Protecting Services with Smartcard-Based Access Control: A Case Study at Technical University Berlin

Interdepartmental Research Center for Networking and Multimedia Technology
PRZ / FSP-PV / TUBKOM
Technical University Berlin
Thomas Hildmann, Thomas Gebhardt
hildmann@prz.tu-berlin,de, gepard@prz.tu-berlin.de

Abstract

Technical University Berlin is in the process of issuing smartcards to employees and students and providing a wide range of campus-related services over the internet. Therefore an infrastructure supplying security services like user-authentication, secure connections and access control is necessary. A further goal is the reuse of existing applications and network technology to keep costs reasonably low. This calls for an application-independent, highly flexible security framework. Security measures must also be scalable, since applications operate on different levels of confidentiality. This paper describes the implemented security framework, which uses application level firewalls to implement smartcard-based authentication and a Single-Sign-On (SSO) mechanism. It will be pointed out that such a system will only be maintainable in the future if role-based access control is introduced. The paper describes the migration from a password- to a smartcard-based authentication which will be extended with a role-based access control (RBAC) mechanism in the next step.

1 Introduction

The introduction of smartcards is an important part of the ongoing restructuring process at the Technical University Berlin which will make services and applications available over the internet/intranet to employees and students without sacrificing security or confidentiality. The main goals are:

- Smartcards as means for identification, authentication and access control.
- Streamlining the IT-infrastructure to cut costs.
- Migrating services to a web-based client-server architecture.
- Conforming to german privacy law.
- Integration of legacy applications.
- Scalable security, depending of the confidentiality of the application data.

As an example we will look at the migration of the TUB-specific application LinF (“Leistungserfassung in der Forschung“, which approximately means “measurement of scientific performance“), from a password-based to a certificate-based authentication. LinF is composed of a database and a HTML-frontend and represents a class of applications widely in use today in many organisations. Employees of the different faculties are using the frontend to enter data about scientific activities into the database and this data is evaluated later. This means that user authentication is necessary and the result of the authentication must be mapped into the user model of the database.

2 Development Steps

Here we discuss the technolgies used in the migration of the LinF application and their advantages and disadvantages. We also look at the next steps of the development, especially the role-based access control.

2.1 The Single-Sign-On Mechanism (SSO)

Today the protection of certain web-pages against unauthorised access is a common task in distributed network architectures. HTTP provides the necessary features, the password-based basic-authentication mechanism and the use of SSL-encrypted connections. But difficulties arise if a seamless transition to another web-server is needed, or even to an application not based on HTTP but other protocols. There are many different network management architectures in existence today [1] and at TUB a very heterogeneous network structure has evolved over the years. Therefore we developed a simple system, which implements Single-Sign-On without the need for centralised SSO-servers, using public/private-key management.

The basic idea of the system is that, after a successful authentication, a person receives a ticket, which contains user data that can be converted for use by different applications. The ticket is valid for a limited amount of time and is digitally signed. This means the ticket can be stored on the client host, via the cookie-mechanism or a dedicated browser plugin, without the danger of undetected manipulation.

Every server in the SSO domain needs to know only the shared SSO-public-key to verify the digital signature of the user ticket. For easy maintenance a standard configuration for web-servers has been developed, which can be applied in the most cases. This configuration is described below:

- In front of the web-server hosting the service that should be protected, a proxy-server is placed to enforce access control. The web-server itself must evaluate variables that are transmitted by the proxy-server and contain user data extracted from the signed ticket, in the most simple case, when there is no user data, even this is not necessary. Figure 1 shows this architecture.

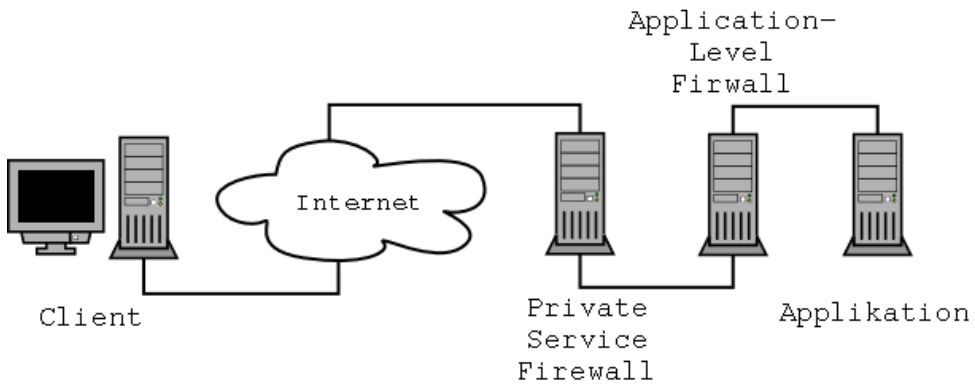

Figure1: logical architecture of the secured application

- The proxy-server is running a filter (i.e. a CGI or Servlet), which is applied for every request. The filter uses the algorithm that is described in a simplified from in figure 2.

    1. Delete all variables that contain user data.

    2. If ticket verification is successful: assign the user data from the ticket to the appropriate variables and forward the request.

3. If ticket verification fails: send authentication page.

4. If authentication is successful: generate ticket, store it in a cookie and load the requested page.

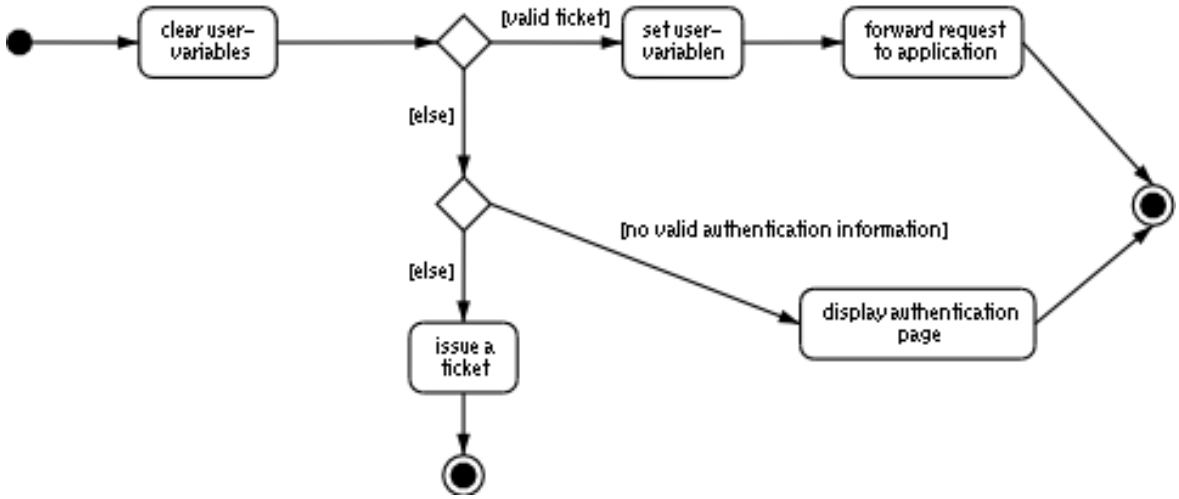

Figure 2: simplified filter algorithm

It is mandatory to use SSL-encrypted connections to protect the username and password, transmitted during the authentication, and to avoid that the ticket can be copied. In our configuration the proxy-server executes this task, too. It is only accessible via SSL-connections. The application web-server is connected to the proxy over a crossed TP-cable and is running plain HTTP. This gives us the following advantages:

- development of the LinF-application and the security module could be executed, for the most part, independently.
- the CPU load for connection encryption and signature verification is completely located on the proxy-server, which increases the scalability of the system.
- proxy-servers can be configured, upgraded, tested and exchanged independently from the application.

Normally GUI designers do not have sufficient knowledge on security issues while on the other hand security experts usually do not know enough about the user's needs. Being a security expert is a full-time job. Being a GUI designer and doing requirement elicitation, too. Therefore having separate security and GUI modules can help to design secure and usable applications.

From the security point of view it is the right way to have very small and simple modules which can easily be reviewed, fixed or totally replaced depending on the latest news, discovered bugs or changed requirements.

Of course packet-filtering firewall cannot only be placed in front of the proxy-server but also between the proxy-server and the application-server to implement even higher security and a more flexible network architecture, i.e. to permit direct access to the application from within a private LAN. This approach is also used by integrated commercial security solutions like HPs OpenView webQoS [2].

The disadvantage of using password-based SSO is clearly the necessity to distribute passwords among the users, which yields a set of common problems in regard with password management. Also access lists containing the legitimate users must be distributed to all servers wanting to create their

own tickets. The latter problem could be solved by using a central authentication server, all proxy-servers would then forward users in need of authentication to this server. We consider this design unsatisfactory since a centralised server composes a single point of failure which can lead to the malfunction or corruption of the whole system. A centralised authentication server could also be used for unapproved gathering of user data, the legal requirements of our system are clearly prohibiting this.

Not surprisingly, the distribution and management of passwords was the major practical problem in this stage of the development. The initial passwords given to the LinF users where random passwords and many users had problems to enter them correctly. Some forgot to keep the envelopes containing the passwords. Instead of changing the passwords many users simply glued a post-it note with the password to their monitor. Often the passwords where given to colleagues to simplify the working routine. Passwords chosen by the users often were to easy to guess, so rules for the creation of passwords had to be enforced.

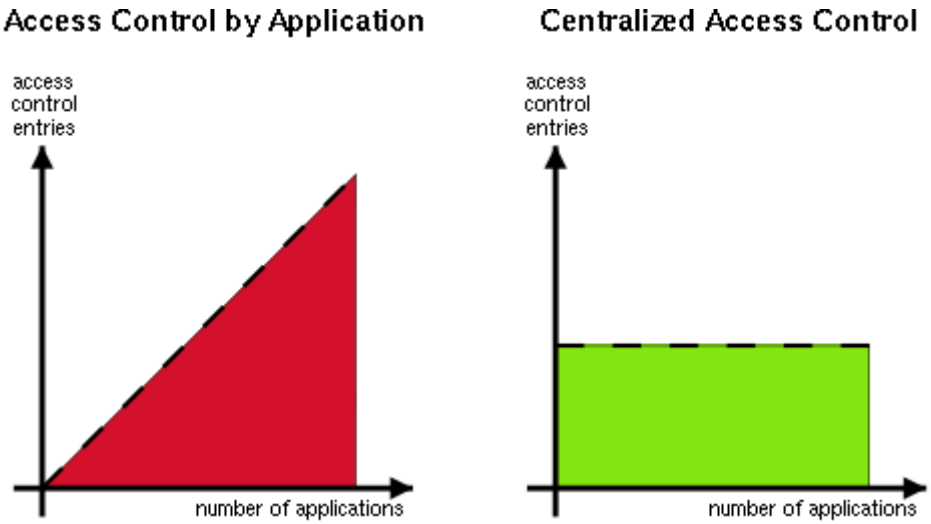

Figure 3: comparing ac by application vs. centralized ac

Figure 3 shows the trend of expense using an isolated access control system for each application compared to a centralized approach. Both diagrams assume a constant number of users. The optimal access control system has to combine the advantages of the centralized access control with the availability of decentralized approaches.

2.2 Certificate-based Authentication

If the authentication via username and password is not sufficient, the system can perform an authentication process based on X.509 certificates. TUB is in the process of creating its own trust-centre to establish a public key infrastructure (PKI) and other universities will be integrated into the PKI. Certificates and private keys can be stored on hard-disks (i.e. a software personal security environment) or smartcards. For the TUB campuscard, javacards containing a cryptographic coprocessor are used to store certificates and keys for authentication, encryption and secure email and to execute cryptographic operations on the card. The first steps towards this solution were implemented in the E2S project (End-to-End-Security over the Internet) and since then have been further expanded and customized [3]. The question if password-authentication, certificates in software PSEs or certificates on smartcards should be used can be decided separately for every single application.

In the first phase of the TUB project “Campuskarte” an infrastructure will be established that allows all web-based applications to use smartcard-based authentication. The migration of other services and the creation of new services are planned.

At this point, we should take a closer look at the password related problems, described in the first development step. The sharing of password between co-workers is of course not a sign of malevolence or incompetence on behalf of the TUB employees! It is simply a pragmatic way of creating a useful division of labour within departments or teams, that is done in a dynamic, non-bureaucratic way. By introducing smartcards alone a big part of this useful flexibility would be no longer possible, since the cards are serving multiple purposes, i.e. they are also identification cards, and will be used for the digital signing of documents. Employees therefore cannot give their cards and PINs to their fellow co-workers. To avoid the introduction of application specific smartcards and still keep this flexibility in the day-to-day work processes, we need a means for the users to nominate deputies and to grant rights to other users.

2.3 Role-based Access Control

A certificate-based authentication alone grants either *all* rights to a user or *none* at all. Many applications dealing with the administration of TUB of course need a more sophisticated access control. I.e. a student may use an application to register for an exam, but only the examiner may enter a result. Right now those access right are mostly achieved through operating system mechanisms and are managed by system administrators. It is possible to embed access rights into the certificates but this strains the PKI in cases where rights are changing frequently. For every change new certificates would have to be issued and the old ones revoked, therefore we deem this approach not suitable for our case.

The management of access control lists already occupies a great amount of time of the system administrators and the demand will continue to grow. Often small changes to the access policy will have to be implemented very fast creating the danger of inconsistencies and security holes. Access right will no longer be efficiently manageable on this level.

Role-based access control is a state-of-the-art, well researched alternative to conventional access control lists [4]. TUB has examined and developed RBAC solutions in the context of the ESPRIT project MultiPLECX (Multiparty Processes for Large-scale Electronic Commerce Transactions) [5]. Unlike most other RBAC systems the TUB system employs a server-pull method. This means that the roles and access rights are not assigned statically at the login, for each request the RBAC system evaluates the role hierarchy for specific permissions. This allows a very dynamic application of role memberships and the definition of permissions depending on runtime parameters like time of day or available resources [6]. The role model is defined using an object-oriented modelling language developed at the PRZ institute of TUB. There have been various proposals for object-oriented languages to specify policies, i.e. by Lupu, Sloman, et al. [7]. But often the implementation aspect is left open. At the PRZ we implemented the working prototype of an interpreter for our language.

An important feature for the use in an administration context is the ability of the TUB RBAC system to apply the access control to the role model itself. This allows the delegation of certain parts of the model administration to the departments and workgroups of the application domain. The model itself supports the generation of customised user-interfaces for such administration tasks. In this way the head of a department can assign roles to his employees or external users and proxies can be assigned the appropriate roles for a limited time frame, all this without consulting the administration of the core RBAC system.

The work of the system administration can concentrate more on the correct definition of roles and access rights, while large parts of the user administration can be executed by the users themselves. This gives the users back the needed flexibility and eliminates bureaucratic delays that are imminent in a centrally managed RBAC model.

To achieve these features our RBAC implementation is designed as a distributed system. It allows the creation of a network of RBAC servers throughout the organisation, each one holding a part of the model. A set of servers can also be used for load balanced interpretation of very large models. All logging data generated by the RBAC system can be digitally signed and encrypted to ensure the confidentiality and integrity of the gathered information. The distribution and the trust boundaries between the models also ensure the confidentiality of the model data, which can contain sensible

information about business processes and relations. It also allows to customise the system to comply with the strong privacy laws that are applied to government organisations in Germany.

3 Conclusion and Outlook

Our development work over the last years led us from the implementation of a password-based authentication system towards the use of certificates and smartcards. In the context of the project “Campuskarte” at he TU Berlin we demonstrated that the migration to a smartcard-based authentication can be achieved without dramatic changes to the application itself.

To provide users with enough flexibility within their workplace context and to reduce bureaucratic hindrances we plan to establish a distributed role-based access control system. The working prototype implemented at the PRZ institute of TUB will be further developed for use in production environments. Also the authentication mechanism will be enhanced to support new protocols.

Securing a single application with a “private” application-level-firewall can be very expensive. Depending on the value of the protected data such a solution can be worth each penny. On the other hand there must be a more scalable solution protecting each application even if for example an other server behind this firewall is hacked.

Figure 4 shows how firewalls and gateways can be combined in a way that gives high flexibility without decreasing the amount of security for each application server. The idea is to place each server in a so called security-cell which is a part of a security-zone. Such cells can be enforced using a dedicated network interface on the inner firewall with specialized firewall rules protecting each interface against attacks from other inner interfaces. Several application server can share a single gateway or several gateways can protect a single server if high load on this server makes load-balancing necessary. These ideas are in alpha state but will be discussed in detail in future publications.

The PRZ is interested in collaborations with companies and institutions working in on related topics or implementing similar solutions.

4 References

[1] Hegering H.G., Abeck S., Neumair B.: *Integrated Management of Network Systems*, Morgan Kaufmann, 1999

[2] HP OpenView webQoS with HP virtualvault, http://www.hp.com/security/products/virtualvault/solutions/webqos/

[3] Nagel K.: *User Requirements for Administration Sectors*, E2S/WP.C/TUB/001, End-to-End Security over the internet, 1996

[4] Sandhu R.S., et al.: *Role Based Access Control Models*, IEEE Computer, 1996, Vol. 29, No. 2

[5] Hildmann T., et al: *The Decentralised Trust Center*, Deliverable Phase 3, WP F, Multiparty Processes for Large-scale Electronic Commerce Transactions, 1999

[6] Gebhardt T., Hildmann T.: Enabling Technologies for Role-Based Online Decision Engines, in *Proceedings of Fifth ACM Workshop on Role-Based Access Control*, Berlin, Germany, 2000, ACM Press

[7] Lupu E., Sloman M., Dulay N., Damianou N.: *Ponder: Realising Viewpoint Concepts*, HP OpenView University Association (HP-OVUA), 7th Plenary Workshop, Santorini, Greece, 2000

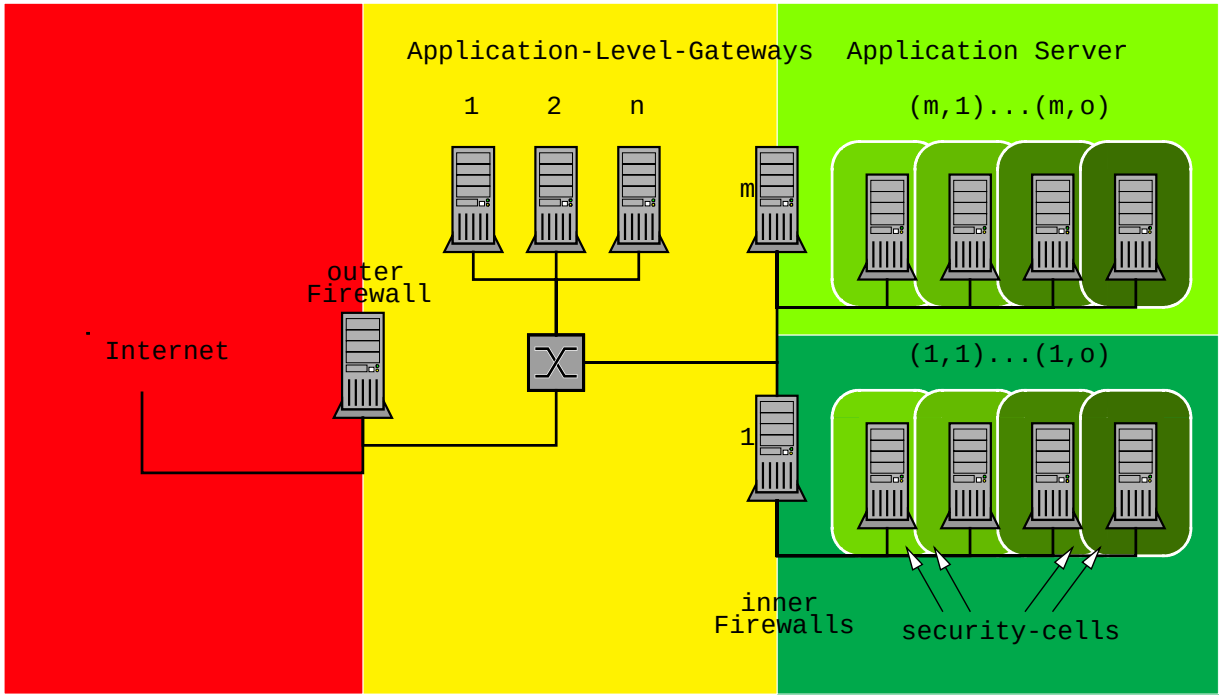

Figure 4: architecture of a scaled application firewall system