

Re: Need to encrypt everything due to new California law? Advice please

Source:

<http://www.derkeiler.com/Newsgroups/microsoft.public.dotnet.framework.aspnet.security/2003-06/0301.html>

From: Alek Davis (*alek_DOT_davis_AT_intel_DOT_com*)

Date: 06/28/03

Date: Fri, 27 Jun 2003 19:06:26 -0700

John,

You raise valid and seemingly easy questions, but unfortunately they do not have simple answers. What is clear that you have to store data in encrypted format. What is not clear is how to protect the encryption key. This problem does not have a perfect solution (if you find one, you will be a very rich man ;-)). Any approach you choose will have disadvantages and the more secure you want to make it, the harder you will have to work and the more difficult it will be to maintain. I am not sure why you categorize it as different from storing a connection string, the idea is pretty much the same.

I would start by assessing the risk. You do not say what kind of personal information you are keeping, but it is very important. For example, if you store customer data, which includes credit card numbers, the security risk is significantly higher if you just keep first or last names. In the former case, a potential hacker can be more motivated to spend time and effort trying to break your system than in the latter. You also do not say whether you have direct administrative access to the computer hosting your application, because if you don't you may be limited to very few options.

Anyway, if you do not consider the risk high and just want to comply with the CA law, you can simply embed the encryption key (or passphrase used to generate the key) in the application source code, and obfuscate the assembly, so it is not easily found. This way you will meet the requirements, since you are string data encrypted (I assume that the law does not specify where to store the key). I know, some security-savvy readers will probably shake their heads when they read this recommendation, so I will repeat that you should take this approach only if your data is not really valuable, the security risk is low, and you just want to comply with the law. There may be other exceptions, such as if your site/company so visible that it gives potential hackers enough incentive to break into, etc.

If your data is more valuable or if you want to use a more secure approach,

the common options are: public/private encryption, symmetric encryption, and DPAPI. DPAPI solves a problem of managing the key because it shifts this responsibility to Windows so you (or your application) do not even know what the encryption key is. The problem with it... well... there are quite a few problems. First, DPAPI does not work with ASP.NET applications (it requires a loaded user profile), unless you use a machine key (which is not secure because it allow any application running on the system to decrypt data). You can use an MSDN example and implement DPAPI with serviced component, so it can be used from an ASP.NET, but it does not solve the problem of authorization, i.e. how is this component protected from allowing other applications to call it to decrypt data owned by your application. Another risk with DPAPI is that since you do not control the key (the OS does), you are at risk of losing your data in case you change the environment, i.e. move the application to another machine, run DPAPI client as a different user, etc. You would not want it to happen.

If you choose any other approach, symmetric or public/private key encryption, you will have to rely on ACLs, i.e. your key – stored in the configuration file or registry – should be protected by ACLs, which is may be also tricky since this file must be accessible by an ASP.NET process. So the question becomes, if a hacker manages to get access to the system, what would be easier: finding your key in a configuration/file or registry, or reverse engineering your assembly trying to find the key there (if you choose my first suggestion). Maybe this is why Microsoft does not reject the idea of string a secondary entropy to be used with DPAPI for machine keys in the source code, which is essentially the same as storing the key (or entropy from which the key is derived) with symmetric-key encryption (check <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/SecNetch12.asp>). Actually, DPAPI would be worse here for the reason I mentioned above (i.e. implications of moving the application to another server). You can probably spend a lot of time and come up with an ingenious approach, but I bet you that if you think long enough about it, you will still find vulnerabilities, so just do your best, but do not spend too much time looking for something, which does not exist.

I cannot say anything about hardware dongles, although I would not set your hopes high here. The basic problem with any type of software or hardware you will need to solve is: how does it authorize entities, which are allowed to use the key from the ones which are not, and I am not sure that there is a device smart enough to handle this decision.

If you find a good solution, let us know.

Alek

"John Smith" <ruu@pacbell.net> wrote in message
news:98133416.0306261430.144d0c10@posting.google.com...

> *Hi,*

>

> *California, USA has a new law that takes effect soon that says that*

> *any California business that has a website that is hacked and from*

- > *which personally identifiable information is stolen must quickly*
- > *notify the people whose data was stolen. This could be expensive and*
- > *embarrassing. Plus, how would one know if one has had information*
- > *stolen, without an Intrusion Detection System?*
- >
- > *The new law only applies if the data is NOT encrypted, so, it seems*
- > *the easiest way to comply with the law is to encrypt all personally*
- > *identifiable information, as a routine matter.*
- >
- > *The many encryption articles I read seem to focus on encrypting*
- > *database connection strings and passwords. What would you suggest for*
- > *encrypting most of what I store in Microsoft SQL Server 2000? One way*
- > *or another, most of the data would fall under the new law. I use only*
- > *stored procedures to get data into and out of the database.*
- >
- > *I read that the DPAPI is cool, but the only examples I've seen are in*
- > *C#, while I program in VB.NET only. Also, it appears the 'key' is*
- > *composed of the users' login password, but, I think they mean the OS*
- > *level password for the account ASP.NET runs under, not the individual*
- > *user password used in conjunction with my Forms Based security under*
- > *ASP.NET. Is that right? If so, then DPAPI doesn't seem so cool*
- > *anymore, since all users run under the same account – the account*
- > *ASP.NET runs under. I don't think it's OK for one real user to be*
- > *able to use that right as a way to hack into the data of any other*
- > *user.*
- >
- > *I read about symmetrical encryption, but where do I store the key*
- > *safely? Maybe via DPAPI, but running under a Windows Service logged*
- > *in with a different OS account and password? Or, is there some*
- > *hardware dongle I can buy that will store a key?*
- >
- > *It seems better to accept the extra work of encrypting everything now*
- > *than ever having to send out a broadcast email saying data was stolen.*
- > *Plus, it also seems like the right thing to do.*
- >
- > *I've spent hours and hours researching this issue, but couldn't find*
- > *any on point advice, thus this long posting.*
- >
- > *Thanks!*