

PAM.NET: A .NET Framework For Pluggable Authentication Modules

Bert Lagaisse Frank Piessens

Report CW 361, June 2003



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

PAM.NET: A .NET Framework For Pluggable Authentication Modules

Bert Lagaisse *Frank Piessens*

Report CW 361, June 2003

Department of Computer Science, K.U.Leuven

Abstract

Authentication is one of the core security services of a distributed application platform. A variety of mechanisms to perform authentication have been developed, and it is beneficial if the choice for a specific mechanism is not hard-coded in an application. Pluggable Authentication Modules (PAM) is the name of a technology to support flexible changes with respect to the authentication mechanisms that an operating system or application uses. In this report, we describe an implementation of this technology, PAM.NET, on the .NET Common Language Runtime, Microsoft's new application platform. PAM.NET is well integrated with .NET's role-based access control, and enables a developer to build platform independent, extensible and configurable authentication and access control into an application.

1 Introduction

Authentication is one of the core security services of a distributed application platform. A variety of mechanisms to perform authentication have been developed, including username/password-based mechanisms, token-based mechanisms, biometrics and protocols for authentication over a network. It is bad practice to hard-code an authentication mechanism in an application for various reasons:

- Hard-coding the mechanism makes it difficult to replace the mechanism in case it is found to contain vulnerabilities.
- Different applications will use different mechanisms making it harder (if not impossible) to realize single sign-on.
- A system administrator can not change an authentication mechanism to be in line with company policy.

Pluggable Authentication Modules (PAM) [5] is a technology that supports pluggability of authentication mechanisms. It was originally designed to make authentication in operating systems pluggable, but is also incorporated in the Java platform under the name of Java Authentication and Authorization Services [2].

PAM is a software framework that offers a standardized authentication API to applications. Various implementations of authentication mechanisms can then be plugged into the framework, and a configuration file determines what modules should be activated for a particular authentication attempt.

Microsoft's new application platform, the .NET Framework ([6]) does not come with support for pluggable authentication. Authentication of a user in a .NET application is possible in two different ways. The .NET Common Language Runtime (CLR) can use the authenticated user from the underlying Windows operating system. Or the developer can implement an authentication system himself, but with little support.¹

In this paper, we first discuss why this current support for authentication in the CLR is insufficient, and then we discuss a possible solution: PAM.NET. PAM.NET is an implementation of the PAM framework on the .NET CLR. Together with .NET's role based access control mechanism, it provides a user-centric, role-based security system for authentication and authorization.

2 Problems with the current user authentication infrastructure in .NET

Authorization in .NET's role-based access control system is based on the user's identity or his roles defined in the thread's principal. This principal can be a `WindowsPrincipal` when the authenticated user of the underlying Windows operating system is used. Or it can be a `GenericPrincipal`, when the developer has implemented the authentication and created the principal himself [3].

Relying on Windows authentication has a number of disadvantages:

- The roles of the `WindowsPrincipal` are the Windows groups to which the user belongs. As a consequence, application-specific roles will have to be added to the Windows operating system, with for instance the risk of name clashes with roles of other applications on the same machine.

¹ASP.NET introduces other ways of authenticating a user: those are not considered here.

- Windows authentication is obviously not present on non-Windows implementations of the CLR. In particular, the Shared Source Common Language Implementation [1] does not support it.

Developing a custom authentication scheme also has its disadvantages:

- Implementing authentication logic is extremely error-prone.
- Hard coded authentication logic makes it hard or impossible for a system administrator to change the authentication system, e.g. to implement single sign-on, or to comply with company policy.

One possible solution is to offer a standardized framework for authentication services, in which reusable implementations of authentication technology can be plugged in. PAM.NET is such a framework. We describe it in the following sections.

3 PAM.NET: A PAM-based authentication framework

3.1 Architecture

In this section the architecture of PAM.NET will be explained. Key concepts of this architecture are sessions, principals and credentials. After that, the authentication model of PAM.NET is discussed. Important parts of this model are the authentication stack and authentication modules. On top of this model an abstract, implementation-independent interface is defined: the authentication context. The architecture of PAM.NET is similar to the implementation of PAM in the Java Authentication and Authorization Service [2].

3.1.1 Sessions

A session object represents a session of a user. A user has a session for each service he interacts with. The session contains security-related material needed during interaction with a service.

3.1.2 Principals

A principal object represents an entity that can be authenticated (e.g. a user). Principals are already a part of the .NET security system. Principal objects can be attached to a thread, indicating that the thread is executing on behalf of that principal. Based on the principal of the current thread, the .NET role-based access control system allows or disallows operations in an application. A principal contains the identity of a user and his roles.

During a session, a user can be authenticated as multiple principals. Each principal is stored in the session. To activate an obtained principal during a session, the session attaches the principal to the current thread. The .NET role-based access control system will make its decisions based on the currently activated principal.

3.1.3 Credentials

Next to principals, also other security-related material can be associated with the session. This generic security-related material is called *credentials*. This material can contain information to authenticate the user to other services, so

a single sign-on system can be obtained. Examples of such credentials are passwords, certificates, Another type of credentials gives the user the potential to do a certain action. An example of such a credential is a cryptographic key to sign data.

The credentials are divided into two parts. The first part are the public credentials like certificates, the second part are the private credentials like private keys or symmetric keys.

The specification of the Session class and IPrincipal interface:

```
namespace System.Security.Principal{
    public interface IPrincipal{
        IIdentity Identity{get;}
        bool IsInRole(string role);
    }
    public interface IIdentity{
        string AuthenticationType{get;}
        bool IsAuthenticated{get;}
        string Name{get;}
    }
}
namespace Be.Ac.Kuleuven.Cs.Pam{
    public sealed class Session{
        IPrincipal[] Principals{get;}
        void AddPrincipal(IPrincipal principal);
        void Activate(IPrincipal principal);
        ArrayList Publics{get;}
        ArrayList Privates{get;}
        void Clear();
    }
}
```

3.1.4 The authentication modules

The PAM.NET authentication framework is based on pluggable authentication modules (PAM). System administrators can plug authentication modules into an application according to their authentication requirements. The application has an abstract interface to the whole authentication system: the authentication context. This is explained in the next section. Updating and reconfiguring the authentication modules can occur in an application-independent way.

The IAuthenticationModule interface:

```
namespace Be.Ac.Kuleuven.Cs.Pam{
    public interface IAuthenticationModule{
        bool Authenticate();
        bool Commit();
        bool Abort();
        bool Logout();
    }
}
```

PAM.NET uses a configuration file to decide which authentication modules should be loaded for a certain application. More details about the configuration are explained in one of the next sections.

3.1.5 The authentication stack

Another concept in a PAM-framework is the authentication stack. All configured authentication modules for an application are put on a stack. For the moment, the requirements of an authentication module will be simplified, assuming that all of the modules have to succeed or else none of them may succeed. Therefore PAM.NET uses a two-phase commit algorithm. In a first phase all modules are asked to start authentication. If all succeed, a second iteration on the stack is started to commit each module. If a failure occurs during the authenticate or commit phase, all modules will be aborted. During the commit phase the authentication module will associate the obtained security-related

material with the session. This material consists of principals and credentials. When an authentication module has to abort, it will clean up all associated principals and credentials in the session.

3.1.6 The authentication context

The authentication context is the abstract interface on the stack and authentication modules in the PAM.NET framework. It is used in an application to request the PAM.NET authentication service and ask for authentication of the user. This happens based on the application's configuration made by the system administrator.

Specification of the AuthenticationContext class:

```
namespace Be.Ac.Kuleuven.Cs.Pam{
    public sealed class AuthenticationContext{
        public AuthenticationContext(string app){}
        public void Authenticate(){ }
        public void Logout(){ }
        public Session Session{get{}}
    }
}
```

3.2 The configuration of PAM.NET

PAM.NET uses an XML configuration file to load the authentication modules for a certain application. Each configuration entry defines one authentication module for one application. So the entry contains the name of the application, the authentication module, a flag for the requirements of success of the module and module-specific options.

In the previous section we oversimplified the requirements of the different modules on the stack. Not all modules are required to succeed. There are four different types of modules on the stack, based on their necessity to succeed. In the configuration file the system administrator specifies the flag defining the requirements of a module.

The four different types are:

- **Requisite:** The module must succeed. On failure, the iteration of the stack is stopped and an exception is thrown immediately.
- **Required:** The module must succeed. On failure, the stack is further iterated. At the end, an exception about the first failure is thrown.
- **Optional:** The module is optional. On failure, this will not lead to overall failure and the stack is further iterated.
- **Sufficient:** The module is optional, but on success the authentication method will report success immediately (if no required module prior to the current one failed).

The module-specific options, defined in the configuration file, are passed to each loaded module. These options can be anything: the prompt of the password box, the location of the log file, . . . How the options for the framework and the loaded modules are specified in the configuration file is illustrated in the next example.

Illustration of a PAM.NET configuration file:

```
...
<entry>
    <name>test3</name>
    <class>AuthenticationModules.TextfileAuthentication</class>
    <assembly>
```

```

        textfileauthentication ,
        Version = 1.0.0.0,
        Culture=neutral,
        PublicKeyToken=e422e913d06d6c79
    </assembly>
    <controlflag>required</controlflag>
    <options>
        <userfile>users.xml</userfile>
        <hash>MD5</hash>
        <usernameprompt>Loginnaam:</usernameprompt>
        <pwdprompt>Paswoord:</pwdprompt>
        <logspace>logmessage</logspace>
    </options>
</entry>
<entry>
    <name>test3</name>
    <class>AuthenticationModules.LogModule</class>
    <assembly>
        logmodule,
        Version = 1.0.0.0,
        Culture=neutral,
        PublicKeyToken=7ee1855643ac0722
    </assembly>
    <controlflag>optional</controlflag>
    <options>
        <logspace>logmessage</logspace>
        <logfile>log.txt</logfile>
    </options>
</entry>
...

```

3.3 Protection of the PAM.NET framework

The PAM.NET assembly and the authentication modules used on a system are located in the *Global Assembly Cache* (GAC). This ensures that an attacker cannot endanger the integrity of the framework, nor try a spoofing attack with authentication modules created by the attacker. Only the system administrator is able to add, remove or modify the modules in the GAC. The integrity of the configuration file is guaranteed because it is allocated in the system directory. Normally, only system administrators have write access in this directory.

4 Supported platforms

The PAM.NET framework is designed to be compliant with the ECMA standard of the .NET CLI. The commercial CLR as well as the Shared Source CLI ([1]) are supported. The architecture of PAM.NET relies heavily on the Global Assembly Cache of the .NET platform. Because MONO ([4]) still lacks this GAC, PAM.NET is not supported on this runtime environment. Using PAM.NET outside the GAC, with authentication modules not located in the GAC, exposes the authentication framework to possible spoofing attacks.

5 Conclusion

PAM.NET introduces a PAM-framework into .NET. The existing authentication schemes supported in the .NET CLR have some problems related to extensibility, configurability, maintainability and pluggability. The PAM.NET framework solves these problems and releases the developer from the authentication responsibility within an application. This responsibility is moved to the system administrator, the actual person in charge of this. Extending .NET security with a widely-used framework like PAM enlarges the possibilities of using the .NET role-based access control system for any kind of user-based or role-based security.

References

- [1] Ted Neward David Stutz and Geoff Shilling. *Shared Source CLI Essentials*. O'Reilly, 2003.
- [2] C. Lai, L. Gong, L. Koved, A. Nadalin, and R. Schemers. User authentication and authorization in the Java platform. In *15th Annual Computer Security Applications Conference*, pages 285–290. IEEE Computer Society Press, 1999.
- [3] B. LaMacchia, S. Lange, M. Lyons, R. Martin, and K. Price. *.NET Framework Security*. Addison Wesley, 2002.
- [4] <http://www.go-mono.org>.
- [5] V. Samar and C. Lai. Making login services independent from authentication technologies. In *Proceedings of the SunSoft Developer Conference*, March 1996.
- [6] Thuan Thai and Hoang Q. Lam. *.NET Framework Essentials*. O'Reilly, 2002.