June 13, 2009

Windows 7 UAC code-injection vulnerability: video demonstration, source code released

If I'm beginning to sound like a broken record to you, I respectfully ask you to hear me out for what I would hope is the last time.

[flv:uaccodeinject.mp4 640 400]

I know from my own experience that large chunks of technical blabber on the topic of software security is not the most enjoyable reading experience. To help illustrate my point better, I've embedded above a very brief 2-minute long screencast to demonstrate the Windows 7 UAC code-injection vulnerability I've been touting. If you don't plan on reading any further, please at least watch that.

Assuming you have some insights into how this code-injection vulnerability works, I want to elaborate on a couple points to reinforce my case.

Firstly, I want to touch on the nature of remote code-execution vulnerabilities and how it relates to this code-injection vulnerability. If you're an everyday Windows user, you would have without a slither of doubt come across the words "remote code-execution" (RCE) sometime in the past or even as recent as today assuming you've applied your Windows patches which covers several RCE vulnerabilities. In case you're not entirely sure what it means, at the most basic level it describes a system executing code provided to it by a remote source without any intervention from the user. RCE vulnerabilities not only affect Microsoft products, but Adobe Reader, Mozilla Firefox and many popular third-party softwares millions of users trust.

RCE by itself warrants some attention, but with the introduction of default UAC policy in Windows Vista, the potential impact of RCE vulnerabilities were actually reduced because the malicious code can no longer assume full administrative privileges, instead, limited to what the target application was running which in most scenarios was medium-level or even low-integrity like in Internet Explorer. However, in conjunction with the default Windows 7 UAC policy and this vulnerability, the potential impact of RCE vulnerabilities is raised, as the malicious code executed could silently elevate itself to have much more free reign over the system than before. If this isn't enough indication that the default security policy in 7 is worse than Vista, than I don't know what is.

Secondly, besides the obvious malicious use for the UAC vulnerability, there is nothing stopping it from being abused by legitimate developers and their applications. After suggesting such a scenario in my original article, one such developer have already expressed

interest in using this vulnerability in such a way that will remove UAC prompts from his applications.

Now I'm fairly confident that this developer has the best intentions for his users, but what this means if it is ever applied in practice is that for the large majority of users who will use the default UAC policy, UAC prompts be only a waste of their time. I say this because if some applications can elevate themselves with or without the user agreeing to a prompt, the prompt's effects are nullified. To look at it in another light, at the default Windows 7 UAC policy, it's as good as having UAC prompts turned off entirely.

Last but not least, since Microsoft has known about this for half a year as well as indirectly acknowledged and ignored this vulnerability, I have asked Leo Davidson to release the proof-of-concept source code and test application into the wild for public scrutiny. If Microsoft is right in saying this has no security implications, then this should mean nothing. If they are not then, well, at least there is still time to do something about it. A month to be exact.

I realize Microsoft will *not* by any stretch of the imagination return Windows 7 to the Windows Vista "always on" mode of UAC, there's too much to lose. What I would like is for Microsoft to acknowledge that there is an increased security risk with using the default Windows 7 UAC policy, and communicate this to users where appropriate.

I'm not saying this is the end of the line for Windows 7, it's an amazing operating system. But for Microsoft to simply ignore this seems irresponsible to me. There are so many people I'd like to evangelize the product to once it ships, and I'd hate this to be one thing I'd also have to mention.

← Older Newer →

102 insightful thoughts

June 13, 2009 at 1:16 am

A nice demonstration video of a rather worrying weakness in UAC.

Matt Sharpe

December 22, 2010 at 10:32 am

This UAC issue has been fixed over a year ago. All of these comments are over 18 months old.

UAC issue fixed

June 13, 2009 at 1:48 am

Rahul

i think action center must alert the user at default UAC policy with a warning message & have the ability to raise the UAC level to “Always Alert” from within action center. this would serve all the purposes. a) microsoft could leave UAC setting unchanged. b). End user at the same time know security implication of it.

June 13, 2009 at 1:51 am

GoodThings2Life

I’d be content if they switched the default user account creation to Standard and left the UAC default alone but provided some type of initial dialog and description in UAC configuration that explicitly described the risk.

June 13, 2009 at 1:57 am

http://images.chron.com/blogs/techblog/uac.jpg

Matteo Gazzoni

Default – Recommended if you use familiar programs and visit familiar websites.

June 13, 2009 at 2:01 am

Mike

If anyone purposefully tries to circumvent UAC in their software product they will be subjugating themselves to future problems the same way people did when they wrote their web applications to work only with IE6. The day will come when Microsoft will make the default user a standard user account..

June 13, 2009 at 2:03 am

Xepol

I still feel that if the apis that allows this happen required elevation, the whole argument would be moot. Elevation would be required before the attack.

I do not personally feel that DLL injection has significantly improved anything I have seen. For MS, if it wants to patch the OS, the can do it properly. For anyone else – they probably shouldn't be able to do it at all.

Yes, it might cause a few apps to fail in the process, any those few valuable programs that use this will have to figure out a better way to get the job done (or live with elevation prompts), but it will be worth it in the end.

And MS can implement that solution pretty quickly (although testing might be more extensive and time consuming).

June 13, 2009 at 2:54 am

WELL-DONE EXPOSÉ OF THE DANGEROUS FLAW!

Mark Russinovich SOLD OUT THE REST OF US for a fat paycheck at Microsoft.

Mark Russinovich's CORPORATE-BULLSHIT SPIN on the UAC debacle and Microsoft's UNWILLINGNESS TO REWRITE WINDOWS AS A SECURE PRODUCT leaves us all wasting millions of dollars and hours on an UNSTABLE O/S and third-party crap from scammers like Symantec and McAfee (who were just fined $750,000 by New York's Attorney General, for credit-card scamming of their "beloved" customers).

LONG ZHENG, KEEP UP THE GOOD WORK!!!!!

June 13, 2009 at 3:01 am

Laslow

@Xepol – The trick is, the app that perform these tasks *doesn't* require elevation in order to do the code injection. That's the point. If you'll note in the video, no UAC prompt is presented when the proof-of-concept app is launched, therefore any malicious program that does the same thing will also not display a UAC prompt.

Think 'Bonzi Buddy' installing itself silently via this method – that's scary enough!

June 13, 2009 at 3:05 am

This leak isn't working.

Peter van Dam

I'm on 64bit running this app (64bit version) I hit any button and it requests UAC approval like it should do. This with default UAC settings. So this problem is FIXED

June 13, 2009 at 3:09 am

★Long Zheng

@Peter van Dam: That might be a bug. Leo was reporting problems with the compiler, and for some reason the UAC prompts would be triggered. I have personally tested the updated 32bit app, which still exploits this vulnerability. The 64bit proof of concept might be bugged.

I'll make sure Leo knows of this. But please for the moment do not assume it has been fixed. 🙂

June 13, 2009 at 3:27 am

Quppa

So, after reading the Channel 9 thread (http://channel9.msdn.com/forums/Coffeehouse/473037-UAC-controversy-the-last-episode/), the situation seems to be:
– When running as Admin in Vista or 7, UAC is not going to prevent malicious code getting admin privileges.
– The default setting in Windows 7 makes the above a bit easier than the default Windows Vista setting did.
– Running as a Standard User is the only way to be properly secure, but that's too annoying for most people.
– The main point of UAC prompts in Vista/7 when running as admin is to annoy people in order to encourage developers to write better code.

June 13, 2009 at 3:29 am

★Long Zheng

@Quppa: The sad fact is that this invalidates the last point too.

June 13, 2009 at 4:35 am

Matteo Gazzoni

@Long Zheng:
If a developer prefers to write hacks instead of properly fixing his code it's not Microsoft fault.

June 13, 2009 at 5:36 am

d1

i dont see what the problem is:

– the first account during windows installation is the admin accout; we want this because the person installing windows IS an admin user and NEEDS admin rights

– the second (and any other) account created, is by default a standard account.

– if you ever installed windows you should be smart enough (being of admin status) to create a second account account (which defaults to standad user) for your own personal use (keeping the admin account solely for admin duties)

– standard users arent affected by this at all; this is what EVERYONE should be using.

– i will not go into the argument that UAC is/isn't a security boundary, neither on how its perceived, nor would i on the fact that the system would need to be compromised first, nor that an admin user would have to explicitly activate their choice applications first (by an abusive or not developer); if an admin user is stupid, its not a windows concern

– YES, the majority of users ARE and WILL remains IDIOTS; that is not a windows concern.

– admin users know what they are doing; and DO NOT need the extra prompt; and LIKE the less prompts

– if an admin is unsure (a though that should not cross a real admin user), the admin user can increase the default UAC behavior

– There is a case that upon windows installation Microsoft whould at that point recommend that the user create a standard account for their own use; leaving the admin account (required on installation) solely for admin duties.

– I repeat, admin users LIKE this, and standard users DO NOT care.

Did I miss something?

June 13, 2009 at 5:42 am

JeffU

And again I say, who cares. People are GOING to say YES to any prompt that comes on their screen, 95% of windows users are already Administrator.

Therefore your point is Moot Long, you're making mountains out of mole hills. It's insignificant.. and you're really making yourself look very silly at this point for reposting this.

June 13, 2009 at 6:12 am

Leo Davidson

@Peter van Dam: Are you using RC1 or a different build? I've tested the 64-bit version on RC1 and it seems to work fine.

Do you see a UAC prompt if you create a folder below Program Files using Explorer? If so your UAC settings are not the default.

I'll double-check the 64-bit version again in a moment but it definitely worked a few hours ago.

June 13, 2009 at 6:13 am

Leo Davidson

@JeffU: People who say "yes" to everything cannot be helped. By your logic there is zero point to Standard User accounts either as those stupid people will say yes to elevate everything into the Administrator account.

June 13, 2009 at 6:25 am

Leo Davidson

@Quppa: See my comment in the Channel 9 thread.

To say that things can get admin access in both Vista and Win7 belittles the point that in Vista stuff has to lurk waiting for the user to do something particular — or put up a UAC prompt and hope the user

clicks on it — while in Win7 (by default) it anything code that is running (e.g. through an RCE or buffer-overflow) in just about any process (other than low-integrity IE) can *immediately* and *silently* get full admin access and install itself as a rootkit.

Anyone who says that none of this matters because stuff could spoof UAC prompts in Vista is either trying to make excuses for a broken system or must, if they follow their logic, also argue that it is pointless to have standard user accounts. You can spoof or hijack a UAC prompt triggered in a standard user account the same as you can spoof one from an admin account.

It's BS logic and quite frustrating to see if used as an argument. Those people are essentially saying that "security is pointless so why bother with anything at all?" Except they won't admit it and they cling to the standard user case as if it is immune from UAC prompt spoofing, as if it isn't more irritating than the admin case on Vista was (and thus unlikely to be used by many people), as if it was the default (when it's not, although one of the posters in that thread claimed it was).

June 13, 2009 at 6:27 am

@d1: "Did I miss something?"

Leo Davidson

Yes, your second point deviates from reality.

June 13, 2009 at 6:30 am

@d1 — Oops, I mean your third point.

Leo Davidson

Almost nobody creates a standard user account. You can live in a dream world where people do, but they don't. They don't know that they have to, they aren't told to, OEMs don't do it for them, and even if people did know they should use standard user accounts, they will not do so because the UAC prompt spamming and password typing with standard users is *worse* than what drove many people crazy on Vista.

Plus the UAC prompt spoofing issues which supposedly make all of this moot (according to some apologists from Microsoft) applies to standard user accounts the same as it does to admin accounts. So appearntly there's no security there either. We should all give up on this crazy idea of security and stop complaining, apparently!

June 13, 2009 at 6:39 am

Leo Davidson

@Matteo Gazzoni:
“If a developer prefers to write hacks instead of properly fixing his code it’s not Microsoft fault.”

That’s true, and I wouldn’t use this for a legitimate app.

Still, what Microsoft have put in for their own apps is a hack too. If Microsoft are using a hack it is their fault. And they won’t let anyone else use their hack, which is unfair on third-party developers.

Why do Microsoft get a backdoor hack that means they don’t have to properly design their code (so that it doesn’t prompt so damn much, and to reduce the annoyance of their stupid prompts-about-prompts) when nobody else is allowed to do the same?

Why are Microsoft arguing that they want everyone to move to the standard user case when their code is still awful to use for admin work via elevation from standard user?

Why are Microsoft saying that the UAC prompts are enabled for admin accounts on third-party code to purposely make elevation requests annoying for everyone and reduce the amount of them as much as possible, when they have not applied that rule to themselves and their code is, and always has been, the absolute worst offender???

June 13, 2009 at 6:41 am

Leo Davidson

@Peter van Dam: Just tested the 64-bit version again on Build 7100 (the RC1) and it definitely works.

It probably won’t work on Build 7000 (the beta) because of differences in which processes can auto-elevate. I could compile a version that does work on the beta but I assume most people have upgraded to RC1 by now.

I’ll put a note on my page to let people know it only works with RC1 and above, though.

June 13, 2009 at 6:55 am

Mike Regan

Long, I appreciate the extra time you and Leo spent with the post and video. I feel you have expounded your argument quite precisely and it gives me pause on my faith in the UAC default setings for Windows 7 while running as an administrator.

If Microsoft's Windows 7 UAC default setting in conjunction with the white list is compromised, is there any suggested way to fix it or should they move the default setting to the Vista level? Could the Windows process manager scan for an injection before spawning a process from a trusted white list process? It seems like the best way to patch this is require a process to be elevated before allowing injection into a white-listed/already-elevated process.

June 13, 2009 at 7:21 am

Leith Bade

I have the feeling Microsoft will not do anything about this until someone releases a virus into the wild that makes use of this. Real viruses always seem to prompt Microsoft to release patches far sooner than if there is not one.

June 13, 2009 at 7:42 am

Matteo Gazzoni

One should not rely on a non-security feature for his security. This kind of vulnerability requires that something is executed on the user machine. There are some measures to avoid/limit this; UAC is not one of them. The fact that the user can prevent the execution of that something (by clicking No) is, i think, only a collateral effect.

About standard users and UAC. This is a convenience for not to switch to an admin account, it is clear that more convenience (often) means less security.

I agree that Microsoft with 7 has done nothing or very little to ease the switch to standard user; and Explorer + standard user is still a joke.

June 13, 2009 at 7:51 am

@Matteo Gazzoni:
"This kind of vulnerability requires that something is executed on the

Leo Davidson

user machine”

There have been loads of drive-by RCE vulns lately in stuff like Flash and Adobe Reader. If you visited a site which silently dropped a vuln on your system, would you rather it was restricted to medium-integrity or that it could instantly and silently get full-admin rights to install a rootkit that ensures the malware is never detected?

(Your personal files are at risk either way, unless you put them somewhere which requires full admin to read/write, but UAC used to make it much harder for that thing to install itself deep in your system than it does with Win 7 by default.)

Obviously the RCE flaws are a much bigger deal, but they exist and more keep being found. To me it makes sense to limit what can be achieved via those flaws while we wait weeks for slow companies like Adobe to respond to them.

June 13, 2009 at 7:57 am

Leo Davidson

@Mike Regan:
“is there any suggested way to fix it or should they move the default setting to the Vista level?”

In the short term, I think MS should either set UAC to silently elevate everything by default, or to always prompt by default. The current default of showing easily bypassed prompts only for third-party apps makes no sense from any angle, except snakeoil marketing and security theatre, and MS are doing a disservice to everyone by pretending otherwise.

In the long term, I think MS should have refactored their code so that it does not prompt as often and so that the UAC dialogs could display more information/context about what is about to be run (which makes spoofing harder and removes the need for prompts-about-prompts). I go into mind-numbing detail about this on my page if you fancy a numb mind. hehe 😀

June 13, 2009 at 8:16 am

What about the Low Integrity level in IE (Protected Mode)?

Matteo Gazzoni

June 13, 2009 at 8:28 am

Nick

OK, so Windows 7's auto-elevating UAC does open up a window for vulnerabilities, but you are essentially trading security for convenience. Auto-elevation is convenient. Constant nagging isn't. An unfortunate side effect of auto-elevation is lower security.

A point could be made that if one UAC prompt can be bypassed, all UAC prompts are effectively useless. I still see quite a few mitigating factors:

– The demo doesn't show anything scary, i.e. a program installing itself surreptitiously without UAC prompt:
1. in the command-line elevation scenario, you're *manually* entering a command. This is pretty much a remake of the old deltree c: /y joke!
2. in the second scenario, you're launching an elevated Internet Explorer process. But you can't instruct it to go to a particular webpage! What good is it to have an elevated Internet Explorer process lying around if you can't do anything with it?

What would be scary would be to be able to inject arbitrary commands, either in the command-line scenario (without the user having to TYPE it) or the elevated Internet Explorer (loading a webpage automatically). But the flaw doesn't reach that far.

So correct me if I'm wrong, but in order to have access to enough privileges to do serious damage, elevating your own program is still needed. I don't see how a malware program could benefit from this auto-elevating flaw.

– Some clamor that the solution is to change, by default, the UAC security level. I'm not sure that's really useful. When UAC is set to a more stringent level, the elevation prompt is the "signed program" one, and it isn't scary. It says that a *signed* program wants to make changes to this computer. A non-expert user could be easily coaxed into clicking "Yes".

– If you want less prompts, you will have a more vulnerable system. I think the minor added risk is worth the added convenience, but if you disagree, the UAC setting can be changed to a more stringent level if desired!

So, while this UAC vulnerability is not an non-issue, its consequences are being greatly exaggerated.

June 13, 2009 at 8:33 am

Leo Davidson

Low-integrity IE doesn't help with Flash vulns. because Flash is still hosted inside a medium-IL proxy process for compatibility. (You'll see FlashUtil10b.exe appear in task manager whenever protected-mode IE is on a page that uses Flash. Process Explorer should show it's a medium-IL process.)

Low-integrity IE definitely doesn't save you from something like Adobe Reader (or any other document/media viewer with a buffer-overflow vulnerability) being exploited by a malicious data file, since those things run at medium-IL outside of IE.

(It's be nice if document viewers ran at low-IL but almost none of them do. That includes stuff like Microsoft's own Preview Handlers for viewing Office documents in Explorer which, last time I checked, all explicitly opt-out of low-IL. If MS can't do it then it seems a lot to expect everyone else to.)

IE and Chrome the only things I can think of which run at low-IL and even those have exceptions like Flash which they host at medium-IL.

Plenty of other network connected processes run at medium-IL, including Firefox, Safari, Opera, torrent clients, etc. etc..

Which isn't to say that low-integrity IE/Chrome is useless. It's a great idea and I wish more things could be run at low-IL. It just isn't a silver bullet.

Microsoft had the "defence in depth" mantra and it's a good principal. It should be applied here!

June 13, 2009 at 8:37 am

Leo Davidson

@Nick:

The demo app could easilly be turned into a command-line tool, or piece of injected code, which runs whatever it wants elevated without the user seeing anything.

Combine that with an RCE vuln., like those recently exploited in the wild in Flash and Adobe Reader, and you've got a vuln which can silently gain full admin rights on a default install.

If you're asking us to create such a combination of vulnerabilities then I think you're missing the point.

"Auto-elevation is convenient. Constant nagging isn't. An unfortunate side effect of auto-elevation is lower security."

As I keep saying, people asked for MS to improve things but nobody asked for *this*.

They could have reduced the nagging in better ways, and improved the Standard User experience to boot, if they wanted to do things right. My page suggests a bunch of ways they could've done this.

June 13, 2009 at 9:24 am

Matteo Gazzoni

@Leo Davidson:
You've said that using a standard user is too annoying because of OTS prompts. What about the first user created, the so-called admin? It's a standard user, isn't it? (except for the split token and the local admin group, I think)

I say this because it's ridiculous, IMHO, to create two accounts (admin and standard) for a computer with only one user. The standard user accounts, as of now, seem more a thing for business.

June 13, 2009 at 10:07 am

Brandon

@Laslow –

That is a poor example. Bonzi Buddy doesn't need admin privileges, it can do what it does just fine even under a standard user account. Honestly, there's really little or no advantage for a single-user system from a malware perspective. Besides, UAC prompts were never meant to stop malware. If you agree to run an untrusted executable like BonziBuddy.exe, you're already on your way to trouble, and your best hope is for something like Windows Defender or IE's Safety Filter to save you.

Note: If Bonzi Buddy were running at low integrity, such as if it took advantage of an exploit in Internet Explorer, it would NOT be able to use this mechanism to break out of the low integrity sandbox and install itself.

June 13, 2009 at 10:10 am

@Leo –

Brandon

In order for anything to run outside of Low IL from IE, such as opening a PDF document, the user needs to consent to an elevation prompt (the Protected Mode IE prompt in this case). The exploit demonstrated does not prevent that prompt from showing.

That seems to be a big part of what you're missing here.

June 13, 2009 at 10:12 am

@Matteo Gazzoni:

Leo Davidson

"You've said that using a standard user is too annoying because of OTS prompts."

I should clarify that I mean it's going to be too annoying for most people to use, given the complaints about the less-annoying UAC prompts on Vista and Microsoft's botched response to those complaints.

I personally do use a Standard User account on one of my Vista machines and I don't find it annoying, but it is a machine that is used as a home theatre PC where the desktop is rarely seen at all and where I don't want random people in my living room to have admin access.

"What about the first user created, the so-called admin? It's a standard user, isn't it?"

No, an admin account is never a Standard User account. Standard User accounts have limited rights and can't do admin stuff.

(The name "Standard User" is confusing because it *isn't* the standard. Admin accounts are what get created by default and what almost everyone uses outside of locked-down business environments (where UAC is largely out of the picture). So using an admin account might be

"the standard" — as in "the normal thing that most people do" — but admin accounts are not Standard User accounts.)

"It's ridiculous, IMHO, to create two accounts (admin and standard) for a computer with only one user"

I don't think it's ridiculous. One user can have multiple roles and separate accounts seem like the best mechanism for segregating the privileges which those roles require. What needs to improve is the amount of hassle involved with switching roles/accounts. UAC over-the-shoulder elevation was a step in the right direction compared to the old way of using fast-user-switching, but there's still a lot of room for improvement.

Speaking of roles, I don't see why you couldn't have a Standard User account that was allowed to elevate to a nominated Admin account via a button click on the secure desktop, without having to type a password. So long as the secure desktop actually lives up to its name that should be as secure as typing a password and gives you a proper security boundary combined with the convenience of single-click UAC prompts.

(It's still be vulnerable to spoofed prompts, but so is the current over-the-shoulder elevation. It's really annoying me that Microsoft are dismissing the code-injection issue because of prompt spoofing when prompt spoofing affects every UAC mode and, by that logic, means that standard user accounts are insecure. It's also really annoying me that MS refuse to be drawn into a conversation about how they could make it harder for prompts to be spoofed. IMO they're just looking for stuff to say that might trick people into dismissing the issue, but what they're saying right now doesn't add-up.)

June 13, 2009 at 10:14 am

@Brandon: I think you're missing something, not me.

Leo Davidson

Do you see a UAC prompt when you save a PDF document and open it in Adobe Reader? Nope. People assume that documents are harmless.

Do you see a UAC prompt (or any prompt at all) when you visit a website using Flash, after installing the Flash plugin like everyone has for YouTube etc.? Nope.

In both those cases you could get malicious code running on your machine from doing routine actions that most people consider harmless.

And in both those cases that code can cause more of a problem if it can immediately and silently install itself as a rootkit. (Not that it cannot do a lot of damage without admin access, but things are worse and harder to fix with it.)

June 13, 2009 at 10:20 am

@Brandon:

Leo Davidson

In fact, I thought I'd double-check what happens with PDF files in IE, since I usually use Firefox… I clicked on a PDF file and it immediately opened inside of the IE browser without a single prompt of any kind whatsoever.

What's more, it spawns AcroBroker.exe (Adobe PDF Broker Process for Internet Explorer) as a medium-IL process in which the Adobe Reader ActiveX control is hosted.

So even in protected-mode IE, with PDF files, I think your argument falls apart.

I could disabled the Adobe Reader browser plugin, but even then the only prompt that IE displays is a Save As dialog asking me where I want to save the file. The document is saved to disk and I double-click it and it opens in Adobe Reader without any further prompts.

(I'm testing using IE8 on Vista. Apologies if something is vastly different on Windows 7.)

June 13, 2009 at 10:34 am

Maurice

Sorry to burst your bubble but the latest build of Windows 7 does create Standard Users by default, after the first admin of course.

June 13, 2009 at 10:39 am

Matteo Gazzoni

@Leo Davidson:
"No, an admin account is never a Standard User account. Standard User accounts have limited rights and can't do admin stuff."
But the first user is an admin with admin approval mode. When he logs in, he has the same rights of a standard user.

"Speaking of roles, I don't see why you couldn't have a Standard User account that was allowed to elevate to a nominated Admin".
For personal computers (= with one user which administer himself) would be nice to rework the concept of privileges and accounts. There should not be only one set of privileges per account. The user could be both "standard" and "admin" in the same account (in fact, in the same way that works Admin Approval Mode).

June 13, 2009 at 11:02 am

Leo Davidson

@Maurice: "Sorry to burst your bubble but the latest build of Windows 7 does create Standard Users by default, after the first admin of course."

Nobody claimed that Windows *can't* create standard user accounts. Of course it can! It just doesn't by default.

The argument is about the default security that 99.999% of people will use. Saying that a user can create an additional account and switch to using it is about as relevant and likely to happen as the same user switching to Linux. Equally, people who know about thigns in detail can choose to set UAC to the Always Propt level for their admin account. Doesn't change the defaults and the absolute stupidity (aside from marketing!) of the defaults.

And in case you haven't noticed, there was a huge backlash against the UAC prompts in Vista. Those prompts are *worse* for standard users even in Windows 7.

Almost nobody is going to create a standard user account in Windows 7.

@Matteo Gazzoni:

"But the first user is an admin with admin approval mode. When he logs in, he has the same rights of a standard user."

Sort-of. UAC tries to do that, and succeeds in many ways if it's set to Always Prompt, but it's not literally the same as having a standard user account. (It is not a strict security boundary like a separate users is. There are holes in UAC which don't exist for standard users. Some of those holes are inherent and some are ones that could be improved if MS wanted to. Some issues like prompt-spoofing, or just tricking people to run stuff they shouldn't, exist for standard users as well.)

So what UAC does (or did) is in many ways similar to using a standard user account but it is not the same and it is not as secure (nor as

annoying).

However, now with Windows 7, by default, UAC does a much worse job because it's now so easy for code to silently and immediately bypass the UAC prompts and jump from the limted-admin token to the full-admin token.

June 13, 2009 at 6:54 pm

Great job, Long Zheng !

OutOfTimer

June 13, 2009 at 9:27 pm

Is it possible to set UAC then to Always prompt yet turn off the secure desktop via Group Policy like Vista?

LongLiveLeo

June 13, 2009 at 10:28 pm

The source code is now online in HTML format as well. Start here:

Leo Davidson

http://www.pretentiousname.com/misc/W7E_Source/Win7Elevate_Inject.cpp.html

I also converted the step-by-step guide in the readme into HTML:

http://www.pretentiousname.com/misc/W7E_Source/win7_uac_poc_details.html

Now you don't have to download the source zip or have Visual Studio to see how simple it all is.

June 13, 2009 at 10:33 pm

@LongLiveLeo: Yes – http://technet.microsoft.com/en-us/library/dd835564(WS.10).aspx

Quppa

But not using the Secure Desktop defeats the purpose. (Or at least it defeats what lots of us assumed was the purpose.)

June 14, 2009 at 3:22 am

@Leo Davidson

Peter van Dam

Sorry for my late reply. I'm running build 7100, and like I said, i'ts really in default UAC mode. So to answer your question, No I donnot see a prompt when adding a folder to program files.

I'm running no antivirus, no antispyware (except defender), and I haven't changed any diffecult registery settings that might effect UAC.

June 14, 2009 at 3:24 am

@Leo Davidson,

Peter van Dam

oh, and Leo, the prompt tells that sysprep.exe wants to get admin privileges in case it helps.

June 14, 2009 at 5:01 am

@Peter van Dam:

Leo Davidson

Weird. Could be an update made sysprep.exe no longer auto-elevate but AFAIK it's working for everyone else who has tried it, including people I know who use Win7 daily and would have it kept up-to-date.

Something could've gone wrong with the permissions on sysprep's directory, perhaps. The auto-elevate stuff only happens for exes in specially permissioned folders.

June 14, 2009 at 2:21 pm

AgNr

So, you don't expect MSFT to change the UAC but that MSFT should "communicate" this? How? Tell everyone that UAC is fine but "oh btw the default setting isn't secure, so we suggest that YOU change the setting from its default value".

"… and I'd hate this to be one thing I'd also have to mention." If a default setting isn't changed by MSFT, and we do know that a lot of people run on default settings and don't read much, then given the above I am afraid you will have to mention it.

June 15, 2009 at 4:20 am

people

Windows 7 is what stupid people asked for. Windows Vista is better than Windows 7.

June 15, 2009 at 11:17 pm

OutOfTimer

"Windows 7 is what stupid people asked for."

This may unfortunately be true. The fact that Software Explorer was removed from Windows Defender is another example of that. Now they're also going to control everything we do with our computers.

June 16, 2009 at 1:43 am

Peter van Dam

@OutOfTimer

Yeah, I also miss that software explorer in Windows Defender. Why the removal??? Just why? The program worked perfectly fine, and I didn't hear anyone complaining about the software explorer in Windows Defender…. More like they liked it alot, but didn't really think it should have been in Windows Defender.

Right now it's just removed. No more one place to go to have a nice overview on what should startup or not, and something that monitors it. This is just another really bad change in 7.

Windows 7 makes some mistakes fixed. But why add new bugs, new security problems, new startup issues??? Is this something you remove just so they can sell Windows 8 that brings back these things???

June 17, 2009 at 11:12 pm

link8583

@Leo Davidson
"I clicked on a PDF file and it immediately opened inside of the IE browser without a single prompt of any kind whatsoever.
What's more, it spawns AcroBroker.exe (Adobe PDF Broker Process for Internet Explorer) as a medium-IL process in which the Adobe Reader

ActiveX control is hosted.So even in protected-mode IE, with PDF files, I think your argument falls apart."

You're wrong.

take a look at this document :
http://www.adobe.com/devnet/reader/articles/reader_compatibility/readercomp_pdfrenc

it says the following : "One of the benefits of using the in-proc DLL model in Acrobat 9 is that Acrobat can operate more securely in the context of browsing the Web with Internet Explorer 7 on Vista. By default, InternetExplorer 7 on Vista runs in a low-rights mode, which means that the Internet Explorer 7 process, and all in-proc DLLs such as Acrobat 9, are more limited in what they can do."

It clearly states that adobe reader's rendering is done within the IE process, at low integrity.

AcroBroker.exe does NOT host the adobe reader plugin, it is there only to show the "save as" dialog box which asks the user where the file should be saved if the user clicks on "file", "save as", and then copy the file from protected storage to the real file system accordingly to the user decision.

It's exactly the same as the IE process broker (ieuser.exe with IE7, ielowutil.exe with IE8). A malware cannot write where it wants if it manages to take control of iexplore.Exe, the worst thing he can do is to show the "save as" dialog to the user. If the user confirms it, then he will have to execute manually the file that the malware wanted to copy to the user profile… It's exactly the same as if a site asks you to download a malware and run it yourself ^^

About adobe flash player, I didn't find a document to confirm what I think, but I looks like it is the same thing as with adobe reader: flashutil9b.exe runs in the background as medium integrity process to allow flash player to persist its cache and settings outside protected mode in his own folder only (a flaw would not permit to control flashutil9b.exe to write elsewhere).

Flash player rendering takes place INSIDE iexplore.exe process, not in flashutil9b.exe. You can confirm this by visiting youtube, you will see that while a video is playing, flashutil9b.exe stays at 0 or 1 % cpu usage, as opposite to IE which suddenly use much more CPU. This clearly shows that the flash ax plugin works INSIDE the IE process, in low integrity mode.

If you can confirm that a malware can't use the flaw you discovered from a low integrity process, then it means that Internet Explorer will be the ONLY secure browser when running windows 7 with default UAC settings, and that the UAC is not useless for IE users.

A firefox user would be exposed by firefox, adobe reader, and flash flaws, all of these could allow a malware to gain admin rights

But a IE user would be safe because IE, flash, and adobe reader run in low integrity mode, thus a malware could not even install as user mode.

June 18, 2009 at 6:26 am

Leo Davidson

@link8583:

Thanks for the corrections. I didn't realise that was all the brokers did. I'll post corrections to places where I've made incorrect statements about that (as far as I can remember at least!).

In that case IE (and probably Chrome) should be safe from the problem, I agree.

Still leaves Firefox etc. and issues that affect local media players/readers/viewers. (e.g. The PDF issue will still affect you if you download the PDF and then view it. Ditto a buffer-overflow exploit affecting, say, MP3 tag parsing in a music player, or something like the old WMF issue.)

June 18, 2009 at 8:52 am

link8583

"In that case IE (and probably Chrome) should be safe from the problem, I agree"

according to these pages,
http://dev.chromium.org/developers/design-documents/sandbox
http://blog.chromium.org/2008/10/new-approach-to-browser-security-google.html

Chrome 1 does not sandbox plug-ins by default.
Is it still the same with chrome 2? I did not find any information about it.

"Still leaves Firefox etc. and issues that affect local media players/readers/viewers."

Too bad that mozilla is too confident about firefox security to care about implementing a sandbox…
firefox + flash/adobe reader is the next IE6 SP0!

However, I think most antivirus will block code injection between process by default as a proactive security measure. I believe Kaspersky internet security has been able to do that for a long time. Not a panacea, but better than nothing for most users who are going to use the default settings (if they use a decent antivirus at least!)

“The PDF issue will still affect you if you download the PDF and then view it.”

Indeed. And unfortunately, it’s not possible to launch an adobe reader embedded inside IE to view a downloaded pdf file. That’s a shame, it would be so easy for adobe to provide a stand alone pdf reader running at low integrity as they have done most of the job for the IE plugin implementation. That would be really usefull since adobe reader is heavily targeted by hackers (and aslr and dep are irrelevant for non-buffer-overflow flaws).

June 19, 2009 at 1:23 am

Jad

Trackback:
http://www.theregister.co.uk/2009/06/18/windows7_security_hole/

June 19, 2009 at 1:31 am

Steve

In my experience, those who are conscious about security and understand computers turn UAC off, and accept their own responsibility in doing so. Those who don’t understand (average user) click ok regardless – “A window popped up on IE saying i didn’t have a virus checker, so i downloaded something called ‘wibbly wonky fake virus checker’, then click ok on everything to install it” – happens far too often. UAC is a smokescreen to fob off the responsibility of security to the user. Yes, microsoft should fix this, but better still just sort out user proveleges so this junk UAC system becomes unnecessary!!! (and I can stop handing out linux disks…)

June 19, 2009 at 6:46 am

Windows 7 Rocks. The rest sucks.

Hassan

June 19, 2009 at 7:15 am

A serious, long-standing question, not a flame:

James Butler

Red Hat's Fedora Linux distribution (and many other Posix distributions) creates a "root" account (which is actually *more* privileged than Windows' admin accounts) during installation, and at the end of installation the admin *must* create an unprivileged user account to run the system as. After the installation has completed, your are simply unable to log in as "root", forcing any user to use their unprivileged login, and then switch (su) to root when needed. There is no "do you want to temporarily switch to root to allow this thing to work" prompting, because everything (with few exceptions) is designed to run strictly in userland. If you need root (typically for application installation and a few other things), you go through the simple process to become root, execute the operation and logout of root, back to the unprivileged identity. No UAC. No prompting. Just an alert that says "no can do" and the ability for the user to then take manual steps to elevate themselves for the task at hand.

Again, a serious question and not a flame: What is stopping Microsoft from doing the same thing? Thanks.

June 19, 2009 at 7:32 am

CConsultant

In testing last of win 7using IE8 to test Bing I had an attack that did take advantage of this hole.
I was doing a random search in Bing and was noting that action were being done in the background wile Bing was loading . i was alerted by AV software an attack was trying to install files and I had no UAC pop up to tell me some was accessing or trying to access the system . the system crashed and I had to do a repair of the RC install and logs were lost or some deleted as I could not fine them but there were 2 new folder n the system 32 that were not part of the install and had nothing in them.

I did not see this happen in Firefox or Flock when testing only in IE 8 and on IE 7 in XP does the pre-load video in a search using Bing seem

to have problems.. Files and Data are being search for when you are using Bing. in IE8 on the Win7. it was even more aggressive in searching my system in some of my testing. IE8 is something I would never use or tell my client to use but as we know many new users and other that just do not understand secuirty as we do will use Many end users are clueless to theses issues with Ms and how bad these holes are. I re-posting info to this site to Clients and media the more info we get out should get MS to do something I would hope

CC

June 19, 2009 at 8:48 am

link8583

"There is no "do you want to temporarily switch to root to allow this thing to work" prompting, because everything (with few exceptions) is designed to run strictly in userland."

Windows only asks you administrator rights when you want to do something that has system wide consequences (change the time/date, install an application, install an update, register an activex control, start/stop a service, alter a system file or a file you are not allowed by ACLs to write to, …).
Everything else runs as limited user without UAC prompts.

Could you give us an exemple of something on windows that prompts the user for admin rights, but not on linux?

" If you need root (typically for application installation and a few other things), you go through the simple process to become root, execute the operation and logout of root, back to the unprivileged identity. No UAC. No prompting. Just an alert that says "no can do" and the ability for the user to then take manual steps to elevate themselves for the task at hand."

that's much more complicated and time wasting than just clicking on "Continue" when the uac prompt shows up.

"Again, a serious question and not a flame: What is stopping Microsoft from doing the same thing? Thanks."

the users.
Windows users just want the task done. They don't want to worry with security. That's why Windows 7 UAC with reduced efficiency was so well received by journalists and users who don't care if there is a side effect on security.

But the best approach was the Vista UAC one. Much more simple that the linux approach where you have to type the root password to give root rights to an application. And it is not less secure (a malware could not automatically click on continue to gain admin rights).
Finally, to get a good security level on windows 7, just set the uac to his max level, the same as on Vista.

“In testing last of win 7using IE8 to test Bing I had an attack that did take advantage of this hole.”

I would be really surprised if a malware would already try to exploit ANY Windows 7 specific flaw, as it is a lot of work for hackers for less than 0.5% of potential victims.

In fact this flaw in UAC cannot be exploited to run malware from the web. There need to be another flaw for this (or the user needs to lauch an infected exe).

“I was doing a random search in Bing and was noting that action were being done in the background wile Bing was loading . i was alerted by AV software an attack was trying to install files and I had no UAC pop up to tell me some was accessing or trying to access the system .”

Most antivirus alert the user when they detect that the site you visit is trying to exploit some security flaws to install malware, even if your system is not vulnerable and even if you are not at risk at all. Some people believe that the antivirus has prevented them from being infected, but it’s wrong, it has just alerted them that the site contains malicious code : for example, this code would try to exploit a flaw in quicktime although you don’t have quicktime installed, but you will still get the antivirus alert.

” the system crashed and I had to do a repair of the RC install and logs were lost or some deleted as I could not fine them but there were 2 new folder n the system 32 that were not part of the install and had nothing in them.”

that could be an unrelated crash, or it could be caused by malware you already had in your system (by installing a malware infected software) but whose activity was only detected later by your AV (with newer virus definitions).

“I did not see this happen in Firefox or Flock when testing only in IE 8 and on IE 7 in XP does the pre-load video in a search using Bing seem to have problems.. Files and Data are being search for when you are

using Bing. in IE8 on the Win7. it was even more aggressive in searching my system in some of my testing."

what do you mean by "files and data are being searched for when using bing"?
maybe you installed a malicious software that installed a BHO in IE. But IE is probably not the origin of your problem, as there are no know 0day critical flaw in IE, and even if there was any flaw, the protected mode should prevent any malware from writing on the hard disk.

Could you give us the link to the page where you pretend to have been infected?

"IE8 is something I would never use or tell my client to use but as we know many new users and other that just do not understand secuirty as we do will use Many end users are clueless to theses issues with Ms and how bad these holes are. I re-posting info to this site to Clients and media the more info we get out should get MS to do something I would hope"

actually, IE7 and 8 on vista and windows 7 are more secure that any other browser.

People who don't understand security use to install a lot of software from potentially malicious source (warez, unknown websites), and then complain Windows is unsecure or that IE is full of toolbars and crappy BHO/plugins.

June 19, 2009 at 9:10 am

James Butler

"Could you give us an exemple of something on windows that prompts the user for admin rights, but not on linux?"

Installing an application like Firefox or installing a printer come to mind, among many others. You should try it!

"that's much more complicated and time wasting than just clicking on "Continue" when the uac prompt shows up."

I believe this is precisely the problem with the Microsoft "multi-user" model. After a very brief period of time, the UAC becomes irrelevant as the vast majority of Windows users will just click OK regardless of what the prompt is telling them, because they are impatient to finish whatever they did that triggered the prompt. And with regard to the exploit/feature being discussed, the UAC IS irrelevant, as this

exploit/feature bypasses/disables UAC, anyway, so the UAC clearly is just a “security blanket” and does nothing serious with regard to system security.

“In fact this flaw in UAC cannot be exploited to run malware from the web. There need to be another flaw for this (or the user needs to lauch an infected exe).”

Read the article and followup notes. When this “feature” is exploited, the user will never know the web page they just visited dropped a rootkit onto their system.

“actually, IE7 and 8 on vista and windows 7 are more secure that any other browser.”

Do you have any data that supports that? ANY browser that does NOT access Windows core is more secure.

So, aside from “the users”, why can’t Microsoft follow the Linux example, technically? I imagine their developers are quite skilled at working within userland … or is it simply that you cannot build anything serious in win32 WITHOUT tapping into the Windows core?

Thanks, again. I really want to know why Microsoft is so resistant to following the proven security practices of every other major operating system out there.

June 19, 2009 at 10:10 am

“Installing an application like Firefox or installing a printer come to mind, among many others. You should try it!”

link8583

last time I checked, I needed root rights to setup Opera on linux ^^ When you install a program that is shared with all users (in program files), you need admin rights. But if you install a program in your user profile (as does chrome by default), you don’t need admin rights. You can install firefox in a folder you have write access to, and you won’t need admin privileges.

When installing a printer, you will need admin right to install drivers to the system, since it is a system wide action. But if you plug a printer whose driver is included in windows, you don’t need admin rights.

“I believe this is precisely the problem with the Microsoft “multi-user” model. After a very brief period of time, the UAC becomes irrelevant as

the vast majority of Windows users will just click OK regardless of what the prompt is telling them, because they are impatient to finish whatever they did that triggered the prompt."

yes, but even if they had to log out and type a password to log on as root to setup some crappy software, they would do so. After all, they can't live without that video codec that a random porn site suggested them to install…

"And with regard to the exploit/feature being discussed, the UAC IS irrelevant, as this exploit/feature bypasses/disables UAC, anyway, so the UAC clearly is just a "security blanket" and does nothing serious with regard to system security."

At his highest setting, UAC does protect the system, as malware can't gain admin privilege without the user approval.
But even at his default/medium setting, althoug this flaw allow malware to gain admin privileges silently, malwares can't auto elevate from Internet Explorer sandbox, so UAC is usefull for IE users, at least.

Nevertheless, the UAC highest protection setting on Windows 7 does protect the user from system wide malware installations.
However, nothing under linux, osx or windows protects the users from usermode malwares (which can send spam, steal credit card numbers, show unwanted ads, delete user files, do DDoS attacks, spread in IM clients …)

"Read the article and followup notes. When this "feature" is exploited, the user will never know the web page they just visited dropped a rootkit onto their system."

wrong, you misunderstood the article.
The flaw only allows an already running malware to elevate silently his privileges from usermode to administrator. This flaw does not allow an hacker to run a malware onto users PC. There need to be another flaw for this, or the hacker needs to convince the user to download an infected file and run it himself. Then the malware can get admin privileges and install as rootkit with no UAC prompt.

"Do you have any data that supports that? ANY browser that does NOT access Windows core is more secure."

yes, just read the previous comments.
IE and its plugins (flash, adobe reader) run in low integrity mode. A malware exploiting a 0day flaw in IE or its plugin will run in low integrity,

and thus cannot install in user mode (write forbidden), nor exploit the UAC flaw to gain administrator privileges.

IE doesn't run into the windows kernel, contrary to what some people think.
It is a user mode application that calls the same system routines as any other browser or other application.
The only difference is that some applications from Microsoft (including the windows sidebar) and third party editors use the IE engine to build their user interface, which makes the IE rendering engine a fundamental component in windows (that won't even be removed from Windows 7 E).
This does not have any security consequence, but this explains why you have to restart to install an IE update (because IE files can be in use at any time, even when IE is closed).

"So, aside from "the users", why can't Microsoft follow the Linux example, technically? I imagine their developers are quite skilled at working within userland … or is it simply that you cannot build anything serious in win32 WITHOUT tapping into the Windows core?"

I never have any UAC prompt in my day to day use of Windows Vista, except when some applications like adobe reader, flash, java or firefox tries to update themselves. They need administrator rights to update their own files, since their auto updater runs in usermode. That's not windows fault, these application could install a planified task that run once a day as administrator to search for updates and install them without prompting users for admin rights, but they don't… Maybe in the future they will, and you won't see any UAC prompt, except if you install a new software or change a system wide setting.

I am developper, and I never need administrative privileges in my applications (except one which changes the login screen wallpaper, but that's normal since it is a system wide setting). There is nothing missing in Windows that prevents developpers from writing usermode friendly applications.

"Thanks, again. I really want to know why Microsoft is so resistant to following the proven security practices of every other major operating system out there."

nothing prevents you from creating a limited user account to feel the same as if you were a limited user under linux or windows…
But in Windows Vista (or Windows 7 with uac at his highest level), you already have the same level of protection as a limited user account under linux or windows, except that it is much more easy to install

software or change a system setting since you just have to press continue instead of having to type a password. It is NOT less secure.

June 19, 2009 at 10:27 am

James, I'm not sure what you mean by "the Windows core."

Leo Davidson

All web browsers access the Windows API in pretty much the same way. IE ships with Windows, and components within IE like the HTML rendering engine are used by other programs (not just parts of Windows itself but third-party apps as well), but I don't think any of that poses a risk, except by its popularity. (I am not an IE fan, FWIW, but that's due to features and UI.)

link8583 is right in that this UAC stuff is not what gets exploited for malware to run from the web. UAC's job isn't to prevent things from running; it's to enable or prevent things from elevating.

Once code is running it could use the UAC flaw to gain higher access, but the flaw itself won't get the code on the box in the first place, it'll just allow it to do more damage or hide itself better if it gets on. IMO that's still important (if there's any importance to the difference between code running as admin and non-admin at all, which I believe there is). (Flaws which let code get on a box are *more* important, but when they keep being found it seems a good idea to try and limit what they can do.)

I also agree with link8583 that it's highly unlikely anyone is using this flaw in any current malware. It just wouldn't be worth it yet when machines running XP are such a rich target. If MS don't improve things by the time that Win7 is obiquitous then we might start to see things use this flaw to do more damage, in conjunction with some separate exploit or trojan, because they will have to target Win7 and will assume the default settings are the richest thing to attack.

People on both sides of this discussion seem to be confusing the UAC issue with one which allows code to get running on a box. The UAC issue isn't about that. UAC itself isn't about that, so a flaw in UAC cannot, by definition, have anything to do with whether or not code can get on a box. It's about what code can do once it's on the box. (e.g. Install as a rootkit, change security/firewall settings, infect other users, etc.)

June 19, 2009 at 4:04 pm

hd

@Leo
Am I to assume that pushing the slider up in the UAC control panel will solve this issue?

June 19, 2009 at 7:54 pm

Leo Davidson

@hd: Yes, at the Always Prompt level it's like Vista was by default.

Alternatively, for people who think everyone will click every prompt and there's no value in forcing elevation to wait for user consent and always trigger some kind of (possibly spoofed) prompt, setting UAC to Silently Elevate (for all apps) also makes sense.

But, IMO, the defaults make no sense from either point of view.

June 19, 2009 at 10:26 pm

Leo Davidson

There's an interesting post by someone who worked on UAC, Chris Corio, here:

http://www.withinwindows.com/2009/06/10/uac-uac-go-away-come-again-some-other-day/comment-page-1/#comment-3987

It's great of Chris to engage with us in this discussion with both background info and his own take on the current situation.

I've posted a response below his.

June 20, 2009 at 5:05 am

James Butler

Leo (and one more to you, below):

"All web browsers access the Windows API in pretty much the same way. IE ships with Windows, and components within IE like the HTML rendering engine are used by other programs (not just parts of Windows itself but third-party apps as well)"

Why is this? Why doesn't IE run discreetly? There is a huge difference between installing Firefox on Linux and installing Firefox on Windows in

that on Linux, Firefox brings its own code. It's true that it uses whatever graphics rendering engine is already being used by the system (i.e. cairo or whatever), however it does not need to access any system files to function. Your explanation extends my questions in that MSIE, itself, contains code that other, third-party apps need to use in order to function within Windows. Take away MSIE HTML rendering engine, and what breaks? Why is it necessary to rely on system-level dll files and whatnot in Windows? And describing the shared elements within MSIE as non-system-level files is simply not correct. In order for them to be shared, they must be accessible on a system-wide basis, which makes them "system-level" files.

This is the "Windows core": Any files required by all users. This is different from "shared" files that are simply permissive. If any app installed in userland is required to use a file but that restricted-rights user cannot delete or modify it, that file is a system-level file. Iexplore.exe is one of those files. Firefox.exe is not.

link8583:

"yes, but even if they had to log out and type a password to log on as root to setup some crappy software, they would do so."

Um … not exactly how it works, but it does show one of the issues with communicating to people who are unfamiliar with Linux. You don't need to log out, you 'su', and there's no profile switching, like there is in Windows, so the switch is nearly instantaneous … all of a sudden, you ARE the elevated user. Besides, the behavior you describe is advanced behavior that the majority of Windows users would not engage in, if they were forced to do so. That would help protect them, IMHO, and in the opinion of others here. The UAC prompt is there to provide a simple way for limited user identities to elevate their privileges so that a program that requires access to protected files is allowed to do so without the user needing to log out and log back in with elevated privileges ("Switch User"). The simplicity of that mechanism is part of its problem in that it becomes less of a security mechanism and more of an annoyance to the limited user, and will frequently cause them to simply acknowledge the prompt without understanding its implications, and possibly suffering from unintended exposure as a result.

If limited Windows users did not have the UAC, but simply received a notice that they needed administrator rights in order to execute the current action, there would be far fewer "accidental" installations of malware on Windows systems, because limited users would be *forced* to "Switch User", just like Posix users are … an inconvenience that would result in much greater protection for their system, as they are

forced to consider whether they really want to take the requested action. It's not just a "stupid thing … click" activity, it becomes a "oh man, do I really want to?" activity, which logically would result in fewer such actions being taken on a casual basis. And part of the problem is that it IS a much greater inconvenience for Windows users than it is for Posix users, because in Windows, switching users takes a minute or more as a second profile is loaded into the system, along with additional desktops and other user-related elements, where in a Posix system it takes just a moment for the su to occur, and only a moment to switch back when the task is accomplished.

This is completely compatible with the topic of this discussion which questions the efficacy of the UAC as it is currently implemented within Windows 7. Quoting from the article:

"RCE by itself warrants some attention, but with the introduction of default UAC policy in Windows Vista, the potential impact of RCE vulnerabilities were actually reduced because the malicious code can no longer assume full administrative privileges, instead, limited to what the target application was running which in most scenarios was medium-level or even low-integrity like in Internet Explorer. However, in conjunction with the default Windows 7 UAC policy and this vulnerability, the potential impact of RCE vulnerabilities is raised, as the malicious code executed could silently elevate itself to have much more free reign over the system than before. If this isn't enough indication that the default security policy in 7 is worse than Vista, than I don't know what is."

And:

"UAC prompts be only a waste of their time. I say this because if some applications can elevate themselves with or without the user agreeing to a prompt, the prompt's effects are nullified. To look at it in another light, at the default Windows 7 UAC policy, it's as good as having UAC prompts turned off entirely."

So the UAC in Vista was acceptable to the author in limiting the impact of RCE attacks, however with Windows 7's default UAC policy, "the potential impact of RCE vulnerabilities is raised" and "it's as good as having the UAC prompts turned off entirely". Does this not indicate that the UAC as implemented in Windows 7 is a badly implemented security tool?

Continuing the quote from the article:

"Secondly, besides the obvious malicious use for the UAC vulnerability, there is nothing stopping it from being abused by legitimate developers

and their applications. After suggesting such a scenario in my original article, one such developer have already expressed interest in using this vulnerability in such a way that will remove UAC prompts from his applications."

When any developer can simply disable the UAC, then it is irrelevant. It is no longer a system protection device except in the most superficial terms.

Leo (quoted from your posts above):

"Low-integrity IE doesn't help with Flash vulns. because Flash is still hosted inside a medium-IL proxy process for compatibility."

So the fact that MSIE is a low-integrity process means nothing when it hosts medium-IL processes, if those processes can execute potentially dangerous code.

And:

"..it's now so easy for code to silently and immediately bypass the UAC prompts and jump from the limted-admin token to the full-admin token."

Which seems to indicate that your testing shows potential exploitation of a vulnerability through a rigged app that runs at medium-IL within the low-intensity MSIE process … like a rigged Flash movie. Which seems to indicate that simply visiting a rigged web page could silently execute dangerous code without a UAC prompt.

And, without any malice toward Microsoft products, this is not an issue with Linux because there is no UAC … there is simply a notice that the user must intentionally and explicitly allow the execution of the code by taking the extra security step of deliberately switching user (su) to an elevated account.

If the Windows 7 process in its default setting were not so easily disabled/bypassed by any developer that so wished, it might be a convenience that worked well. As it is, it is an INconvenience that is irrelevant to the security of the system, GIVEN THE TOPIC OF THIS DISCUSSION. (Trying to stay in context, here.) The author agrees, as did you, Leo.

Please forgive me if all of these statements have led me to become confused about the issue. It seems quite straightforward. If Microsoft wanted a multi-user security model that has been proven effective for decades, why is it so tough to start with the Posix model?

Thanks for your patience.

June 20, 2009 at 6:45 am

“Why is this? Why doesn’t IE run discreetly?”

link8583

What would be the benefit to have an IE installed the same way as Firefox on Windows 2000/xp/vista/linux or IE2/3 for Windows 95? There would be NO benefit on security.

“There is a huge difference between installing Firefox on Linux and installing Firefox on Windows in that on Linux, Firefox brings its own code. It’s true that it uses whatever graphics rendering engine is already being used by the system (i.e. cairo or whatever), however it does not need to access any system files to function.”

Firefox calls system and C runtime APIs, so indirectly, there is system files needed for it to run, as for any software. And if you don’t have an X11 server installed it will not run.
IE has not fundamentaly more “system files requirements” than Firefox or any application to run (it calls GDI, Win32 and C runtime apis. It does not interact with the kernel and it doesn’t have any component running in the kernel).

“Your explanation extends my questions in that MSIE, itself, contains code that other, third-party apps need to use in order to function within Windows. Take away MSIE HTML rendering engine, and what breaks?”

explorer.exe (file explorer, control panel) relies on IE engine to display parts of some windows. Of course, this runs as user mode.
The windows sidebar uses IE engine to run and draw the gadgets.
In fact, more than a third of all applications available for Windows need Internet explorer engine to run.
Ex:
ATI driver installation program uses IE to display some nice but unusefull pictures during install.
Microsoft CHM help viewer uses IE engine to show help pages.
Many C/C++/.net based programs uses IE to draw rich user interfaces because it is easier than using directly GDI.
Even Google update (which can be used to install chrome) uses IE to draw it’s html based user interface!
Some bittorrent clients, download managers, rss readers, Eclipse IDE (java based development enrionment), tens of thousands popular applications need IE engine to run properly.
Think of IE engine as a library that any developer can use.

“Why is it necessary to rely on system-level dll files and whatnot in Windows? And describing the shared elements within MSIE as non-system-level files is simply not correct. In order for them to be shared, they must be accessible on a system-wide basis, which makes them “system-level” files.”

yes, components of IE are part of every version of Windows since windows 98, thus, IE rendering engine files are system components that are guaranteed to be always present in windows, even in Windows 7 E.

“This is the “Windows core”: Any files required by all users.”

At the time of windows 95, IE was not included in windows, but many application began to require IE4 to be installed to run.
This doesn’t make IE a “windows core component”, as Windows base services don’t need IE to run.

“This is different from “shared” files that are simply permissive. If any app installed in userland is required to use a file but that restricted-rights user cannot delete or modify it, that file is a system-level file. Iexplore.exe is one of those files. Firefox.exe is not.”

by default, firefox installs in program files, and users can’t write or remove files in firefox install. So, does firefox becomes a system file in this situation?

“Um … not exactly how it works, but it does show one of the issues with communicating to people who are unfamiliar with Linux.”

Linux was my main OS 9 years ago… So, I’m pretty familiar with it.

“You don’t need to log out, you ’su’, and there’s no profile switching, like there is in Windows, so the switch is nearly instantaneous”

exactly like the runas /noprofile command included in windows since NT4.
But if you type the admin password in an user session, a malware could potentially read it by spoofing your terminal Window. (very easy)
It’s less secure than logging out and logging in as root.
Even worse, if you use SU to setup a package downloaded and stored in your user profile, if you have a malware running in your session, it can infect the rpm/deb file to gain root privileges when you install it. This problem also exists on Windows.

“The simplicity of that mechanism is part of its problem in that it becomes less of a security mechanism and more of an annoyance to the limited user, and will frequently cause them to simply acknowledge the

prompt without understanding its implications, and possibly suffering from unintended exposure as a result."

if users ran a linux distribution that told them 'access denied' when they want to setup an application, of change a system setting (as it was the case until windows vista when running as Limited User Account), then they would run as root to avoid these "limitations", and it would be even worse. So uac is a better solution. In fact, many linux distributions use a similar mechanism to elevate users rights when launching a system control panel.
I've even seen a netbook at a store that ran a linux distrib as root by default, because it is "easier" to use this way…

"So the UAC in Vista was acceptable to the author in limiting the impact of RCE attacks, however with Windows 7's default UAC policy, "the potential impact of RCE vulnerabilities is raised" and "it's as good as having the UAC prompts turned off entirely". Does this not indicate that the UAC as implemented in Windows 7 is a badly implemented security tool?"

The problem only resides in the default UAC configuration which can allow malwares to elevate silently.
This is not possible when the UAC settings are changed to the maximum level of protection (the same as the default settings on vista).

"When any developer can simply disable the UAC, then it is irrelevant. It is no longer a system protection device except in the most superficial terms."

That's the problem of this flaw. But it is not exploitable when uac is set to maximum efficiency.
But even in the default settings, UAC still protects Internet Explorer users, as malwares can't auto elevate using this UAC flaw from a potentially vulnerable iexplore.exe process. In this case, a malware could not even write to the user profile to install an usermode malware.

"So the fact that MSIE is a low-integrity process means nothing when it hosts medium-IL processes, if those processes can execute potentially dangerous code."

read my answer about that. Leo was wrong, Flash and Adobe reader run as Low-IL inside iexplore.exe process. Thus, UAC protects the user from flaws in these plugins.

"Which seems to indicate that your testing shows potential exploitation of a vulnerability through a rigged app that runs at medium-IL within the

low-intensity MSIE process … like a rigged Flash movie. Which seems to indicate that simply visiting a rigged web page could silently execute dangerous code without a UAC prompt."

no, since flash runs inside IE security context, as Low IL.

"And, without any malice toward Microsoft products, this is not an issue with Linux because there is no UAC … there is simply a notice that the user must intentionally and explicitly allow the execution of the code by taking the extra security step of deliberately switching user (su) to an elevated account."

I suppose that depends on the distribution you're using.
On ubuntu, there are prompts similar to UAC when you launch system control panels.
As I said before, if mainstream users were to use linux, they would run as root because they couldn't stand having to logout to install new crappy unusefull software. That's a sad fact. At the moment the users knows about the root account, they will think it is better than the limited user account, because it is less annoying.

"If Microsoft wanted a multi-user security model that has been proven effective for decades, why is it so tough to start with the Posix model?"

Windows is not less secure than any posix system.
In fact Microsoft knows how Unix works, because they used to develop and sell an Unix based OS called Xenix in the 80's.
Then they built OS/2, then Windows NT with the same security features as any unix system.

The only problem is that by default, for making administration easier, every Windows NT based system creates an administrator account by default.
Since users almost always use the default configuration, everybody ran and is still running as administrator (and most developpers assumed it is always the case, thus making their application often not compatible with limited user accounts), althought limited user accounts exists since 1993 in Windows.
Windows is not less secure than Unix/Linux in any way. The same fundamental user/security concepts are implemented in Windows and Unix. The only difference is the OOBE.

June 20, 2009 at 7:26 am

James Butler

link8583:
'then they would run as root to avoid these "limitations"'

Again: You CAN'T log in and run the system as root on most modern Linux desktops.
You MUST su from a limited account if you need root.
You CAN (and most DO) run as a full-on Administrator in Windows.
THAT is a HUGE difference in security models, wouldn't you agree?

I think you are the only person here who is not advocating that Microsoft needs to change their basic security model so that by DEFAULT, a Standard User is the primary identity. Even Microsoft techs are saying that, but they are unable to implement it, according to Leo's link, above.

If you're going to argue that MSIE 2&3 under Windows 95 (!) proves that Windows 7's implementation of MSIE 8 means that MSIE 8 is not a system-level program, then I've got nothing to say to you. If MSIE 8 were not a system-level program, then it could be easily and completely removed, just like any userland app. You're not making sense … you're just trying an argument for the sake of defending Windows. Your comments about installation and removal of Firefox seem remarkably uninformed. Have you TRIED deleting/renaming/uninstalling Firefox as the limited user who installed it in Linux? It's quite different from doing the same thing in Windows.

Frankly, your claims of having used Linux as your "main OS" seem exaggerated. You reinforce my point when you ask about Firefox installing in Program Files within Windows … a situation ONLY made possible by the fact that you MUST install Firefox as an administrator in Windows! Where did you install your programs when Linux was your "main OS"? I can assure you that it was NOT in a protected directory, unless you mistakenly became a root-level user and installed it into a protected directory while using that identity. Becoming root is ONLY required for the VAST majority of Linux app installations ONLY during the final build process, when protected libraries MUST be accessed in order to complete the build specific to that system. EVERY other part of the installation process is EXCLUSIVE to userland, and root is NOT required to uninstall an app unless you made the mistake of installing it while you were root.

I am not defending Linux. I am simply asking if anyone here can tell me why Microsoft has chosen to continue their efforts at creating a security model that completely ignores the success of the Posix model. I want to understand the limitations of the platform that have forced MSFT to

take this route, so that I can better explain to my clients why things are the way they are. When those who frequent this thread AND at least one of the developers of the security widget in question IN THIS ARTICLE all admit that it's a broken model, I think my question is reasonable.

I have already been attempting to answer queries from my clients about Windows 7, MSIE and the UAC, and the only answer I have for them at the moment is, "It's like that because Microsoft is trying to improve the security of their products." This is a lame explanation, and does nothing to satisfy my actually intelligent clients who are considering moving from Linux to Microsoft.

If you really are a Windows advocate, beyond posting superfluous arguments about irrelevant operating systems, then give me something correct and immediate to support Microsoft's position. Otherwise, it's easier and more true for me to say, "You are already using the best security model available, and Microsoft will need to start from scratch to overcome their security issues because they have been on the wrong track for the past 10 years."

Why CAN'T Microsoft use a similar multi-user model? What's stopping them?
This is a basic question about a serious issue.

Sorry about the long posts. Thanks, again for any SERIOUS attempt to answer a TECHNICAL question.

June 20, 2009 at 8:27 am

Leo Davidson

James, everything link8583 says is reasonable and I also agree with most* of it.

(*I'd say "all" but I haven't re-read every word to confirm that.)

I don't think he's advocating anything as if it was a perfect solution, he's just explaining how things are.

Whether or not IE is system-level is a matter of semantics. It depends whether you consider the user/application-level components that come with Windows as a part of the OS or just something that comes with the OS. Technically, you can boot Windows without IE or Explorer. A lot of software won't work because a lot of software assumes those components are available, but they are not inherently required by the OS.

Component re-use is a GOOD thing, as is an OS that comes with lots of useful components. There's no sense in every developer re-writing code that already someone has already written and is willing to share, unless a significantly different component is wanted. I'm glad somebody wrote Firefox as I prefer it to IE but I'm also glad that when I just want to display a bit of HTML in an About window in my own app I can use the MSHTML component (part of IE) to avoid having to spend six months writing my own HTML rendering engine. I'm also glad that I can depend on MSHTML being on people's machines already and not have to tell my users to download some extra thing or include that large component in my own app's installer.

"Becoming root is ONLY required for the VAST majority of Linux app installations ONLY during the final build process, when protected libraries MUST be accessed in order to complete the build specific to that system."

That is something many people wish MS would improve. They cannot do much about existing installers as there's no way to know what rights an installer — which is just an exe file really — will need. But it would be nice if new installers could say "I just need to copy these files to my own folder in Program Files" and be given rights to do just that, rather than full admin rights. Eventually we'd expect all installers to declare what access they needed and be more suspicious of ones which didn't.

For example, I'm still pissed off that installing Daemon Tools installed, without my knowledge or consent, a root certificate authority which meant that I was allowing some third party I knew very little about to vouch for code signatures on stuff I downloaded. I don't want any installer to have those rights without me knowing about it, but as it stands today all installers get those rights.

(It's not that I think the people behind that root CA will do something malicious, but I know nothing about the company and how well they protected that certificate from being stolen. Most of all I resent the fact that some extra root CA was put on my system to, as far as I can tell, save some company the small cost of getting their code signed by one of the main CAs like everyone else does.)

June 20, 2009 at 8:59 am

James Butler

Thank you for the response, Leo. While I don't believe that link8583 has provided correct information much of the time, I respect your right to agree with it.

link8583 is defending Windows the way it is, and is not saying WHY, except that “it is easier for the user” … which also, in this case, means it is less safe. link8583 brings up one particular Linux distro, Ubuntu, when describing behavior similar to the UAC, in that under recent versions of Ubuntu, one is able to su without launching a command prompt. Other distros also use this feature.

But this is not to be confused with the underlying security model. Windows uses a split token. Linux uses separate tokens. The split token is vulnerable in some situations simply because it is a single entity. So my question comes down to: What is so valuable about the split token that Microsoft believes, despite continuing evidence (this article’s proof of concept, among many others) that it is a worse security model than the one used by Posix systems?

Why reinvent the wheel? (as you noted):

“There’s no sense in every developer re-writing code that already someone has already written and is willing to share, unless a significantly different component is wanted.”

I agree with that, while still thinking that MSIE is, in fact, a file manager that is able to access url-defined locations rather than a “web browser”, and, as such, is not in the same class as a “web browser”, and as such is less secure than a “web browser” should be. But in Microsoft’s engineering case, they created a big security hole when shared files are system files … because when a system file gets compromised, it exposes the system. When a userland file is compromised, that only affects the userland … not the system.

In the case of the Microsoft multi-user security model, it is continuing to be proven that what they have been doing is adding more layers on top of their system rather than simplifying and returning to a model that is demonstrably more secure and only moderately more effort for the end user. (Every process takes training. No computer use is “intuitive” until a person has already developed a relationship with it.)

The Chris Corio article you linked to indicates that he (and probably other MS developers) WANTED to keep the UAC as a security model, but could not as it was proven to be unreliable for security purposes, even as they are proud of what they have done with the split token. Maybe its a political thing … Microsoft doesn’t want to admit that they have been pursuing a flawed model or something. I don’t know. But I DO know that I have yet to see any technical explanation as to WHY the Posix multi-user security model was not used, or at least why Microsoft insists on continuing to follow their current path.

If the answer is not available, that's fine. It is a question which I have often been asked, and I have yet to be able to answer it, so if I still can't answer it, so be it. Thanks, again.

June 20, 2009 at 9:14 am

Sorry ..

James Butler

"What is so valuable about the split token that Microsoft believes … that it is a BETTER security model than the one used by Posix systems?"

(To much crap … not enough continuity. 😉)

June 20, 2009 at 9:16 am

One more try … although I hope my point was already made …

James Butler

"What is so valuable about the split token that Microsoft believes, despite continuing evidence TO THE CONTRARY (this article's proof of concept, among many others), that it is a BETTER security model than the one used by Posix systems?"

(Done.)

June 20, 2009 at 9:42 am

"But this is not to be confused with the underlying security model. Windows uses a split token."

Leo Davidson

Windows doesn't have to use a split-token account, it's just the default.

I think the reason Windows doesn't make (or at least encourage) people to use Standard User accounts is that they'd be too annoying to use right now for the average home user. The reaction to Vista's UAC prompts was bad enough; making people put up with the same prompts where they have to type a password every time would have terrible results. MS have a lot of work to do in this area, IMO, if they seriously want home users to switch to Standard User accounts one day.

Big corporate Windows installs will almost always make everyone use Standard User accounts, though. Windows can do it, and home users can choose to work that way if they want. It's not the default, out-of-the-box setup but it is completely possible. (The PC that runs my TV runs with a Standard User account, for example.)

The split-token stuff was new in Vista. Before that the NT-based OS (e.g. NT3.51, NT4, Windows 2000, XP) didn't have such a thing; they had Standard User and Administrator (without the split token). (Vista and Windows 7 still have those account types, but they're not the default.)

With XP and earlier OS, almost all home users ran as admin and so a lot of software aimed at (or written by) home users assumed admin rights. MS rightly wanted to change that but wanted to do so in a way which wasn't too inconvenient and didn't break too much existing software by default, so the split-token stuff was created with, I presume, the intention of not requiring a password to be entered to switch from non-admin to admin.

When Vista came out people had to run a lot of their apps as admin (that is much better now) and people wouldn't have put up with typing a password every time they wanted to start an app. The split-token account was a pretty good compromise, I think.

Also note that with a standard user account, if you still use elevation then things can slip through from standard user to admin. If you use elevation at all then there is always that risk. The isolation of the two modes is better with standard user but it is not (and cannot be) absolute.

"I agree with that, while still thinking that MSIE is, in fact, a file manager that is able to access url-defined locations rather than a "web browser", and, as such, is not in the same class as a "web browser", and as such is less secure than a "web browser" should be."

I don't know what you're getting at there, sorry. I don't see why the category a program falls into affects how secure that program is.

"But in Microsoft's engineering case, they created a big security hole when shared files are system files … because when a system file gets compromised, it exposes the system. When a userland file is compromised, that only affects the userland … not the system."

IE lives in the userland. IE isn't part of the kernel.

I'm not sure exactly what you mean by "compromised" there but if a component of IE is compromised it is no different to a component of Firefox, or some shared library like gzip, or whatever, being compromised.

June 20, 2009 at 10:36 am

James Butler

"I don't know what you're getting at there, sorry. I don't see why the category a program falls into affects how secure that program is."

A "web browser" SHOULD NOT require hooks into the system files. It is an extremely exposed, public face and needs to be kept as separate from the underlying OS as possible. Internet Explorer provides many system files, as you noted with regard to the HTML rendering engine, for example, and is therefore less of a "web browser" and more of a "system" application. I mentioned this in response to your statement, "I'm also glad that I can depend on MSHTML being on people's machines already and not have to tell my users to download some extra thing or include that large component in my own app's installer." If the HTML rendering engine were NOT part of MSIE, that would be great … but it is an integral part of MSIE and that changes things quite a lot. MSIE is a special class of application and provides many system-level files like the HTML rendering engine. This is one of the many reasons why it has been difficult for Microsoft to remove it completely from the system … it IS a part of the OS, and not simply something that CAME with the OS.

When a system-level application is compromised, it is far more dangerous to the system than when a userland application is compromised, as long as userland/system segregation is enforced.

"IE lives in the userland. IE isn't part of the kernel."

"Userland" does not include any system files. And there is a big difference between the kernel and the rest of the files that make up an OS, including the rest of the system-level files, of which the kernel is one … on non-Windows systems. (Have you ever tried replacing the kernel on Windows with a newer version? Just the kernel, not a whole upgrade pack? Try it. Works in Linux.)

It is common to "compromise" (exploit a vulnerability that leads to a successful attack on) any computer by using a vector into system-level files and processes, and, since you brought it up, under Windows there is an excellent chance of compromising the kernel, too, because other

system files tie into it so heavily. (Another big difference in security models.)

A buffer overflow attack is a typical example. This is exactly the reason why MSIE is a low-priority process. It's a sort of additional protection after so many attacks got through to the system level in earlier versions. It is trivial to overflow a program that accesses web pages by using something as simple as a rigged JPEG image, including MSIE and Firefox, and if that program were privileged enough, then it could easily be used to "compromise" the system. MSIE does not run exclusively in userland, and on Windows, neither does Firefox, due to the reuse of code that you mentioned earlier. It's cool with me if we blame that one on lazy Firefox developers … I'm not particular. However it is possible that building a web browser for Microsoft REQUIRES system-level files, because there is no alternative provided or allowed, so that would let a few developers off the hook.

Under Linux, if Firefox were compromised by an exploit, then the worst that could happen is that that user's files could be modified by the attacker and/or the attacker could add/remove stuff in that user's restricted space. "del C:*.* /y" simply won't work. To repair even the most screwed up profile requires removing the compromised userspace and establishing a clean one. The rest of the system remains untouched. Compromising the MODERN kernel is unheard of. (Find a validated example regarding a 2.x Linux kernel and I'll eat my words with mustard.)

Compromising the Windows kernel is not only heard of, it's how the most recent Windows rootkits are evading detection. Hell, they can even get through to the BIOS on a Windows system! (Don't get me started on why BIOS are still being used … 🙂 You're with me on that one, right?)

Also don't get me started on how Windows is compromised so often by viruses and rootkits and other malware because it is so popular. That is one reason, definitely it's a big target, but if that were the primary reason, then Linux would be experiencing similar rates of compromise, as it is the number one platform for web servers in the world and you'd better believe that hackers are very anxious to penetrate them. Not to gain access to the databases and whatnot stored on them, which happens all the time due to poor application design, but to gain access to the SYSTEMS from where they could both hide and control a far greater range of activity. The difference comes down to the security model.

This all speaks to two issues: (1) The underlying multi-user security model and (2) how the UAC is a mechanism that seems to provide

security but (as Chris Corio noted) is not capable of providing security.
And yet the UAC is used every time elevation is requested.

Anyway, I'm sure it will be fine. Windows 7 will come out, people will buy it, security will be an issue, patches will be the result, life will go on.

I don't think an answer to my question is available, yet.
Thanks for the article and comments. Always interesting. Bye bye.

June 20, 2009 at 12:47 pm

link8583

"You MUST su from a limited account if you need root.
You CAN (and most DO) run as a full-on Administrator in Windows.
THAT is a HUGE difference in security models, wouldn't you agree?"

There's no difference between using runas /noprofile on Windows and using su on linux.
It has the same security implications, and the same result.
But UAC on windows is easier to use than runas, and it does the same thing when called from a limited user account, with no security issue.

Of course if you use runas or su to install an application you just downloaded and stored in your userprofile, if a usermode malware is running, then you are giving it root privileges since it can have infected the newly download application setup package.

"I think you are the only person here who is not advocating that Microsoft needs to change their basic security model so that by DEFAULT, a Standard User is the primary identity."

I'm not advocating Microsoft, I'm just explaining why things are currently this way, and why it is not that bad, since there are ways to work securely in Windows (by using a limited user account).
There's something that people don't understand about Microsoft.
Security is one of their priority, but their most important priority is retrocompatibility and productivity, because that's all their customers care about.
They can add security features, but if the customers don't want to use them, of if the software they need don't work with these features, then the customers won't be happy and will stay with an older OS, or disable some security features.
That's what happened with companies staying on windows xp or home users disabling UAC.
And that's why microsoft stills creates administrator account by default, and is now lowering UAC efficiency.

“If you’re going to argue that MSIE 2&3 under Windows 95 (!) proves that Windows 7’s implementation of MSIE 8 means that MSIE 8 is not a system-level program, then I’ve got nothing to say to you.”

I think you did not understand my point.
I was just saying that firefox and IE2/3 for windows 95 are installed the same way: just a bunch of files in a program files subfolder.
Since IE4 however, IE has its components more deeply installed, in windows system folders, but this does not makes it less secure than firefox or IE2/3 on Windows NT4. When IE runs, it always runs with the same privileges as the current user (except on vista and 7 where IE has actually less privileges than the current user), no matter where its dll or exe are stored.

” If MSIE 8 were not a system-level program, then it could be easily and completely removed, just like any userland app.”

IE does not run as a service, it is not part of the windows core, and it is not used by the kernel. It runs in userland. It is part of the windows APIs that are always distributed with windows since win98.
Microsoft could remove IE, but explore.exe would need some modifications to continue working, and a third of the applications built for windows would not run anymore.

Do you know about gdiplus.dll ? that’s exactly the same question as IE. This dll is provided with each version of windows since windows ME, and removing it from windows would be disastrous since more than 50% of the current applications need it to run. It could be added on Windows 98, but since it is no more supported, gdiplus.dll doesn’t ship anymore with third party program installers (but it used to for a long time). Thus, programs would stop working on Windows 7 if this dll was not included, since it is no more included with most program installers.

IE and gdiplus.dll run in userland, but they cannot be removed from windows since they are part of the windows apis and libraries.

Since they both ship with windows, you can say they are system level components, but they run with the same level as the current user, and windows can start without them (but not explorer.exe)

“Your comments about installation and removal of Firefox seem remarkably uninformed. Have you TRIED deleting/renaming/uninstalling Firefox as the limited user who installed it in Linux? It’s quite different from doing the same thing in Windows.”

It's not.
If you installed firefox for all users on linux or windows, you can't alter its files with a limited user account.
If you installed firefox in your home directory on linux or your user profile on windows, you will be able to alter its files.
YOU seem "remarkably uninformed".

"Frankly, your claims of having used Linux as your "main OS" seem exaggerated. You reinforce my point when you ask about Firefox installing in Program Files within Windows … a situation ONLY made possible by the fact that you MUST install Firefox as an administrator in Windows!"

You're wrong.
Firefox on windows can be installed in the user profile. No admin right needed.
In fact you can install many application with no admin rights. Just disable the installation program detection heuristic in windows vista (using GPO) to avoid UAC from prompting for unneeded admin rights, and give the right to create subfolders in program files to every user. Most applications will install successfully with no admin rights. Maybe you need to also give folder creation permission to limited users in the "all users" profile.

"Where did you install your programs when Linux was your "main OS"?"

typically in /usr/bin, but the deb/rpm packages decides where the software is supposed to be installed.
Most of the software I used to use on linux needed root rights to setup (oracle, java, development tools, …).

"I can assure you that it was NOT in a protected directory, unless you mistakenly became a root-level user and installed it into a protected directory while using that identity."

when you use mainstream distributions, you typically install software from the distribution repository, and you need to give root rights to the package manager, which installs the packages for all users.

"Becoming root is ONLY required for the VAST majority of Linux app installations ONLY during the final build process, when protected libraries MUST be accessed in order to complete the build specific to that system."

that's not clear. By accessed, you mean written? And you're saying that the vast majority of linux app need root rights after being compiled for

their installation to shared folders to complete. You're in contradiction with your earlier bashing about windows app that need admin rights to install.
And a usermode malware can gain root privileges in this process.

"I am simply asking if anyone here can tell me why Microsoft has chosen to continue their efforts at creating a security model that completely ignores the success of the Posix model."

with 0.5% of market share, the "posix model" is not a success.
Enterprises and users rarely use linux on client side, hence the lack of interest for malware developpers who need to concentrate on the default target: Windows XP with user running as Administrator (and often with no security patches). There are exactly the same security threats on linux and on windows. Nothing prevents a malware on linux or windows to run in userland. And once the user gave root rights to an installer, it can do anything on the system. Look at OSX users getting infected with malwares coming with a pirated copy of iLife and photoshop CS4.

"I want to understand the limitations of the platform that have forced MSFT to take this route"

the limitation does not resides in the platform.
It resides in the fact that some poorly written applications need administrator rights because they write their settings and data in their installation folder (subfolder of program files, write protected).
It's exactly like if linux software developpers decided to write their applications data in /bin or /usr because by default, linux users always run as root (it's just an example, it is not the case), and then complaining that linux has no security features.

" so that I can better explain to my clients why things are the way they are. When those who frequent this thread AND at least one of the developers of the security widget in question IN THIS ARTICLE all admit that it's a broken model, I think my question is reasonable."

trying to reduce the steps to do administrative tasks the way the default settings of UAC does, is a broken model.
But the windows limited user account model that is there since windows NT 3.5 (1993) is NOT broken. It works the same way as linux/unix and is as secure.
The only problem is that most programs ignored it because users were administrator by default on their workstation. But the situation is changing thanks to UAC, which forces developpers to write usermode

friendly applications, which store their data in the user profile, not in program files anymore.

“It’s like that because Microsoft is trying to improve the security of their products.” This is a lame explanation

security has been there for a long time.
The only problem is that windows still creates administrator account by default, AND that users still use this account by default because they don’t care about security, and don’t know what is a user account.
This is a choice made to make the use of windows easier (no security prompts until vista and uac), and application compatibility better.

But many business run successfully their windows nt4/2000/xp/vista systems with limited user accounts, once they asserted the compatibility of all their applications.

“Microsoft will need to start from scratch to overcome their security issues because they have been on the wrong track for the past 10 years.”

the only wrong decision with windows NT has been to create an administrator account by default, because users don’t know about security, and they always use the default configuration (which is the default administrator user created by windows installation). Then, the developpers have targetted the default configuration, building applications that write in “program files”, making them incompatible with limited user account, just because of this stupid mistake.
But the Windows NT security model is as secure as the linux model you’ve been touting. You just have to decide to use it by not running as administrator.

“link8583 is defending Windows the way it is, and is not saying WHY, except that “it is easier for the user” … which also, in this case, means it is less safe.”

it is less safe IF users run as administrator.
but it is safe is users run as limited users.

“I agree with that, while still thinking that MSIE is, in fact, a file manager that is able to access url-defined locations rather than a “web browser””

???
what does ie has to do with a file browser??

“, and, as such, is not in the same class as a “web browser”, and as such is less secure than a “web browser” should be.”

under windows xp, explorer.exe windows can be changed as IE6 windows seemlessly, because of the activeX magic. But that does not make IE a file browser, nor does it makes your files accessible on the web. This mechanism does NOT makes IE less secure.

“But in Microsoft’s engineering case, they created a big security hole when shared files are system files … because when a system file gets compromised, it exposes the system. When a userland file is compromised, that only affects the userland … not the system.”

You’re completely WRONG about that!
But this explains many of your misunderstandings 🙂

On windows/linux/osx, when you run a program, it runs with the same right as the current user, no matter if this program is located is system folders or in the user profile (except on linux with programs that have the suid flag set, in which case they run with the privilege of their owner, often the root account)

So, when a program using system files is compromised (it is compromised at execution, with the current user privileges, not at the filesystem level), it does not compromises the whole system, just the user profile.

“Maybe its a political thing … Microsoft doesn’t want to admit that they have been pursuing a flawed model or something”

It’s not.
It’s a retrocompatibility decision, because retrocompatibility is the number 1 reason why windows still have 90% market share.
Microsoft does not want to lower retrocompatibility by creating limited user account by default, because they know that users will still give admin rights to virus infected applications they downloaded from some unknown websites, and get as much infected as if they ran as administrator directly.
So, it’s not worth the pain to reduce application compatibility by enforcing default security policy.
It would be exactly the same with linux. Users always give root rights to applications they want to install… after all, how can they resist to a free screensaver that an ad just suggested them to install?

“I don’t know. But I DO know that I have yet to see any technical explanation as to WHY the Posix multi-user security model was not used, or at least why Microsoft insists on continuing to follow their current path.”

The multi user model is used, by many windows users. Just not those who don't feel concerned about security.
Microsoft can't force the user to change their habits and break legacy application compatibility, or they risk losing money.

"If the answer is not available, that's fine. It is a question which I have often been asked, and I have yet to be able to answer it, so if I still can't answer it, so be it."

I think I've answered that question. Let me know if something is still unclear.

June 20, 2009 at 1:37 pm

link8583

"It is an extremely exposed, public face and needs to be kept as separate from the underlying OS as possible"

You're wrong.
The fact that the IE engine is included in windows and is used in many programs is actually more secure, because if there is a flaw in IE, there need to be only one update, provided by windows update, to fix the flaw in the IE engine for every program that use it. If IE was not included in windows, software developpers would have to use another rendering engine, and it would be their responsability to issue updates when this rendering engine suffers from a security flaw. And most developper don't care about this kind of flaw…
With the mozilla approach, each program that needs gecko engine includes the whole gecko engine. Then, when a flaw is discovered in gecko, every application that uses gecko must be updated, making the customers more exposed to the flaw. Look at thunderbird, it is often updated one week only after Firefox when there are flaws discovered in Gecko.

"Internet Explorer provides many system files, as you noted with regard to the HTML rendering engine, for example, and is therefore less of a "web browser" and more of a "system" application."

Think of IE engine as a shared library. That does not make it more dangerous than firefox.

"When a system-level application is compromised, it is far more dangerous to the system than when a userland application is compromised"

as I said earlier, that's absolutely wrong.
IE does not run with higher privileges than firefox. It's even the opposite since windows vista. Firefox has more privileges than IE, and can do more damage if it is exploited by a malware.

"It is common to "compromise" (exploit a vulnerability that leads to a successful attack on) any computer by using a vector into system-level files and processes, and, since you brought it up, under Windows there is an excellent chance of compromising the kernel, too, because other system files tie into it so heavily. (Another big difference in security models.)"

wrong again.
You confused system services (on windows) or daemons (on linux) with system files and processes that are run within a user account context.
Exploiting a windows service give you the same privileges as the service.
In the case of IIS7 webserver, you will get low privileges and won't be able to do much damage.
You won't get any admin rights by compromising IE. You won't even get the current users rights, since IE runs in Low IL.

"A buffer overflow attack is a typical example. This is exactly the reason why MSIE is a low-priority process."

integrity, not priority

"It's a sort of additional protection after so many attacks got through to the system level in earlier versions."

you're wrong again.
Attacks under IE on windows xp used to go through the system level because most users ran as administrator.
Firefox users running as administrator also face the same risk, even if firefox is stored in the user profile : if this user has admin rights, once firefox is exploited by a malware, it gets admin rights.

"It is trivial to overflow a program that accesses web pages by using something as simple as a rigged JPEG image, including MSIE and Firefox, and if that program were privileged enough, then it could easily be used to "compromise" the system."

indeed, but not on windows vista and current linux kernels thanks to ASLR and DEP.

"MSIE does not run exclusively in userland, and on Windows, neither does Firefox, due to the reuse of code that you mentioned earlier."

IE and firefox, on windows xp, run with exactly the same privileges. In limited user account, they can't harm the system if they get exploited.

"However it is possible that building a web browser for Microsoft REQUIRES system-level files, because there is no alternative provided or allowed, so that would let a few developers off the hook."

that's again completely wrong.
Where did you get that idea that firefox or IE runs with some special privileges??

"Under Linux, if Firefox were compromised by an exploit, then the worst that could happen is that that user's files could be modified by the attacker and/or the attacker could add/remove stuff in that user's restricted space."

that's the same with windows, under limited user account
but remember that even under linux, with limited rights, a malware can still send spam, show unwanted ads, steal users data, credit card number, be part of a botnet and participate to DDoS attacks. No need for root privileges to do that!

"To repair even the most screwed up profile requires removing the compromised userspace and establishing a clean one. The rest of the system remains untouched. Compromising the MODERN kernel is unheard of. (Find a validated example regarding a 2.x Linux kernel and I'll eat my words with mustard.)"

Last year, there had been a critical 0day flaw that allowed any limited user account to get root privileges, and many shared web servers with C based CGI allowed or ssh servers have been hacked through this flaw by users who had limited accounts. This flaw did allow the hackers to compromise the kernel, since a root user could do anything.
But there are flaw in windows and linux kernel in a regular basis. This one was just widely used because its exploitation details have been widely known before sysadmins had the time to patch their system.

"Compromising the Windows kernel is not only heard of, it's how the most recent Windows rootkits are evading detection. Hell, they can even get through to the BIOS on a Windows system!"

rootkits have been existing on linux systems since the 90's. There is no need to exploit a security flaw to install a rootkit. A rootkit is just a kernel module that hiddens its presence by altering some api outputs. There's nothing than a system can do to prevent rootkit from existing.

Once the user give root rights to a program, it can install as a rootkit.
No flaw here, it's just a feature, in windows/linux/osx.

"Also don't get me started on how Windows is compromised so often by viruses and rootkits and other malware because it is so popular. That is one reason, definitely it's a big target, but if that were the primary reason, then Linux would be experiencing similar rates of compromise, as it is the number one platform for web servers in the world and you'd better believe that hackers are very anxious to penetrate them."

Linux server are often compromised, at least the websites that runs onto them.
More than IIS webservers, since there have been really not much flaws in IIS and ASP.net these last years, compared to Apache/PHP and the popular php based solutions that are often deployed on top of them.

"Not to gain access to the databases and whatnot stored on them, which happens all the time due to poor application design, but to gain access to the SYSTEMS from where they could both hide and control a far greater range of activity. The difference comes down to the security model."

I advise you to update your knowledge about IIS. Security has changed since IIS4, and it's now very secure. Not to mention that ASP.net is also a very secure platform, that has rarely suffered from security flaws. Websites built using asp.net are less likely to be hacked than php based websites, because asp.net eliminates many of the common attack vectors that bad developpers are often responsible of (sql injection, script/html injection due to lack of input validation, server side includes).

June 22, 2009 at 2:20 am

вы все тупые американосы у ха ха

Евген

June 23, 2009 at 1:12 am

More food for thought from Chris Corio:

Leo Davidson

http://www.withinwindows.com/2009/06/10/uac-uac-go-away-come-again-some-other-day/comment-page-1/#comment-4025

If you keep reading after his reply, skip my “7:38am” reply as I had a HTML mishap. I re-posted it with corrections directly afterwards (“7:52am”):

http://www.withinwindows.com/2009/06/10/uac-uac-go-away-come-again-some-other-day/comment-page-1/#comment-4036

I also stumbled on this off-message post which is another example of MS doing a bad job of explaining what they’re now saying UAC is supposed to be for (and not for), even internally:

http://shippingseven.blogspot.com/2008/04/okso.html

(Assuming the blog is really by someone working on Windows 7.) FWIW, I agree with what the blog says and am pointing it out because it contradicts “the message” and not because I think it’s wrong. I agree with the blog that UAC is, or was, an imperfect but useful defence against some malware doing worse than it could, e.g. http://en.wikipedia.org/wiki/Conficker#Operation

Pingback: DotNetBurner - Security

June 24, 2009 at 7:48 am

“slither of doubt”? You mean “sliver of doubt”.

Rob

June 24, 2009 at 11:50 pm

STKD

Oh here we are in front of this highly secure Linux/OSX system I just found running. Let’s try to run something…
Oh bugger, it needs a password I don’t have. I guess that’s me well and truly defe… WAIT… what if…
/types sudo passwrd in terminal
Shit i defeated their uber complex security! LINUX IS NOW VULNERABLE! OSX IS NOW VULNERABLE!
HOLY SHART DOES ANYONE KNOW ABOUT THIS?!
Now to publish a tabloid-quality article for osnews about how every Linux/OSX system is unsecure to an admin.

June 25, 2009 at 5:16 am

Leo Davidson

Congratulations, STKD, on completely failing to understand how sudo works! To collect your free prize please run “sudo clue” on your imaginary Linux system. Have a nice day!

June 25, 2009 at 5:19 am

Leo Davidson

(Though, yes, the terminal can grab passwords, just like a spoofed UAC prompt can. Sudo works very differently to UAC, though, and to claim they are equivalent is ridiculous.)

October 8, 2012 at 6:19 pm

jono

soo were can i get this tool

March 11, 2013 at 6:37 am

Derryp7r

What we can do if sysprep.exe doesn’t exists in the folder¿?

Comments are closed.

Long Zheng

User experience entrepreneur
Melbourne, Australia

I'm a person and stuff. Mostly person, sometimes stuff. Proud introvert.

I make/made stuff people love to use:
MyPal: unofficial Melbourne myki mobile app, Omny Studio: enterprise podcast hosting, PTVGlass: Melbourne bus, tram & train timetable on Google Glass, Map2Glass: type and send addresses to Google Glass, SoundGecko: text-to-speech web reader, ChevronWP7: Windows

Phone community unlocking, MetroTwit: Twitter app for Windows, Speedo Plus: Windows Phone GPS app, Bing Image Archive: browse daily backgrounds and Windows UI Taskforce: crowdsourced bug tracker.