks. IEEE Communications **32**(9) (September 1994) 33–38
14. Cantor, S.: Shibboleth Architecture—Protocols and Profiles `http://shibboleth.internet2.edu/shibboleth-documents.html`.
15. Maler, E., et al: Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1 (September 2003) OASIS.
16. GridShib: Project web page (April 2008) `http:/gridshib.globus.org/`.
17. ShibGrid: Project web page (April 2008) `http://www.oesc.ox.ac.uk/activities/projects/index.xml?ID=ShibGrid`.
18. Haller, N.: The s/key one-time password system. IETF RFC 1760 (1995)
19. Haller, N., Metz, C., Nesser, P., Straw, M.: A one-time password system. IETF RFC 2289 (1998)
20. Lamport, L.: Password authentication with insecure communication. In: Communications of the ACM. (1981)
21. Kuhn, M.: Otpw - a one-time passoword login package (2003)
22. M'Raihi, Bellare, M., Hoornaert, F., Naccache, D., Ranen, O.: Hotp: An hmac-based one-time password algorithm. IETF RFC 4226 (December 2005)

23. Kouřil, D., Matyska, L., Procházka, M., Kubina, T.: Kerberos and identity federations. AFS & Kerberos Best Practices Workshop 2008 http://workshop.openafs.org/afsbpw08/talks/thu_2/kouril.pdf.
24. Zhu, L., Tung, B.: Public Key Cryptography for Initial Authentication in Kerberos (PKINIT). IETF RFC 4556 (2006)
25. Kornievskaia, O., Honeyman, P., Doster, B., Coffman, K.: Kerberized Credential Translation: A Solution to Web Access Control. In: Proceedings of the 10th USENIX Security Symposium. (2001)