.NET and vicinity... devCity.NET

fill forms.PDF

Home | Search | Forums | Blogs | FAQ | Newsletter Archive | Site Map | Contact Us | Syndicate

Article Options
Rate Article
Print Article
Add to Favorites
Add to 'Articles To Read'
Email to Friend

Site Menu
View Blogs
View Authors
Become an Author
Author Login

Premium Sponsor
got tools?
SubMain()

Encrypting QueryStrings with .NET

by **Tiberius OsBurn** | Published 09/04/2002 | **Security** , **Web Development** | Rating:

Encrypting QueryStrings with .NET

Once upon a time in the tech world, obscurity was security - this being most true in the early years of the industry, when there were gaping holes in privacy policies and confidential client information was bandied about from site to site without a care as to who actually could read the information.

With the new Cryptography classes in .NET, there's absolutely no excuse for not hiding even the most innocuous user data. If you ever need to 'piggy-back' information from one web page to another, whether it is within a POST or a GET parameter, you're passing clear information that anyone can sniff - and that's a bad thing.

If you're not going to use a session variable for storing end user information, you're most likely going to keep some sort of State by passing the information to a cookie or push it around with GET/POST parameters. If you're passing around any sort of ID or user information like their name, it's better to err on the side of caution and encrypt the information.

GET Vs. POST

A POST parameter keeps the information out of the URL, but it can still be sniffed quite easily as it passes in clear text across your network or the Internet. Using POST will keep the mere curious at bay, as the information is not contained in the URL - but this will not stop someone determined to snag out your data.

A `QueryString` parameter passes information within the site's URL. Why would you even use a QueryString? Well, maybe you need to let your user bookmark a particular page, or maybe you have to refer directly to a page in a URL via a link - you can't do either if you're using POST. A QueryString puts data in the URL for the entire world to see, so if you don't know if the end user is malicious, I'd think hard about using a QueryString for anything but site-related information.

Be smart and encrypt any and all data you're moving around from page to page, especially if that in used maliciously. You may trust your users, but you still need that extra level of security that clear te doesn't provide.

Imagine this scenario - you've been passing the customer's ID in the database around in a QueryStr looks like this:

```
http://yoursite.com?cust_id=29
```

Tiberius OsB

Developer/Sy
The Gallup C
(http://www.g
recently com
warehousing
archived data
from 1935 to
coded in C#,
ASP.NET.

tiberi
Hardcore Deve

Tiberius has
experience ir
SQL Server,
various other
Be sure to vis
latest articles
developers.

http://tiberi.us

View all articl
OsBurn...

PDF4NET

PDF enable your .NET applications

You know what a user is going to do? Switch that 29 to a 30 or 12 or some other number, and if you invalid requests, you'll be dishing up some other customer's data.

Enter Encryption

What I was looking for was a quick way to encrypt and decrypt parts of a QueryString - it had to be dirty.

I chose `Base64` because it wouldn't throw bizarre characters in my QueryString that I couldn't pass know that I'd hit a snag while passing around my encrypted QueryString - Apparently, the `Request` object interprets the '+' sign as a space! So, with a quick `Replace` function slapped on my decrypt s foul.

Symmetric Key

The whole trick to this working is that the QueryString is encrypted and decrypted with the same priv secret key - if anyone gets a hold of your key, they can decrypt the data themselves, so keep it a se

We're going to use a hard-to-crack 8 byte key, !#$a54?3, to keep parts of our QueryString secret.

Let's Walk through the C# portion of the code:

Notice our two functions that abstract the dirty work that our `Encryption64` class. The first, `encry` is used to encrypt the value of a QueryString. The second, `decryptQueryString`, is used to decr encrypted QueryString.

```
public string encryptQueryString(string strQueryString) {
    ExtractAndSerialize.Encryption64 oES =
        new ExtractAndSerialize.Encryption64();
    return oES.Encrypt(strQueryString,"!#$a54?3");
}

public string decryptQueryString(string strQueryString) {
    ExtractAndSerialize.Encryption64 oES =
        new ExtractAndSerialize.Encryption64();
    return oES.Decrypt(strQueryString,"!#$a54?3");
}
```

If we wanted to encrypt our QueryString on our first page, we could do something like this:

```
string strValues = "search term";
string strURL = "http://yoursite.com?search="
    + encryptQueryString(strValues);
Response.Redirect(strURL);
```

Inside our code-behind in our second page, we pass the contents our QueryString to a variable nam After that, we replace the '+' signs that our wonderful `Request.QueryString` has replaced with a that string into our function, `decryptQueryString`, and retrieve the decrypted string.

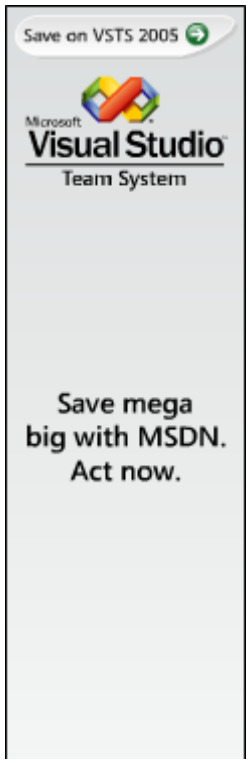
```
string strScramble =  Request.QueryString["search"];
string strdeCrypt = decryptQueryString(
    strScramble.Replace(" ", "+"));
```

Now we've decrypted the value of the QueryString, 'search', and we can do whatever we want with i going to see a URL that looks like:

```
http://yoursite.com?search=da00992Lo39+343dw
```

They'll never be able guess what's going on in your QueryString, and if they try to fool around with it crack the code without knowing the Symmetric key.

VB.NET




```
Imports System
Imports System.IO
Imports System.Xml
Imports System.Text
Imports System.Security.Cryptography

Public Class Encryption64
    Private key() As Byte = {}
    Private IV() As Byte = {&H12, &H34, &H56, &H78, &H90, &HAB, &HCI

    Public Function Decrypt(ByVal stringToDecrypt As String, _
        ByVal sEncryptionKey As String) As String
        Dim inputByteArray(stringToDecrypt.Length) As Byte
         Try
            key = System.Text.Encoding.UTF8.GetBytes(Left(sEncryptic
            Dim des As New DESCryptoServiceProvider()
            inputByteArray = Convert.FromBase64String(stringToDecryp
            Dim ms As New MemoryStream()
            Dim cs As New CryptoStream(ms, des.CreateDecryptor(key,
                CryptoStreamMode.Write)
            cs.Write(inputByteArray, 0, inputByteArray.Length)
            cs.FlushFinalBlock()
            Dim encoding As System.Text.Encoding = System.Text.Encoc
            Return encoding.GetString(ms.ToArray())
        Catch e As Exception
            Return e.Message
        End Try
    End Function

    Public Function Encrypt(ByVal stringToEncrypt As String, _
        ByVal SEncryptionKey As String) As String
        Try
            key = System.Text.Encoding.UTF8.GetBytes(Left(SEncryptic
            Dim des As New DESCryptoServiceProvider()
            Dim inputByteArray() As Byte = Encoding.UTF8.GetBytes( _
                stringToEncrypt)
            Dim ms As New MemoryStream()
            Dim cs As New CryptoStream(ms, des.CreateEncryptor(key,
                CryptoStreamMode.Write)
            cs.Write(inputByteArray, 0, inputByteArray.Length)
            cs.FlushFinalBlock()
            Return Convert.ToBase64String(ms.ToArray())
        Catch e As Exception
            Return e.Message
```

```
            End Try
        End Function

    End Class
```

Generated using

How would you rate the quality of this article?

1 2 3 4 5

Poor ○ ○ ○ ○ ○ Excellent

Tell us why you rated this way (optional):

☑ Send to Author ☑ Post on Site

Submit

Article Rating

The average rating is: No-one else has r

Article rating: **4.03571428571429** ou

28 people have rated thi

Article Score 5515

Comments

Comment #1 (Posted by wan)

hi..

can you give the full source code in C#??

TQ

Comment #2 (Posted by josh manning)

I second that. Could you please provide the rest of the C# code?

Thanks.

Josh

Comment #3 (Posted by tiberius)

I'll work on getting the code converted sometime within the week.
It shouldn't take all that long!

Thanks...

Tiberius

Comment #4 (Posted by William)

Great article! However, I am unclear where your key (!#$a54?3) came from. Is it generated throu
class?

Comment #5 (Posted by tiberius)

RE: We're going to use a hard-to-crack 8 byte key, !#$a54?3, to keep parts of our QueryString s

Actually, I just made the key up. I tried to make it as difficult as a key to crack as I could. You co
key like: abcdefgh or 12345678 but that would be too easy to crack!

Comment #6 (Posted by Josh Manning)

Sorry to bug you about this but do you know if you'll have a chance to add the c# code this week

Comment #7 (Posted by Buck)

Can you translate the C# code to VB.NET? I'm having trouble converting it over. Thanks

Comment #8 (Posted by Jason)

How about everyone that needs to do code conversion go buy the C# to VB.NET code conversi
published by O'Reilly.

Before that though I would reccomend learning C# if you're going to be doing much .NET Develo
guru by any means but I could translate the VB.NET code in this article in my head on the fly.

Jason

Comment #9 (Posted by John Mandia)

Hi all,

Just tried converting it for those of you who want it. Not too familiar with VB.Net so there are sor
hopefully someone can post the answers to as I have a load of things to do and don't have any s
these down. Might have time over the weekend. These have been commented.

```
using System;
using System.IO;
using System.Xml;
using System.Text;
using System.Security.Cryptography;

namespace TotalIngenuity.ManipulateData
{
///
/// This class is used for encrypting and decrypting data such as querystrings.
///

public class DataEncryption
{
public DataEncryption()
{
}

private byte [] key = {};
private byte [] IV = {&H12,&H34,&H56,&H78,&H90,&HAB,&HCD,&HEF}; // This throws an error.
// get these values in c# so if anyone can
// help it will be helping everyone out


public string Decrypt(string stringToDecrypt, string sEncryptionKey)
{
byte [] inputByteArray = new byte[stringToDecrypt.Length];

try
{

key = System.Text.Encoding.UTF8.GetBytes(Left(sEncryptionKey, 8)); // Left is VB.NET Specifi
// Would Trim achieve the same?
DESCryptoServiceProvider des = new DESCryptoServiceProvider();
inputByteArray = Convert.FromBase64String(stringToDecrypt);
MemoryStream ms = new MemoryStream();
```

```
CryptoStream cs = new CryptoStream(ms, des.CreateDecryptor(key, IV), CryptoStreamMode.W
cs.Write(inputByteArray, 0, inputByteArray.Length);
cs.FlushFinalBlock();

System.Text.Encoding encoding; // not too sure if this will work

return encoding.GetString(ms.ToArray());

}
catch(Exception ex)
{
throw ex;
}

}

public string Encrypt(string stringToEncrypt, string SEncryptionKey)
{
try
{
key = System.Text.Encoding.UTF8.GetBytes(Left(SEncryptionKey, 8)); // Left is VB.NET Specifi
// Trim achieve the same?
DESCryptoServiceProvider des = new DESCryptoServiceProvider();
byte [] inputByteArray = Encoding.UTF8.GetBytes(stringToEncrypt);
MemoryStream ms = new MemoryStream();
CryptoStream cs = new CryptoStream(ms, des.CreateDecryptor(key, IV), CryptoStreamMode.W

cs.Write(inputByteArray, 0, inputByteArray.Length);
cs.FlushFinalBlock();

return Convert.ToBase64String(ms.ToArray());

}
catch(Exception ex)
{
throw ex;
}
}

}

}
```

Comment #10 (Posted by Matias Pelenur)

Encryption does not equal security. In the example you gave where you pass around a custome start), encrypting it the way you describe with a symmetric key provides almost no improvement can still get the query string and re-send it as it, encrypted, and still gain access to the same cus the server-side just descrypts it and gets the same content). You could argue that they wouldn't customer id that was being passed around (if you encrypt the whole thing), but then that's secur as you point out that doesn't really work in the long term.

It's always best to use sessions, but that can also be eavesdropped. The only really secure way Otherwise, you could keep track of the users's IP address in the session object. If someone tries from a different IP, you deny the request.

My 2 cents,
Matias

Comment #11 (Posted by tiberius)

You make some great points, Matias. HTTPS is the way to go in the long run... unfortunately, sc afford an https certificate. I agree that session variables are nice, but aren't always the way to g you're working in a web farm environment...

The fact is that if your site is passing around user information, some security is better than no se

(SSL) is totally the way to go when point-to-point secure communication is needed - but in the m
more difficult for the casual 'hacker' to gain access is key.

You've brought up some excellent points.

Comment #12 (Posted by Jason)

Web Farms are no longer an excuse for not using Session variables. Specify a session server.

Comment #13 (Posted by an unknown user)

Matias Pelenur

Comment #14 (Posted by Joe Feser)

Matias Pelenur : Quote
"It's always best to use sessions, but that can also be eavesdropped. The only really secure way
Otherwise, you could keep track of the users's IP address in the session object. If someone tries
from a different IP, you deny the request. "

You are aware that every request from AOL comes from a different IP address.

AOL is not the only one, the only time the ip address works is for a local intranet app.

Even users from my own house would not work since all the computers go thru a router, so ever
has the same IP.

This solution would never work.

Joe

Comment #15 (Posted by pcbear)

can you give the full source code in C#??

Comment #16 (Posted by augusten)

I've been planning to use the great crypto built in to .NET to encrypt my querystrings for a while
behold I find your article. Saves me a lot of time, thanks!

Regarding Matias comments: In the real world, SSL and sessions are a poor solution in many ca
are costly in terms of performance. I am responsible for an app that tens of thousands need to a
simultaneously using as little bandwidth and processing power as possible. Querystrings are the

Sure, if you are passing highly sensitive data (i.e. credit card numbers) you use SSL (duh). But t
the time, it's name, street address, email address, and the like, data which is semi-public to begi
want the public using your website to pull up said data. In these cases it's not the interception of
querystring that needs to be prevented, but rather the ability to pull up any customer's data by cl
querystring id.

Comment #17 (Posted by Turino)

I just tried to finallize the C# code from Jhon Mandia, I've tested it and it works...

Thanks to John Mandia...

```
using System.Text;
using System.Security.Cryptography;


public class Encryption64
{
//private byte[] key = {};
```

```
//private byte[] IV = {10, 20, 30, 40, 50, 60, 70, 80}; // it can be any byte value

public static string Decrypt( string stringToDecrypt,
string sEncryptionKey)
{

byte[] key = {};
byte[] IV = {10, 20, 30, 40, 50, 60, 70, 80};
byte[] inputByteArray = new byte[stringToDecrypt.Length];

try
{
key = Encoding.UTF8.GetBytes(sEncryptionKey.Substring(0,8));
DESCryptoServiceProvider des = new DESCryptoServiceProvider();
inputByteArray = Convert.FromBase64String(stringToDecrypt);

MemoryStream ms = new MemoryStream();
CryptoStream cs = new CryptoStream(ms, des.CreateDecryptor(key, IV), CryptoStreamMode.W
cs.Write(inputByteArray, 0, inputByteArray.Length);
cs.FlushFinalBlock();

Encoding encoding = Encoding.UTF8 ;
return encoding.GetString(ms.ToArray());
}
catch (System.Exception ex)
{
throw ex;
}
}

public static string Encrypt( string stringToEncrypt,
string sEncryptionKey)
{

byte[] key = {};
byte[] IV = {10, 20, 30, 40, 50, 60, 70, 80};
byte[] inputByteArray; //Convert.ToByte(stringToEncrypt.Length)

try
{
key = Encoding.UTF8.GetBytes(sEncryptionKey.Substring(0,8));
DESCryptoServiceProvider des = new DESCryptoServiceProvider();
inputByteArray = Encoding.UTF8.GetBytes(stringToEncrypt);
MemoryStream ms = new MemoryStream();
CryptoStream cs = new CryptoStream(ms, des.CreateEncryptor(key, IV), CryptoStreamMode.W
cs.Write(inputByteArray, 0, inputByteArray.Length);
cs.FlushFinalBlock();

return Convert.ToBase64String(ms.ToArray());
}
catch (System.Exception ex)
{
throw ex;
}
}
}
```

Comment #18 (Posted by John Mandia)

Thanks for tidying up my code....with all the things going on I forgot to finish the c# code off. Tha

John

Comment #19 (Posted by John Mandia)

Just read through this article again and thought wouldn't it make implementation easier if you ad
functionality with the Decrypt method? This would mean that people using this class would not n
always using replace throught their site.

just a thought.

What are people's opinions?

John

Comment #20 (Posted by Tiberius)

John:

Excellent point. Adding the replace in the Decrypt method makes much more sense.

Tiberius

Comment #21 (Posted by John Mandia)

Hi Tiberius,

Just sent you a mail asking if you would mind me writing a quick article based on this one.

Taken the c# version of the code that I contributed and extended it.

Basically the methods can be overloaded for 1 so that users have the option of either supplying
or getting it from the Web.config file (as it makes more sense to me....you can change the settin
from one place) another additional feature is to take into account oddities (Like the fact that the
replaces + with a space) I have added a method to the class an additional parameter to the Dec
you can handle the oddities. e.g Decrypt(string stringToDecrypt, int forType)

forType represents what type of code is calling this decryption e.g 0 default 1 Querystring.
forType and stringToDecrypt get sent to a private method that does a switch based on forType.
sending 1 it knows a querystring wants this so it replaces " " with "+" and then sends it back.

This way as more quirks appear you can handle it within your private method and the users of th
have to worry about it. And you could also add addition forTypes as they arise (maybe down the
Request.Form does something for example...this could become forType =2).

Now only two things are bugging me, (1) should I make all the methods static and (2) Everything
trying to fix an error.

As my intention is to share this code with everyone I was wondering if you would like to have a l
am tripping up.

Thanks,

John

Comment #22 (Posted by an unknown user)

O.K,

Got the class working now with the features mentioned previously and it's CLS compliant so you
VB.NET or any other .NET language.

I'll be posting the code up soon if anyone is interested.

John

Comment #23 (Posted by test)

test

Comment #24 (Posted by Simon H)

Hi John,

Not sure if you still look at this posting any more as it is a little old now! If you are still around did new code? Just I would be interested in taking a look.

Cheers

Simon

Comment #25 (Posted by Kelly)

Hi, I wonder if you've come accross this when using the code. I'm using asp.net
Example:
when passing in the URL "....aspx?MemberRef =" 117 or 114 or 260 or 264 where number is en calling
txtMemberRef.Text = Sec.Decrypt(MemberRef, ConfigurationSettings.AppSettings("Key"))
It works perfectly..
But when I try to use member ref 262 or 259 I get
"Invalid length for a Base-64 char array." when trying to decrypt.
Any help greatly appreciated!
Thanks

Comment #26 (Posted by urfie)

I'm a newbie to encryption, so i kinda need some help understanding this. I am unclear as to how the flow of information goes. is there anyway i can get an explanation or a diagram?

for example, if the client wants to send some data to the server, that client needs to encrypt it before sending it. so i'm guessing the server needs to send the key to the client so that the client will know how to encrypt?

am i understanding that correctly? is there a danger that someone will be able to eavesdrop when the server is sending the key, then, when the client sends the info back, the eavesdropper can use the key to decrypt?

Comment #27 (Posted by Suneel)

I believe the VB.Net version of this code is at: http://www.eggheadcafe.com/articles/20020315.a

There is a comment in the code regarding the definition of IV bytes, and the code shows the VB of Hexadecimal numbers (&H prefix). This should be changed to 0x in C#.

Comment #28 (Posted by parshu)

hi Tiberius ,

gr8 article. Solved my problem within minutes. Thanx.

Comment #29 (Posted by Senthil Nathan)

I am in C#. How to use the c# code for Encrypting Querystrings. I have tried the two methods bu

"The type or namespace ExtractAndSerialize could not be found
The type or namespace oES could not be found"

here, i have used the namespaces
using System.Text;

using System.Security.Cryptography;
using System.IO;
using System.Xml;
anyone help me out....

Comment #30 (Posted by angry)

Am I a fool? You write an article and don't offer the source code?????????????????

Comment #31 (Posted by Petrovsky)

Uh, You are a fool. The SC is in the article.

Comment #32 (Posted by an unknown user)
Rating
code doesn't work like it should. Plus, he left out the c# code, then says it will be posted later, bu
years have gone by)all I wanna know is how to replace the stinkin "Plus" signs.

Comment #33 (Posted by an unknown user)
Rating
outdated code:"Invalid length for a Base-64 char array." when trying to decrypt.

Comment #34 (Posted by BG)
Rating
When I try to decrypt a GUID Value it gives me an error "Invalid length for a Base-64 char array.
encrypt and decrypt the following string"A1C1AEB2-3C75-43D9-9F97-46EE1E0038D9"I am usi
code.Any help will be greatly appreciated

Comment #35 (Posted by an unknown user)
Rating
using System; using System.IO; using System.Xml; using System.Text; using System.Security.C
class Encryption64 { private byte[] key = {}; private byte[] IV = {18, 52, 86, 120, 144, 171, 205, 2
Decrypt(string stringToDecrypt, string sEncryptionKey) { byte[stringToDecrypt.Length] inputByte
System.Text.Encoding.UTF8.GetBytes(Left(sEncryptionKey, 8)); DESCryptoServiceProvider de
DESCryptoServiceProvider(); inputByteArray = Convert.FromBase64String(stringToDecrypt); M
new MemoryStream(); CryptoStream cs = new CryptoStream(ms, des.CreateDecryptor(key, IV)
CryptoStreamMode.Write); cs.Write(inputByteArray, 0, inputByteArray.Length); cs.FlushFinalBlo
System.Text.Encoding encoding = System.Text.Encoding.UTF8; return encoding.GetString(ms.
(Exception e) { return e.Message; } } public string Encrypt(string stringToEncrypt, string SEncryp
= System.Text.Encoding.UTF8.GetBytes(Left(SEncryptionKey, 8)); DESCryptoServiceProvider
DESCryptoServiceProvider(); byte[] inputByteArray = Encoding.UTF8.GetBytes(stringToEncrypt
ms = new MemoryStream(); CryptoStream cs = new CryptoStream(ms, des.CreateEncryptor(ke
CryptoStreamMode.Write); cs.Write(inputByteArray, 0, inputByteArray.Length); cs.FlushFinalBlo
Convert.ToBase64String(ms.ToArray()); } catch (Exception e) { return e.Message; } } }

Comment #36 (Posted by Dietmar)
Rating
Very helpful article and code. What would I need to change to have it output only alphanumerics
string? I need to encrypt a few values after another to be used in a URL and only alphanumerics
cleaner. Thanks!

Comment #37 (Posted by Michiel Erasmus)
Rating
Anser to Comment #25; I have had the same problem but solved it by using the same encryptio
to include spaces. Ek hoop dat dit vir jou iets waardevols is. met vriendelike groet, Michiel Erasr

Comment #38 (Posted by an unknown user)
Rating
Doesn't work very well, will not work on some values, so whats the point if values are going to b

Comment #39 (Posted by Paul Talbot)
Rating
For all those who are having trouble converting VB.Net to C#, take a look at Reflector by Lutz R
http://www.aisto.com/roeder/dotnet/ This will help you figure out the VB.Net > C# > Delphi > IL c
article btw, I will be making use of this.

Comment #40 (Posted by an unknown user)
Rating
Good information, but you've failed to be helpful enough to mention where these class(es) exist

Comment #41 (Posted by an unknown user)
Rating
Good information, but you've failed to be helpful enough to mention where these class(es) exist

Comment #42 (Posted by John)
Rating
Does not work when Encrypted value has double forward slashes, because .NET Request autor
as Single Slash, and correspondind Decryption fails. Please let me know if you fix this bug. Thar

Comment #43 (Posted by an unknown user)
Rating
Thankyou very much Tiberius - really solves ASP DOTNET to old ASP session state issue. Now
out how to go the other way using the same kind of thing

Comment #44 (Posted by an unknown user)
Rating
Comment (Posted by Ankit) Excellent article Tiberius - Really thanks for solving my problem. Ca
the explanation of each line you coded, So that we can actully underdstand the magic of your cc

Comment #45 (Posted by an unknown user)
Rating
Comment (Posted by Ankit) Excellent article Tiberius - Really thanks for solving my problem. Ca
the explanation of each line you coded, So that we can actully underdstand the magic of your cc

Submit Comment

Sponsored Links

PrettyCode.Print **Finally, a professional Visual Basic printout.**
That's PrettyCode.Print - http://www.prettycode.com

NeoTekSystems
A Total Solutions Provider
To Find Out How We Can Help You Click Her

Copyright 1998-2005 vbCity.com, LLC. All rights reserved. Powered by ArticleLive Article Management Software