

Re: COM Interface Security

Source: <http://www.tech-archive.net/Archive/VB/microsoft.public.vb.winapi/2005-01/0317.html>

From: Joseph Geretz (jgeretz_at_nospam.com)

Date: 01/20/05

Date: Wed, 19 Jan 2005 22:37:44 -0500

Hi Ulrich.

> *For passing keys around, I agree with you that not the key should be the
> secret thing but the en/decryption algorithm used to en/decrypt using this
> key.*

Let me clarify. The algorithm I spoke of is a way to protect the secrecy of the key. However, the security of the ultimate encryption does not depend on secrecy of the encryption algorithm, but rather on the secrecy of the key. All industrial strength encryption algorithms are publicly disclosed algorithms. In fact, the integrity of these encryption algorithms are only tested through peer review. Thus the truly secure algorithms are those which are publicly disclosed and can be proven to be secure through objective, mathematic proofs.

So I for one, have no interest in keeping our encryption algorithm secret. We're using Blowfish (which is the only algorithm for which I could find a VB implementation). But the key? Ah, now that's a different story...

(BTW, years ago I developed my own proprietary 'encryption' algorithm which was used successfully in an internal software application. Subsequently I posted a ciphertext snippet to one of the security newsgroups. It was cracked in about 20 minutes. It was then that I learned that encryption algorithms need to be **strong**, rather than secret. It's the key which needs to be kept secret.)

– Joe Geretz –

"Ulrich Korndoerfer" <ulrich_wants_nospam@prosource.de> wrote in message news:41EF1404.40808@prosource.de...

> *Hi,*

>

> *Joseph Geretz wrote:*

>

>> *Where it is undesirable to allow other clients to use our public classes,*

>> *then we'll need to implement a bi-directional handshake. That is our own*

>> *software instantiates the class, passes in a 'public key' and then*

>> *expects the class to return an encrypted value which matches its own*

>> *internal private key. This would allow us to develop our software in a modular manner but would prevent other developers from making use of our own proprietary classes.*

>>

>> *How does that sound? Anyone done anything similar?*

>

> *There are several other ways too to prevent unauthorized users to use a dll.*

>

> *For passing keys around, I agree with you that not the key should be the secret thing but the en/decryption algorithm used to en/decrypt using this key. This algorithms has to be implemented in both the client and the server. Authentication then could go as following:*

>

> *– The client passes a random key (random in content and length) along with a random text to the server. The server encrypts the text with the given key and returns the encrypted text. The client decrypts the text and compares it with the original. If identical, the server is authenticated (at least one can say now that the server knows the encryption algorithm ;–).*

>

> *– Then the client calls the server for a random key and random text. The client encrypts the text and passes it to the server. The server decrypts the text and compares it with the original. If identical, the client is authenticated (again at least one can say now that the client knows the encryption algorithm ;–).*

>

> *The encryption algo could encrypt text to a text which has random length, which makes cracking the algorithm a little bit harder. For example it could include a date/time stamp in the key or the text which decides how long the encrypted text will be. Or use a checksum of the random key or the random text to decide how long the encrypted text will be.*

>

> *ARC4 would be an algorithm well suited. He can both provide the random keys and texts as well as doing the encryption/decryption. Mock it up with varying prefix runs and use some wild calculations to extract the real key embedded in the truly random key eg from information found in the random key. Do not use the random key as the real key in direct. At least do some transformation on it before using it as key for encryption. You eg. could permute all random key bytes to create the real key. For the permutation eg. use a 64 bit value (extract it to a variable of type decimal for calculating the permutation) which defines the permutation to be done (should be sufficient for 20 bytes or so). The permutation derived from a number is quickly done. The number eg. could be simply the first 8 bytes of the random key.*

>

> *The more you scramble the random key to yield the real key the safer you are.*

>

> --

> *Ulrich Korndoerfer*

microsoft.public.vb.winapi: Re: COM Interface Security

>

> *VB tips, helpers, solutions* -> <http://www.proSource.de/Downloads/>