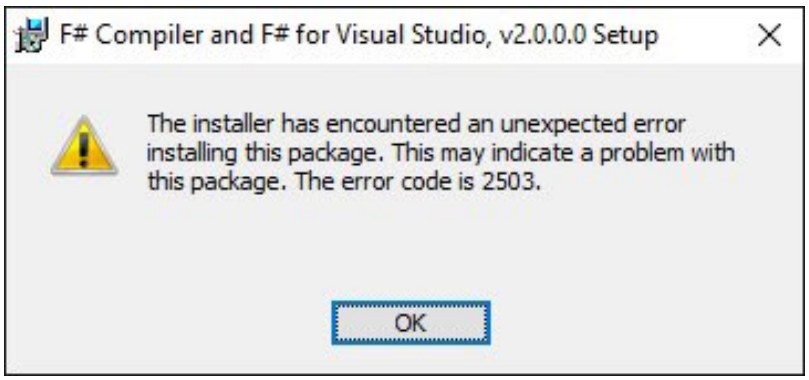
SwiftOnSecurity **@SwiftOnSecurity** *Thu Jan 28 23:28:18 +0000 2021*

We've all had Windows installers error out, especially uninstalling. And you notice they usually look kinda similar, don't they. There's a lot of installer window styles, but this is a certain one.

So why is this? Why does it give up? How do I learn more?

Let's talk MSIEXEC. ■ **https://t.co/qumQl0yjp0**



What is your concept of a program install?

Well you have files and shortcuts that have to get in the right place on the computer. So you run some copy commands. Done.

But very quickly it's not so simple. You have to do OS settings changes, utility registrations, autoruns, etc.

That's all fine in a perfect install. But what if... your installer can't do one of the steps? One of the assumptions your program or OS relies on... just didn't happen?

You have a chain of assumptions broken in the system. You are now in the badlands. An unanticipated state.

How do you make it so everything you want done, is done perfectly in sequence, or it doesn't happen?

You'd have to perfectly reverse any install steps before the failure. Go back to a Known state. Even if borked, >the computer booted at least<.

Enter the Transactional Change.

Imagine antivirus. It has a driver loading before you even see the login screen.

You uninstall antivirus, but all steps fail after a certain point. You have removed the supporting infrastructure the driver relies on, but not the driver.

So what happens? You bluescreen forever.

Computers, as much credit as you give them, are more fragile than you can possibly imagine. It has taken 60 years just to get to now.

You need to avoid unknown, unanticipated situations at all costs. So if your antivirus uninstall fails, you have to make it _never happen at all_

This category of problem is unavoidable in any suitably complex and extensible system with exponential unknown states.

Every advanced user of computers encounters it. Why are there so many Linux package formats? Because it's unsolvable. But you can try.

And Microsoft tried.

I can't even approach the level of complexity entailed here.

What if multiple programs require the same OS change be made, and the computer will break it they don't undo it, but another program needs it to stay - and will make it continue to work because they need it?

Buckle up

So Microsoft watches the world burn.

And in their Redmond offices, some of the smartest people on the planet attempt to design something that addresses all these state issues, integrated with the Windows design philosophy, that will last 100+ years.

They got real fuckin close.

Microsoft designs the MSI format - one of the most complex and detailed rules systems in existence with layers of dependency and delegation and assumption.

In their ideas is utopia distilled. Everything you could dream.

In their assumptions, the statefulness of human chaos.

In 2003, Microsoft attempts to ship the most complex software in history - Microsoft Office - almost exclusively installed by the MSI rules engine.

In theory, it is supremacy of their global dominion. Every piece in its place.

In practice, a warning to children about hubris.

I have incredible respect for the designers of MSI. I have lived it on both sides, which few ever do. I exploited their allowances in what to trigger and when. I am the devil in the shadows. **https://t.co/q0t5Cxgv8A**

Regardless, functionality of the MSI engine is a keyhole into how Windows - the OS itself - works.

If you understand the periphery of what it does, why, how to troubleshoot it, and how to subvert it to solve issues – you are an utterly exceptional technician prepared for hell.

I do job interviews. I have an extremely permissive job description with no hard requirements. I'm looking for people who came up in companies with literally nothing.

And the ability to troubleshoot MSI is a hard indicator of how much shit you've gone through.

Here's how:

<gathering screenshots and examples plz bookmark>