MYOB Exo Business White Paper

Secure Third-Party Logins for Developers

Last modified: 13 August 2020

myob

Introduction

In release 2019.4.1 of Exo Business, we updated the login parameter for launching Exo applications. You can learn more about this in the MYOB Exo Business Release Notes for release 2019.4.1.

This whitepaper contains sample Pascal and C# code that you can use with a third-party application to take advantage of this enhancement.

Sample Pascal Code

Below is a sample of Pascal code that you can incorporate into a third-party application to take advantage of the **<ThirdPartyLogin>** connection parameter:

```
unit ThirdPartyUtils;

interface

{
  Sample code to read and check third party login information
}

function CalcHash(const AppPassword, Salt: string): string;
function CheckPassword(const AppPassword, MapName: string): Boolean;

implementation

uses
  Winapi.Windows,
  System.Classes,
  System.SysUtils,
  System.Hash,
  System.NetEncoding;

function CalcHash(const AppPassword, Salt: string): string;
var
  lHashVal: string;
  HashBytes: TBytes;
begin
  lHashVal := Salt + AppPassword;
  HashBytes := THashSHA2.GetHashBytes(lHashVal);
  Result := TBase64Encoding.Base64.EncodeBytesToString(HashBytes); end;

function CheckPassword(const AppPassword, MapName: string): Boolean; var
  Hash: string;
  MapHandle: THandle;
  ptr: Pointer;
  MapValue: UTF8String;
begin
  MapHandle := OpenFileMapping(FILE_MAP_READ, False, PChar(MapName));
  if MapHandle = 0 then
    RaiseLastOSError;
  try
    ptr := MapViewOfFile(MapHandle, FILE_MAP_READ, 0, 0, 0);
    if ptr = nil then
      RaiseLastOSError;
    try
```

```
      MapValue := PAnsiChar(ptr);
      Hash := CalcHash(AppPassword, MapName);
      Result := UTF8Encode(Hash) = MapValue;
    Finally
      UnmapViewOfFile(ptr);
    end;
  finally
    CloseHandle(MapHandle); end;
  end;
end.
```

Exo Business writes the SHA256 hash of the GUID and the hashed value of the password to Exo's memory map.

Using the code above, a third-party application reads the hash value from Exo's memory map, then calculates the hash from the GUID and the password. If the calculated hash is the same as the hash in the memory map, the user is granted access.

Sample C# Code

As a utility class:

```
using System;
using System.IO;
using System.IO.MemoryMappedFiles;
using System.Security.Cryptography;

namespace MYOB.ThirdParty
{
  public static class LoginHandler
  {
    public static string ReadMemoryMap(string MapName)
    {
      var mappedFile = MemoryMappedFile.OpenExisting(MapName,
MemoryMappedFileRights.ReadWrite); // Read fails, but ReadWrite works
      using (var view = mappedFile.CreateViewStream())
      {
        var reader = new BinaryReader(view);
        var contents = reader.ReadBytes(44); // length of Base64 encoded
SHA256 Hash (256bit -> 32bytes -> 44 characters)
        return System.Text.Encoding.UTF8.GetString(contents);
      }
    }

    public static string CalcHash(string Password, string Salt)
    {
      var hashVal = Salt + Password;

      var textData = System.Text.Encoding.UTF8.GetBytes(hashVal);

      using (var sha256 = SHA256.Create())

      {
        var HashBytes = sha256.ComputeHash(textData);
        return Convert.ToBase64String(HashBytes);
      }
    }
  }
}
```

Example calling code (as a console application):

```
using System;
using MYOB.ThirdParty;

namespace ConsoleApp1
{
  class Program
  {
    static string AppPassword =
"FCw5bhfug4/gFA3y+5Rb+gThozGIFwU4zzkVZC8a/TA="; // Password from Staff
table
    static void Main(string[] args)
    {

      foreach (var arg in args)
      {
        if (arg.StartsWith('{'))
        {
          var payload = LoginHandler.ReadMemoryMap(arg);
          var myHashedVal = LoginHandler.CalcHash(AppPassword, arg);
          if (myHashedVal == payload)
          {
             Console.Write("OK");
          }
          else
          {
             Console.Write("Bad " + myHashedVal + " " + payload);
          }
        }
      }
    }
  }
}
```