The Old New Thing

Why are the Compression and Encryption options check boxes instead of radio buttons?

13 Dec 2010 7:00 AM | 55

Tanveer Badar asks why the file properties Advanced dialog shows two checkboxes (compression and encryption) even though NTFS supports only one or the other at a time. "Why not have two radio buttons instead of these silly check boxes?"

Actually, if you want radio buttons, you'd need three, one to cover the "neither" case. Let's look at what those radio buttons would look like:

- ○ Compress contents to save disk space
- ○ Encrypt contents to secure data
- ○ Neither compress nor encrypt

This takes an implementation detail (that NTFS currently does not support simultaneous compression and encryption) and elevates it to the user interface, so that it can provide maximum confusion to the user. "What a strange way of exposing compression and encryption. If I want to turn on encryption, I should just check a box that says 'Encryption'."

The current implementation (two check boxes) matches user expectations, but then says "Sorry, we can't do that" if the user picks a combination that is not currently supported. But who knows, maybe someday, NTFS will support simultaneously encryption and compression, and all that will happen is that the error message goes away. You don't have to redesign the property sheet (and invalidate all the training materials that had been produced in the meantime).

Either way, it's a leaky abstraction that sucks, but at least the suckiness isn't shoved in the user's face.

Blog - Comment List MSDN TechNet

Comments

The MAZZTer
13 Dec 2010 7:10 AM
#

Compression + Encryption is icky. If you compress first you're good to go, but if you want to compress encrypted data, well, it's not going to compress very well, so you need to unencrypt it first, then you can compress it best, then reencrypt the compressed data. Fun.

And I imagine file access is a bit slower when using both at once rather than using one or the other.

The real radio buttons are a nice touch, btw.

acq
13 Dec 2010 7:27 AM
#

> file access is a bit slower when using both at once rather than using one or the other.

If properly done you encrypt less if the data are compressed and compression can be implemented much faster, so decent encryption with fast compression is typically faster than just decent encryption.

Jeff Tyrrill
13 Dec 2010 7:29 AM
#

I doubt both encryption and compression will ever be supported, because compressing the file before encrypting reveals (a little bit of) information about the contents of the file, because from the difference in disk space between the compressed size and the logical size, one can infer some information about the type of data. This goes against the goal of encryption.

(This doesn't change the validity of the explanation in the post.)

John Topley
13 Dec 2010 8:01 AM
#

Compression generally relies on the presence of repeated sequences of bits, which are exactly the sort of thing that make it possible to break an encryption scheme.

Jeff
13 Dec 2010 8:02 AM
#

I think that would only be possible if disk access was so fast that it wouldn't be an issue when you make it 9x slower.

GWO
13 Dec 2010 8:11 AM
#

@JohnTopley: Compression generally relies on the presence of repeated sequences of bits, which are exactly the sort of thing that make it possible to break an encryption scheme.

What is that supposed to mean? That encrypting compressed files is more secure than encrypting uncompressed files? [Wrong, unless your encryption scheme is completely idiotic]. That encrypting compressed files is less secure than encrypting uncompressed

files? [Also wrong]. That encrypted files don't compress well [true, but not what you wrote].

Joshua Ganes
13 Dec 2010 8:15 AM
#

Raymond says, "at least the suckiness isn't shoved in the user's face." I agree that situation is not ideal. However, I would rather see the options available to me and decide than to be bitten by them after I've already made a decision.

Joshua Ganes
13 Dec 2010 8:15 AM
#

Raymond says, "at least the suckiness isn't shoved in the user's face." I agree that situation is not ideal. However, I would rather see the options available to me and decide than to be bitten by them after I've already made a decision.

Joshua
13 Dec 2010 8:22 AM
#

Who's to say that some other filesystem in driver doesn't already provide both?

Pierre B.
13 Dec 2010 8:28 AM
#

I fail to see how a surprise "no can't do" with a UI that implicitely suggest both should be usable at the same time improves the interaction. You pretty much load the question by framing in a particular way and setting up a strawman. Radio button implies just as much turn thing on and off as much as check-box, and make it explicit that the options are exclusive. The only confusing bit the the strawman of supplying an example UI that is verbose and confusing. A simple UI would do:

Data format:

* Normal

* Encrypted

* Compressed

Random User 423686

13 Dec 2010 8:28 AM
#

The main problem that tends to turn up with (blindly) encrypting compressed data is that many compression schemes will cause predictable data to appear in predictable places. ("Magic numbers", for example.) This can open the door for partial known plain-text attacks. Note that there are ways to compensate, particularly if the algorithms are designed to work together.

HiTechHiTouch
13 Dec 2010 8:32 AM
#

@GWO John is pointing out that perfectly compressed data is close to random, because all patterns have been removed by the compression scheme. If this compressed data is then encrypted, 1) The intercept given to the de-cryptographers is shorter (therefore harder to decrypt), and 2) does not contain some clues (repeating patterns) often used by de-cryptogolists.

Sky the Madboy
13 Dec 2010 8:37 AM
#

@JoshuaGanes: As I've seen written recently - when designing the UI, you need to think like a basic user, and you are not a basic user. You are an administrator. They could always design two different UIs for each class of user (which they've kind of done in Windows 7), but it doesn't strike me as a terribly efficient use of resources.

Personally I think the current setup is the best compromise for all users. A basic user will be less confused by the current design. An administrator may grouse about things not being set up exactly how they want it to be, but at least they can still use it.

GWO
13 Dec 2010 8:41 AM
#

[does not contain some clues (repeating patterns) often used by de-cryptogolists.]

Relying on repeating patterns as cribs for decryption is a 1940s technique. It worked on the Enigma machine, but its going to be completely, completely useless against any sane encryption scheme invented since about 1970. It's certainly not going to help you crack the encryption on an NTFS volume. This is not your grandfather's substitution cipher, or anything even remotely like it.

HiTechHiTouch
13 Dec 2010 8:41 AM
#

@Random True, most compression schemes are predictable at the beginning, which would be a clue. However, I'd not expect too much help from the compression dictionary (if it can be hidden) because there will only be one copy of a 'repeating' pattern -- multiple copies are compressed out.

On the other hand, as you point out, this can be mitigated when encryption knows about the compression. But it can't be completely eliminated because there has to be a starting point to begin decompressing the data.

K
13 Dec 2010 8:42 AM
#

Pierre B. certainly makes a very valid point. When you use good wording (and none of that "neither compress nor encrypt" strawman), the radio button suddenly looks very good. I would prefer that by a mile.

HiTechHiTouch
13 Dec 2010 8:50 AM
#

@GWO You're right about modern encryption. But please consider the problem of recognizing that the text has been successfully decrypted. It's not like you get back a bunch of words from some language you can verify out of a (traditional type) dictionary. The more random the true text appears, the harder it will be to recognize the true text as such, should decryption be successful.

Mark
13 Dec 2010 9:21 AM
#

GWO: not true. Repeating strings, if they span blocks, are useful for modern cryptanalysis. Also bear in mind that many files include long runs of zeroes, giving an attacker plaintext as well.

Compression and encryption are used without problem by mcrypt and SSH, fwiw.

Prerequisites for speculation in the comments: understanding of a few modern crypto algorithms and implementations.

Martins Mozeiko
13 Dec 2010 9:29 AM
#

i38.tinypic.com/sghf9e.gif

How about checkboxes in DXSDK installation?

Why do MS uses checkboxes there instead of radio buttons?

Screenshot is from November 2008, but latest SDK from 2010 also has same UI.

Dave
13 Dec 2010 9:35 AM
#

Agree with Pierre B. Also a UI design can't be justified on the grounds that it could one day be a valid representation of the available choices (however unlikely that may be).

RoboRock
13 Dec 2010 9:38 AM
#

For the first time I find myself really dis-agreeing with Raymond. The user is gonna say, "Well if I can't do that then why did you let me." Your logic could apply to any radio buttons - I mean, sorry you might have to change the interface in the future...I don't think that cost savings justifies this lazy design.

RoboRock
13 Dec 2010 9:40 AM
#

...and as a followup, why would the user even bother trying in the future...you just trained them to think it won't work.

Nick
13 Dec 2010 9:48 AM
#

@Martins Mozeiko, checking both boxes specifies you only want to participate sometimes.

Hidden for a reason
13 Dec 2010 9:49 AM
#

Seems that a lot of people are making wrong assumptions about encryption. During my military service (somewhere in Europe) I was working on an encrypted radio system. We used to transmit the same pattern ("quick brown fox" like) for days at time. I always asked the crypto expert if that was not helping the "enemy" find the characteristics (and eventually keys) used by our system and their answer was always unambiguously "no". Now that I understand crypto a little bit better I tend to strongly agree with them.

dave
13 Dec 2010 9:53 AM
#

I agree with Raymond here. Having a UI 'know' that a particular file system cannot do feature A and feature B at the same time is empirical computer science, rarely a good thing. The file system in this case says (via its volume attributes) 'I can do encryption' and 'I can do compression'. The attributes cannot express 'but not together'.

Therefore, application code (such as Explorer) has no way to know this.

This is of course a programmer's perspective, which holds that single-point-of-truth and avoiding-undue-coupling ultimately result in a more satisfactory user experience than wiring ad-hoc knowledge into source code. And it's also a file-system guy's perspective, which holds that Explorer's nothing special, it's just an app.

John
13 Dec 2010 9:56 AM
#

Just for reference, on XP the two check boxes behave exactly like radio buttons and there is no error message; this is probably the most confusing behavior possible.

Mike Caron
13 Dec 2010 9:57 AM
#

@Martins: Ha ha, yes, I was going to mention that as soon as I saw this blog post! What a silly design. Obviously, the guy responsible for that installer doesn't read this blog :)

pete.d
13 Dec 2010 9:59 AM
#

Joshua and Pierre have it exactly right: it is _much_ better from a UI design point of view to not allow the user to make incorrect choices in the first place, than to allow them to make an incorrect choice and then scold them for it.

In fact, that's one of the main reasons radio button groups are so useful: it allows the software to present the user with mutually exclusive choices, rather than requiring the user to suss out what a valid combination of choices might be.

If there are problems with the exact wording of the radio buttons or the group label, then the answer is to fix the wording. Not resort to a more user-hostile UI by switching to checkboxes.

pete.d
13 Dec 2010 10:06 AM
#

@dave: "Having a UI 'know' that a particular file system cannot do feature A and feature B at the same time is empirical computer science, rarely a good thing."

Au contraire. That's the whole point of having a UI: to know the rules of the software features it's allowing the user to use and act as a guide to the user to navigate those rules. If and when those rules change, then the UI may be changed. But the UI is supposed to be the mediator between the functionality and the user, in a way that provides maximum understanding, minimal confusion, and most efficient use to the user.

The "checkboxes as radio buttons" UI fails in all three metrics.

Maurits [MSFT]
13 Dec 2010 10:08 AM
#

> You don't have to redesign the property sheet (and invalidate all the training materials that had been produced in the meantime).

boggle

Um, if your feature changes, you should update the UI to match. This is, in fact, one of the easiest ways to let your users know that you've added a feature.

Joseph Koss
13 Dec 2010 11:52 AM
#

There seems to be some misconceptions about modern compression schemes here.

Modern compression doesnt store dictionaries in the compressed stream. They build models "on-line" as the data is being compressed or decompressed, and the main reason they do so is that an ideal compressor adapts to changes in its input stream. In the case of English text, for example, paragraphs near each other will use some of the same words many times even though the words themselves are rather rare elsewhere in the text. It has been discovered that "on-line" methods are a very good approximation of a theoretically ideal compressor, so these days very little research is put into storing pre-evaluated information about the compressed stream because nobody is doing that except for legacy compatability (such as ZIP's use of frequency tables both in-header and in-stream)

The final word on entropy compression is Arithmetic Encoding. It is provably an "ideal" method of storing entropy (there are implementation details that amount to only small differences) so all the research done today, at edge of discovery, is on the modeling of the data and not on how to encode it. The upshot of this is that the individual bits of the decompressed stream are spread out over many bits of the compressed stream, and not in a predictable manner, because all new compression schemes use an Arithmetic Encoder unless there is some constraint that prevents it (such as in H.264, which for some things uses different entropy encoding schemes so that hardware can do a lot in parallel)

Combine these two facts and many of the arguments about compression weakening the encryption are clearly wrong. The argument about compressed size being useful information is correct, bu the file extension normally gives the type of data away anyways...

f0dder
13 Dec 2010 12:09 PM
#

I agree that a radio buttons or a combobox would be less confusing for users, but I agree with the current design from a programmer standpoint: as mentioned previously, the information you get about the filesystem tells you whether compression and encryption are available, but it can't represent "but they're mutually exclusive".

So what would a solution be? You can't just hardcode the setting entirely, since there might be filesystems around already that support both. Would you then hardcode logic that does a string-compare to check if it's NTFS and then does the assumption that it's either-or? (While butt-ugly, that might work for explorer.exe since it's shipped with the OS. But it's no-go for 3rd party software. I'm sure some is already doing that, though).

Or do you have some other suggestion for detecting whether both are selected, in order to adapt the user interface? Like trying to create a dummy compressed-and-encrypted file in the folder(s) of the file(s) you are modifying the properties for? Please keep in mind that explorer can apply the settings recursively, and that there's symlinks and NTFS junctions to take into consideration.

Random User 423686
13 Dec 2010 12:57 PM
#

I included the "blindly" qualifier for a reason. The problems are greatly reduced or eliminated as you know more about the algorithms in question. A sample "uninformed" scenario would be ZIP compression with some kind of substitution or simple XOR encryption.

Brian G.
13 Dec 2010 2:17 PM
#

I've never had a problem with the current implementation, but if MS were to hire me as their UI designer, I might suggest something like:

How would you like to store the files on this volume?

Default (Best Performance)

Encrypted (Security)

Compressed (Increased Storage Space)

Gabe
13 Dec 2010 2:35 PM
#

Brian G.: Are you under the impression that non-compressed gives the best performance? If so, I'd love to see the benchmarks that prove it. Compressing and decompressing data usually takes much less time than sending it over the I/O interface, so I wouldn't expect that leaving the data on the disk uncompressed would give the best performance.

roastbeef
13 Dec 2010 2:36 PM
#

@Hidden for a reason:

You were constantly transmitting your "quick brown fox" for another reason: preventing traffic analysis. If an enemy can't decode your transmissions, but can plot the volume of transmissions over time, they can infer changes in activity of your organization. But if you keep your pipe shuttling bits back and forth constantly, then they've got no idea how much actual communication you're actually doing.

Jules
13 Dec 2010 2:56 PM
#

@Gabe

directedge.us/.../to-compress-or-not-to-compress-part-ii

Note that's on a flash drive; performance on a hard disk is likely to be worse, as using compression (apparently) increases fragmentation which is more problematic on hard disks compared to flash.

Ben Cooke
13 Dec 2010 4:26 PM
#

The crux of the problem is in Raymond's closing sentence: it's a leaky abstraction.

Specifically, the design of the interface between Explorer and the filesystem driver is flawed. Assuming other commenters above are correct about the interface, it provides something like the following:

* filesystem.supports_encryption()

* filesystem.supports_compression()

In other words, it assumes that encryption and compression are supported either for all files on disk at all times or for no files on disk at any time.

Given the characteristics of NTFS, this interface should really be more like:

* filesystem.supports_encryption_for_file(file)

* filesystem.supports_compression_for_file(file)

It must of course be documented that changing the compression state of a file might impact whether encryption is supported and vice-versa, so Explorer would then need to re-query these flags after each change to update the state of the UI.

This is of course susceptible to race conditions, but that's true of most UI that changes some state: if I open the dialog box and then someone turns on encryption before I try to turn on compression, my operation will fail despite the UI making it look like it should've worked. This is sub-optimal, but it seems like enough of a corner-case that a suitably helpful dialog box would suffice here.

I think this scenario is an example of one layer having to do the best it can with a poorly-designed lower layer. The lower layer is unable to provide the guarantee it is documented to provide (that it can report encryption and compression as two independent, filesystem-global flags). This is a common problem and is the very definition of a leaky abstraction, and can only really be solved properly by adjusting the interface provided by the lower layer to more accurately reflect the capabilities of that layer.

f0dder
13 Dec 2010 6:29 PM
#

@Ben Cooke: your supports_x_for_file() idea wouldn't really work, though - the encryption/compression isn't performed as soon as you check the checkbox (that would be pretty nasty if you were viewing properties for a deeply nested folder and accidentally fat-fingered the compression checkbox).

Joshua
13 Dec 2010 6:50 PM
#

Just don't try compressing database files. It might be faster to read certain kinds of files serially. Both EXEs (which are memory-mapped) and databases don't behave half as well.

Troll
13 Dec 2010 8:53 PM
#

Well the UI is designed in such a way that I "got it" from using it that both are not supported on the same file and that the third case is important as well. So there's nothing wrong with the UI, it's perfect. The only thing I miss is a context menu item for "Compress" like there is for "Encrypt" (which can added using TweakUI) which also asks me later "Compress the file only or Compress the folder" with a checkbox for "Always compress only the file". :) Btw, the compression algorithm is never right for backward compatibility?

Troll
13 Dec 2010 8:54 PM
#

"is never *updated*, right?" I meant

Thomas Bouton
13 Dec 2010 10:52 PM
#

First : you have to update documentation and training material. Good doc says that both can't be performed at the same time. So your point is quite moot.

Second : in crypto, ANY information on the plain text is a help. It limits your problem space. How much of a help spends on your crypt algorithm but still...

640k
14 Dec 2010 2:35 AM
#

@Pierre B: You pretty much load the question by framing in a particular way and setting up a strawman.

As Raymond always does.

The radio button options should be:

* Normal

* Encrypted

* Compressed

* Encrypted & Compressed (grayed out on ntfs)

Popup errors are evil.

I promise windows UI will be redesigned 10 times before ntfs supports both encryption and compression!

> [*Graying-out would be misleading in this case*. -*Raymond*]

asdbsd
14 Dec 2010 2:58 AM
#

My five cents: it should have been either radioboxes or checkboxes but which you can set both.

If the underlying FS doesn't support compression+encryption, it just silently uses only encryption. After all, there's no saying how good exactly the data should be compressed. It's just "try your best" flag.

Leo Davidson
14 Dec 2010 4:38 AM
#

@dave: "Having a UI 'know' that a particular file system cannot do feature A and feature B at the same time is empirical computer science, rarely a good thing."

But the UI *does* know.

If you try to select both, the UI clears one of the checkboxes. Checkboxes don't clear themselves so clearly the UI knows the two options are mutually exclusive.

People are not arguing about whether or not the UI should know about the filesystem features; only about how it presents the options to the user.

Personally, I can see where other people are coming from but I don't have a problem with the current UI. Generally, checkboxes behaving like radio buttons can be confusing/annoying, but I don't mind it when there are only two of them and they are right next to each other.

GWO
14 Dec 2010 8:21 AM
#

@ThomasBouton: "How much of a help spends on your crypt algorithm but still..."

If your crypt algorithm is "anything modern", the amount it helps you is "a completely negligibly small amount". It would certainly help you brute force it, but by nothing like enough to make brute force worthwhile.

Danny
14 Dec 2010 9:39 AM
#

I believe that Micro$oft will just wait for quantum encryption to be available for technology and they will implement compression and encryption, till then the current GUI is just fine. I doubt that there are many users that use any of those features anyway, since HDD's are cheap therefore you don't need compression and relying on OS default encryption to protect your data is just naive. That encryption is just so easy to crack!

Alex Grigoriev
14 Dec 2010 10:03 AM
#

@Gabe: "Compressing and decompressing data usually takes much less time than sending it over the I/O interface"

Access time with random access voids any possible gains from compression. Most large files are either accessed randomly (databases) or are not well compressible (video/audio). Worst loss you'd get if a compressed file is being written in small pieces, less than a compression block which is usually 64 KB or so.

lefty
14 Dec 2010 11:23 AM
#

I'm wondering if this comment thread is the most usability analysis that's been done on this set of advanced file property configuration options. I'm sure that all 10 people who've tried to use EFS on compressed files appreciate the brainpower being focused on this problem.

f0dder
14 Dec 2010 3:35 PM
#

@Alex Grigoriev: iirc NTFS compression means multiple smaller files can be clustered together in a compression unit, which might be interesting for static text files, like .h files?

OTOH, there's no guarantee that the files you use together are clustered together, and while decompression is very cheap CPU-wise, you're usually a lot of CPU while compiling (though I dunno if the read-and-parse step is expensive, or the CPU-maxout doesn't kick in until the optimizing step).

AndyC
14 Dec 2010 7:46 PM
#

Of course, had Explorer used radio buttons, Raymond's suggestion box would instead be filled with "What idiot decided to use radio buttons for the encrypt/compress options? If they'd used checkboxes we'd be able to select both at the same time. Stupid Windows"

Sometimes you just can't win.

640k
15 Dec 2010 12:17 AM
#

@andyc: "If they'd used checkboxes we'd be able to select both at the same time. Stupid Windows"

With a "encrypted AND compressed" radio option this whould be a possible.

Graying out the "encrypted AND compressed" option is valid because the user could move the file to a better file system which supports both.

If an encrypted AND compressed file is moved from a filesystem which supports both, is the file decompressed or decrypted when stored on ntfs?

dude
15 Dec 2010 12:22 AM
#

If the GUI was designed according to m$ own standard, Encrypt checkbox should be disabled when Compress is checked, the user should manually be required to remove compression first to be able to select Encrypt. And vice versa.

Gechurch
16 Dec 2010 1:45 PM
#

@640k

I would find that a lot more confusing. Having a third option that greyed out would suggest to me that it is unavailable in my configuration. I could easily see myself wasting time googling this to figure out how to "enable" that impossible option. As for file systems, NTFS is currntly the only one that supports either of these options, and there is no file system Windows can read that supports both. This may change in the future, and it it does that is the point the UI should be changed.

As to moving a file that is compressed and encrypted to NTFS - I can't see how this can happen. Windows doesn't support any file system that can do both, so that means the only way to copy from one of these volumes is over the network (or some other interface). If you do this, the sending end has to send it according to the protocol. I'd assume SMB doesn't support sending compressed or encrypted files, so would expect the sending end to have to decrypt/decompress first. Anyone know for sure if this is the case?

I just had a play with the checkboxes and, despite this normally being one of my pet peeves, I think it's very clear. As soon as you go to tick the second option the UI "tells" you that you can't use both. Dude's suggestion of disabling the other option until the first is unticked is a bit clearer again, but is slightly less usable (because it requires more clicking).