Question
Stephen Wilson · Aug 21, 2019

ASP .NET Core Identity (v3) uses PBKDF2 algorithm with HMAC-SHA256, 128-bit salt, 256-bit subkey, and 10,000 iterations. What does Caché use?

The .NET Core Identity model has an IPasswordHasher<> interface for for

- Hashing a password so that it can be stored in a database
- Verifying a provided plain-text password matches a previously stored hash.

I am getting invalid password errors during the login process when the .NET Core Identity model computes a hash from a plain text input and compares it to a password hash value I've returned from Caché. The default hashing algorithm is PBKDF2 with HMAC-SHA256, 128-bit salt, 256-bit subkey, and 10,000 iterations (detailed article on .NET Core Identity PasswordHasher). The algorithm Caché uses is probably different which may be why I am getting errors.

Consider the following in Caché for username "test1"

```
Do ##class(Security.Users).Exists("test1",.user,.status1)
```

The "user" object has properties for Password and Salt.

The class documentation says it uses PBKDF2 and that the Salt value is generated from $System.Encryption.GenCryptRand but doesn't elaborate on the other properties of PBKDF2 such as key length and number of iterations.

I encode the value of user.Password as UTF8 then Base64 before passing it to the web app to compute and verify it's own hash based on plain-text input.

```
$system.Encryption.Base64Encode($ZCONVERT(user.Password),"O","UTF8")
```

This produces something like w67CisO6IGnDk1fDl8OzC8OjYk0bblknA8KYw4g= as the PasswordHash stored in the database. The .NET Core Identity hashing technique must be producing something different which is why I am getting errors.

Any thoughts on how I might solve this problem?

#.NET #ObjectScript #Security #Caché

Source URL:https://community.intersystems.com/post/asp-net-core-identity-v3-uses-pbkdf2-algorithm-hmac-sha256-128-bit-salt-256-bit-subkey-and