
From: Bill Gates
To: billn; bradsi
Subject: RE: AARD drivel
Date: Tuesday, August 03, 1993 9:12AM

This all seems fine to me. Has it been sent to Schulman? If not have legal review it and then make sure he gets it and make sure someone calls him to discuss it.

From: Brad Silverberg
To: Bill Gates
Subject: AARD drivel
Date: Monday, August 02, 1993 1:15PM

> From aaronr Mon Aug 2 11:35:34 1993
> X-MSMail-Message-ID: 15DF2338
> X-MSMail-Conversation-ID: 15DF2338
> X-MSMail-WiseRemark: Microsoft Mail - 3.0.729
> From: Aaron Reynolds <aaronr@microsoft.com>
> To: bradsi
> Date: Mon, 2 Aug 93 11:32:50 PDT
> Subject: AARD drivel
> Cc: aaronr
>
> I read with some interest the "Examining the Windows AARD Detection Code" article in the September issue of Dr. Dobb's Journal. I was generally impressed with the technical aspects of the article, and with a few of the opinions expressed, however the opinions about the purpose of this code and the reasons for its existence were not correct. I should know, I was one of the primary people involved with this code.
>
> I preface my remarks with the following commentary. The rest of this is a complete discussion of the reasons for this codes existence, nothing is left out, this is the complete story. You probably aren't going to believe it though, so to some extent I am probably wasting my time explaining it to you. In this age of sensationalist journalism, the background of this code isn't "juicy" enough, so people will probably use the simple tactic of dismissing it as untrue or incomplete. There seems to be a curious assumption running around the world: "Microsoft is a malevolent, Machiavellian organization which is always doing secret evil things according to hidden agendas and you can't trust anything they say." Not much I can do about it if you are pre-disposed to disbelieve what I say.
>
> Windows is tight coupled to the underlying MS-DOS operating system. It relies on a number of very precise behavioral characteristics of MS-DOS which have nothing to do with the INT 21h API. I should know, I am the person who designed most these characteristics in both MS-DOS and windows, and spent a long time figuring out all of the subtle issues relating to them. The reliance on these characteristics puts windows into a very different class of program than any other MS-DOS application. Because of this tight coupling, an MS-DOS "work a like" must have exactly the proper behavior or all sorts of subtle and not so subtle problems will occur.
>
> Microsoft does not test windows on anything other than Microsoft's MS-DOS. We do not consider it our mission to test windows on MS-DOS

"work a likes". That is a problem for the MS-DOS "work a like" people to worry about. If they want to do all the work, and certify their products for use with windows, fine, we are not going to do it for them. This should not surprise anyone. Look at the market. How many other MS-DOS program developers spend time testing on MS-DOS "work a likes"? Testing is a very expensive and time consuming enterprise. It is important not to waste time testing something when the possible additional sales you might get won't even pay for the testing. Sorry that this is not as "warm and fuzzy" as some people would like it to be. Staying in business is like that. If you are an MS-DOS "work a like" and you are trying to catch up with a dominant market leader, you have to work very very hard and you are not being very smart if you expect that market leader, who is your main competitor, to do a bunch of work to help you out. You have to do that work yourself.

I know of 7 MS-DOS "work a likes". On all of these but one, windows will not even start. On the one that it does manage to start on, depending on which mode of windows you run, it has some other more subtle problems. I am purposely vague here because I am in a difficult position. If I name names, I or my company will probably get dragged into court, so I will not name names or be more specific, sorry. "But wait!!!!" You said you didn't test on MS-DOS "work a likes", how do you know this? During the windows 3.10 betas we got a few bug reports about windows not working correctly on some MS-DOS "work a likes". So it seems that a very small percentage of the market may have some problems if trying to run windows 3.10 on an MS-DOS "work a like". In order to be fair and up front with our windows users it might be a good idea to disclose to them in a timely fashion, before they might encounter some possibly data corrupting problem, that they were running the windows product on a non-Microsoft MS-DOS on which Microsoft had not done any testing. This is what the "AARD" code is for. It detects whether the DOS it is running on is Microsoft MS-DOS. If the DOS is not Microsoft MS-DOS, a disclosure message will be displayed to the user that windows, I include all windows components in this, is being run on a DOS that it has not been tested on.

"But wait!!!!" That is not the form of the message that was in the windows 3.10 betas! That is correct. The message that was in the betas was crafted carefully to produce a desired effect: A report back to Microsoft that the message had been displayed. This code was added very late in the beta cycle, we were extremely concerned about it having some subtle bug in it and/or it "misfiring". For this reason we had a very strong desire to hear about every single occurrence of this message in the beta program so we could follow up and confirm that in fact a non-Microsoft-MS-DOS was being used and the code was working properly. This is why the magic word "error" was used. This is the only reason why the word "error" was used. And based on the statements in Mr. Schulman's article, this strategy was successful beyond our wildest expectations. It is still generating "bug reports" a year and a half after it was disabled!!!! Look at the message: "Please contact the Windows 3.1 beta support." Do you still think that is what the message was going to be in the final product if we had left it enabled? Of course not. If you can change one part of the message, can't you change all of it? I think so, don't you?

Mr. Schulman seems to be trying to make a big deal out of several factors which aren't very interesting. I presume that this is mostly because he doesn't understand the reasons. "The effect of the AARD code is to create a new and highly artificial test of (MS-)DOS compatibility." This code is purposely asking this exact question: "is this Microsoft's MS-DOS that Microsoft has tested windows on?" The strange things the code has to look at to answer this question is to some extent a commentary on the quality of some of the "work a likes". Mr. Schulman goes on at length about how this code is "obfuscated and encrypted" and that this is somehow an indication of malicious intent. He then at the end explains completely the exact reason for it!! "An indication that the AARD code's obfuscation is successful is the fact that Novell's most recent version of DR DOS fails the test..." That is the reason for the obfuscation, the complete reason. This code is likely to be targeted by the "work a likes", which defeats the code's purpose to disclose to the user that windows is being run on a DOS that Microsoft has not tested it on. I am not ignorant enough to think this task is impossible, the intent was simply to make it difficult. Since the primary tool used for figuring things like this out is a debugger, it should surprise nobody that one of the obfuscations is to try and disable a debugger. "Anyone with a copy of Windows 3.1 can hex dump WIN.COM [...] and see the error message [...] and the AARD and RSAA signatures." Welcome to the wonderful world of "fix paranoia". This code was added to the betas very late, the last large beta in fact. The decision is then made to not do this. I will not waste time going into the details about why this decision was made. It should be obvious at this point what the reasons were. Now we find ourselves between the rock and the hard place. We don't want this disclosure message in the product, but we want to make the minimal possible change so that the change does not destabilize the product and require us to do another large beta to make sure that the disable didn't break something (probably due to some weird side effect). Leave all the code and the message in, even run the code, just don't display the message. By the way, Mr. Schulman's analysis of WIN.COM brings up an interesting point.

He goes on about how code was added to look at a byte to see whether or not to display the disclosure message. This code was added after the beta went out, but before the decision to remove the disclosure was made. Unlike SETUP and MSD which are not frequently run things, WIN.COM is run every time the user runs windows. A user who has decided windows works OK on the MS-DOS "work a like" he is using might tend to get a little bit annoyed at having to press a key to dismiss the disclosure message and continue every time windows is started. For this reason it was decided to add a command line switch to WIN.COM which would disable the message and continue. As I recall the byte variable was added to WIN.COM, but the code to parse the command line switch was under conditional assembly and was not assembled into the final product. I agree this sounds a little odd, but that is my recollection of the way it happened. This meant that when we decided to not print the disclosure message at all, all we had to do was change the initial value assembled into WIN.CNF so that the default value was "don't display the disclosure message and continue". This meant that the source code DIFF for WIN.COM was a one byte change which is about as minimal a change as you can get. By the way, don't ask me why the code and message are completely removed from HIMEM.SYS and MSD.EXE because

don't remember although I suspect that the reason was we decided to make an unrelated change to these components in this time frame and decided that removing this too would incur minimal additional destabilization risk.

I also note in passing: "Its presence in five otherwise-unrelated programs also suggests a fairly concerted effort, as it is unlikely that five so different programs are maintained by the same person. In fact, the programs probably fall under the domain of several different product managers or divisions." I agree that a concerted effort was involved, the rest is meaningless and I am at a complete loss as to what point Mr. Schulman was trying to make here. The AARD code was all written by one person for one product, windows 3.10. He is correct about the fact that these five different programs were the responsibility of different people, but what does that mean? "Here is a module with a routine named xxxxx in it, call the routine and look at this to see whether the disclosure message should be displayed." "...given the effort required to write this tricky code." About one man week, all by one person, a large part of this time actually being testing as opposed to writing. How much effort is required depends very much on what the knowledge base is of the person doing the work. By the way those signatures, "AARD" and "RSAA" were debugging aids that would have been removed if the disclosure had not been disabled. Since it was disabled, and the whole thing became uninteresting, the signatures got left in. They probably should have been removed from the beta too, oh well.

As I said above, what is going on with this code is probably just not Machiavellian enough for many people to believe it. All we were interested in doing was disclosing to users in a timely fashion that they were running the windows 3.10 product on something on which Microsoft had not done any testing. It seems that even this is something that you can't do because somebody else who is trying to leverage 10 years of your hard work by copying it feels they have a right to expect you to waste a lot of your money doing all the testing for them for free too. As we observe, something as innocent and well meaning as this does nothing but generate a lot of complaints about the fact that you are being "unfair" to your competitors. Apparently a lot of people feel this is more important than us being fair to our users. This is an opinion I refuse to agree with because it fails to serve the most important people, the users of our products.