Reviving DDE: Using OneNote and Excel for Code Execution

By Matt Nelson

Published: 2018-01-29 · Archived: 2026-04-05 19:47:38 UTC

TL;DR: You can achieve DDE execution with Excel SpreadSheets embedded within OneNote. This bypasses the original Excel mitigation ruleset (Microsoft has released a patch to properly mitigate this) as well as the Protected View sandbox 🙂

Dynamic Data Exchange (DDE) has been a hot topic as of late. For those unfamiliar with DDE, it is designed to transfer data between two applications. In 2014, Contextis put out a nice blog post on using DDE in Microsoft Excel for code execution by utilizing the "=DDE()" formula.

Then, on October 9th 2017, SensePost released a really great blog post on abusing the DDEAUTO field code in Microsoft Word to get code execution. Shortly after, various malware families adopted the technique and it was quickly seen in the wild.

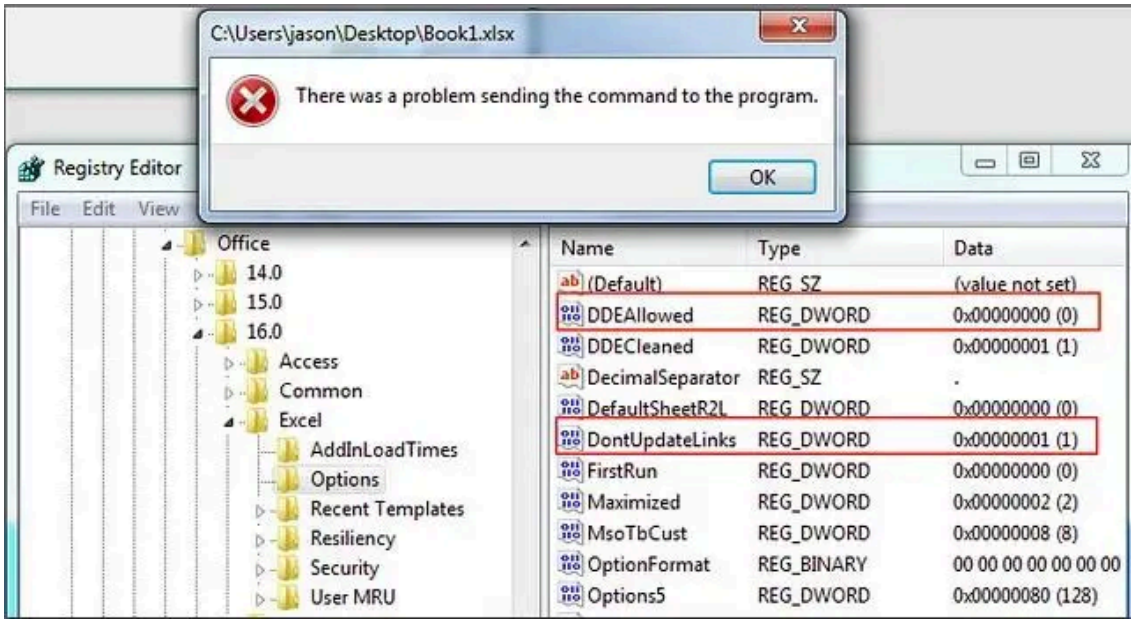
After seeing a spike in malicious use, Will Dormann (@wdormann) of US-CERT published some registry changes that would widely mitigate most DDE threats. These changes disabled DDE and prevented links from automatically updating for Word and Excel. Will added a OneNote block after sharing the details outlined below with him privately. Unfortunately the only fix was to completely kill embedded files, which is less than ideal. You can find these registry changes here: https://gist.github.com/wdormann/732bb88d9b5dd5a66c9f1e1498f31a1b

This guidance was really helpful to those dealing with actors using DDE techniques more and more. Then, on November 8th 2017, Microsoft published an official post that outlines mitigating the DDE threat for those who don't use the protocol in their environment, which was released under Advisory ADV170021 with additional documentation here. These mitigations were largely just for Word and it involved preventing any execution entirely as opposed to stopping automatic link updating. In addition to this post, Microsoft also stated that *Protected View* will prevent automatic DDE execution and that users should open untrusted documents with caution.

After seeing these new DDE mitigation recommendations, I became curious how these were handled when executed from within a different Office application, such as Publisher or OneNote. At the time, Will Dormann's gist was the only source for mitigation options in other Office apps (such as Excel) as Microsoft only released official guidance for Word.

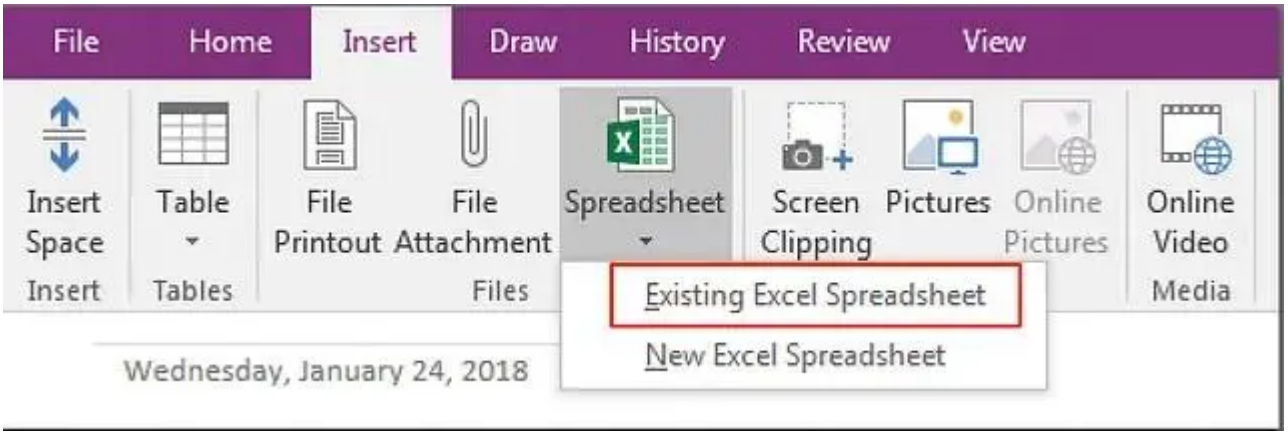
So, why OneNote? Well, it allows a user to embed Excel spreadsheets into a note document and then save it. This provides the end user the ability to reference or use Excel features directly within OneNote. As you may know, you can abuse DDE in Excel to get code execution! Ideally, Will's Excel registry changes to stop DDE attacks would apply to any Excel sheets embedded in OneNote. Unfortunately, this wasn't the case.

When implementing Will's Excel registry change (specifically "DDEAllowed" set to DWORD 0), you will see something like this when opening a spreadsheet that contains a DDE formula:

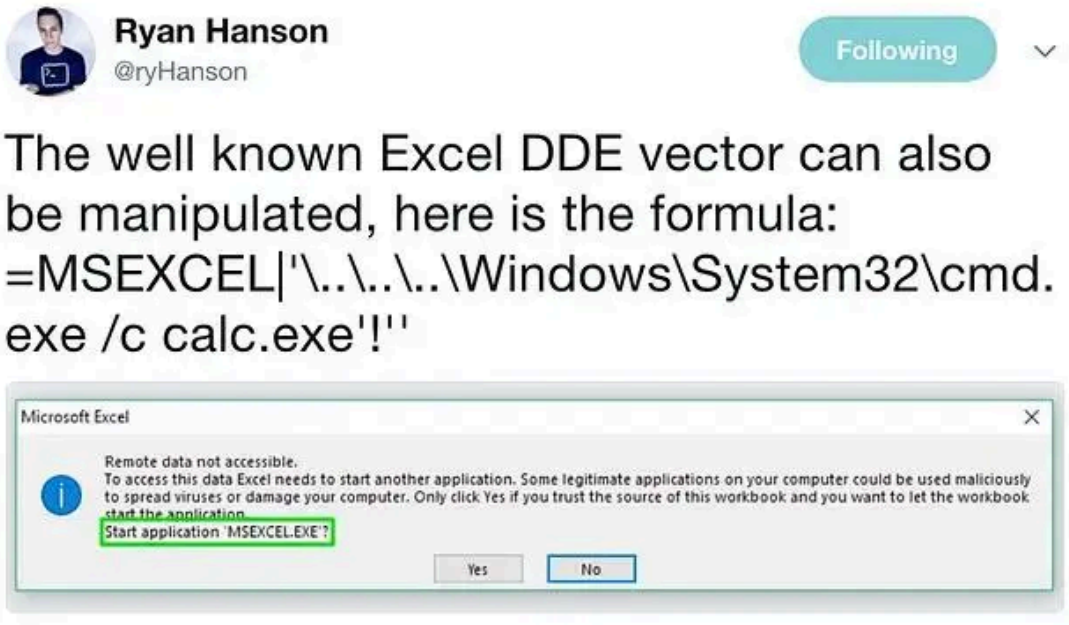


Excel DDE Blocked via Registry Mitigations

So, the Excel DDE block is working as expected. Now, let's look at OneNote. In order to utilize the Excel functionality in OneNote, you can go to "SpreadSheet" under the "Files" tab and either import an existing Excel Spreadsheet or create a new one.

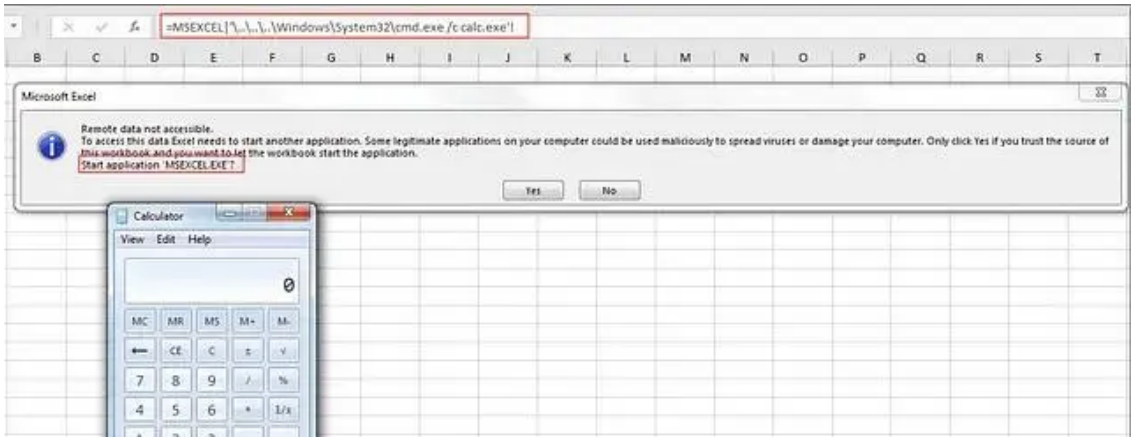


So, OneNote allows us to import an existing spreadsheet. What happens if we import a DDE-laced spreadsheet? First, we need to create it. Ryan Hanson (@ryhanson) put out a tweet showing that you can manipulate the warning box during DDE execution and change the binary name. This can be helpful as you can change it to something like "MSEXCEL.exe" instead of displaying "cmd.exe" or "powershell.exe".



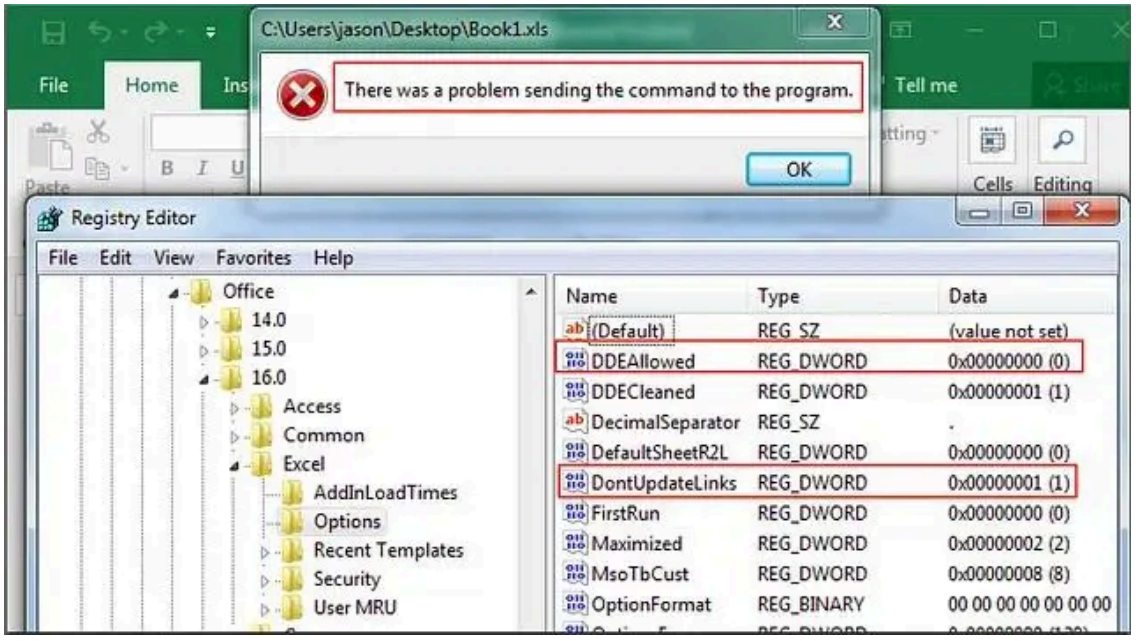
Source: https://twitter.com/ryHanson/status/918598525792935936

After adding that formula to an Excel spreadsheet and saving it, we can now test it to ensure it displays properly. To do so, I have removed the Excel DDE mitigation registry changes.



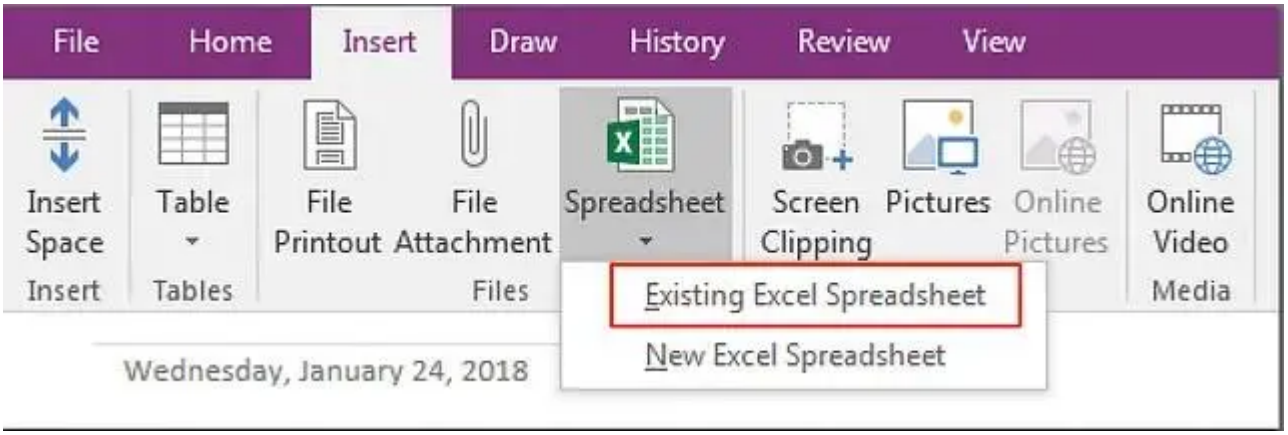
Execution without registry mitigations applied

Great, so it works. Next, let's test it with Will's Excel registry changes applied:

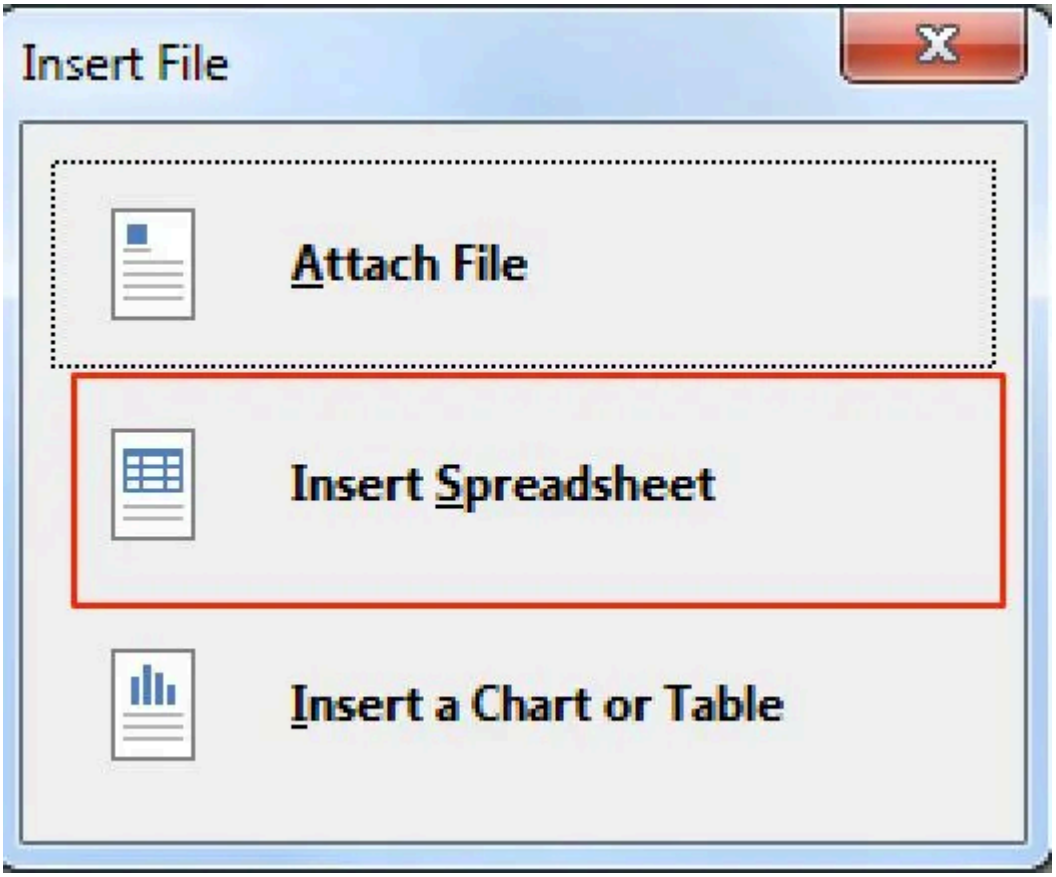


Excel DDE Blocked via Registry Mitigations

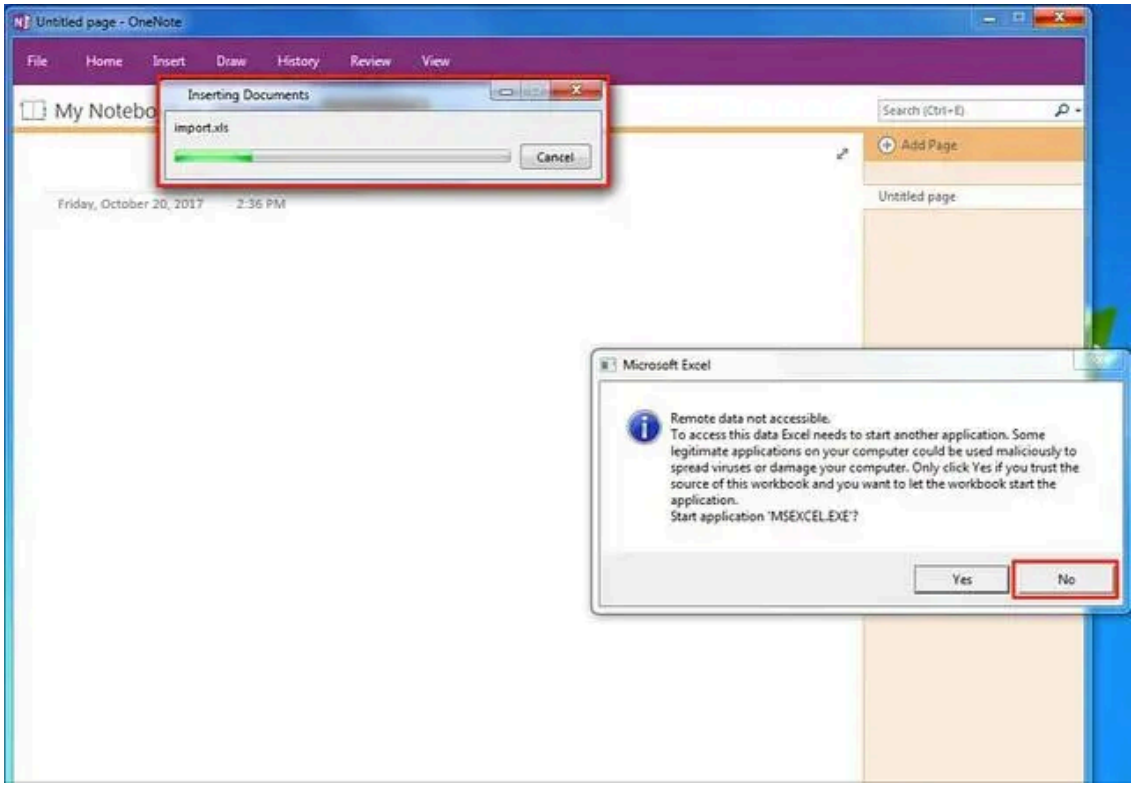
Awesome, so these changes do indeed block the Excel DDE POC that we have just created. Now that we have our DDE spreadsheet ready and tested, we can import it into OneNote by going to “Insert->SpreadSheet->Existing Excel SpreadSheet”



OneNote will ask you to browse to the file you want to import, which will be the previously created DDE laced spreadsheet. Next, it will ask you if you want to attach the file or insert the spreadsheet. We will do “Insert SpreadSheet”

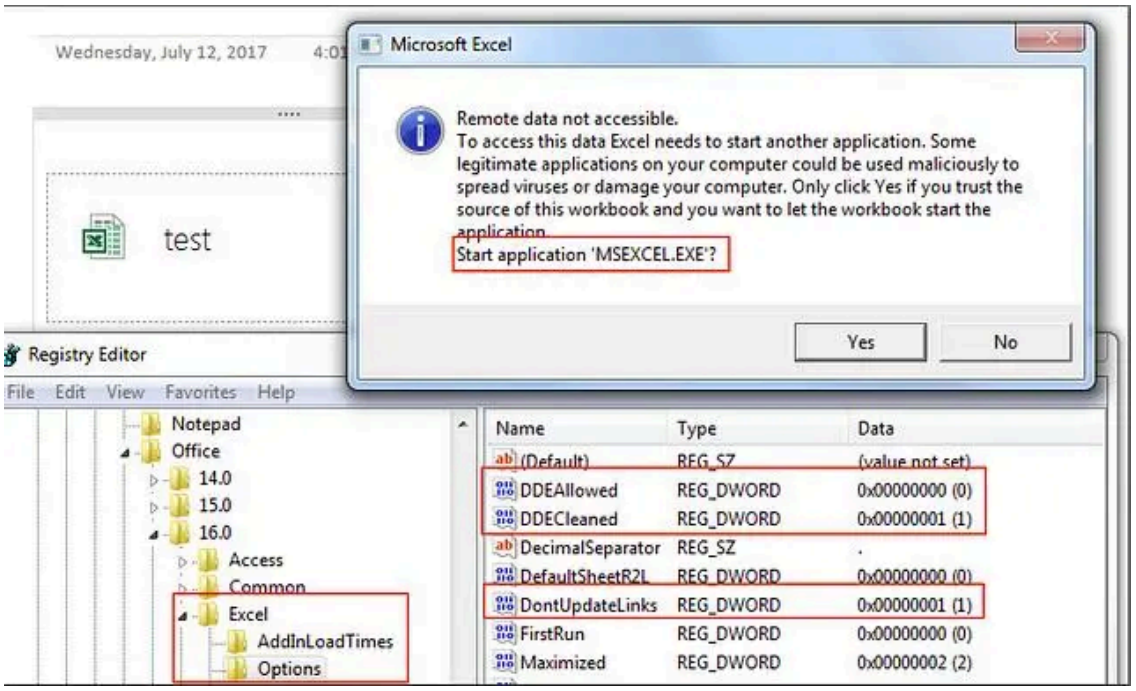


OneNote will then import the spreadsheet and during that process, it will attempt to execute your DDE command. To prevent that, simply click “No”



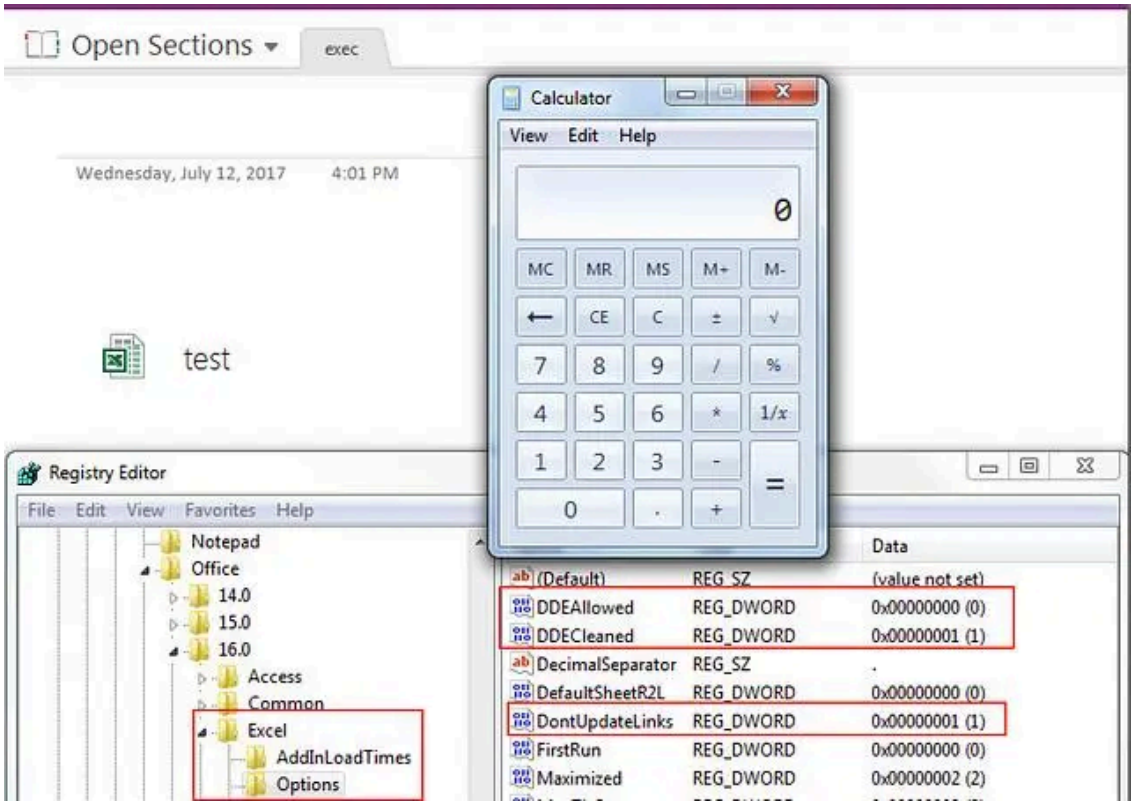
Importing weaponized Excel Sheet

Finally, save the OneNote file. At this point, that OneNote file has a DDE laced Excel SpreadSheet directly embedded in it. Now, let's see what happens when the Excel SpreadSheet is accessed from within the OneNote file with the Excel DDE mitigation registry changes in place:



DDE execution despite Excel registry changes

Clicking "Yes" results in the command being executed:

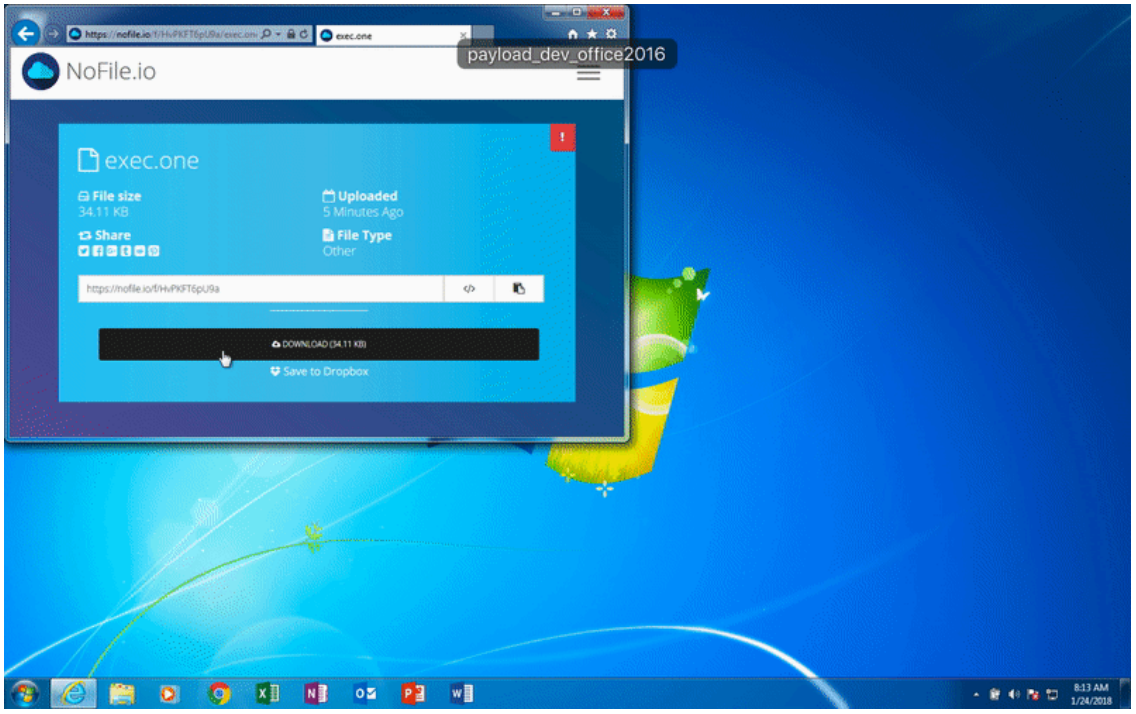


DDE execution despite Excel registry changes

So, despite blocking DDE in Excel via “DDEAllowed”, the functionality is still there when accessed through OneNote. After chatting with Will Dormann, the only working mitigation is to set “DisableEmbeddedFiles” to 1. This obviously kills all file embedding functionality as a side-effect, which isn’t great for usability.

As mentioned above, one of Microsoft’s statements notes that *Protected View* will prevent the DDE vectors when originating from an untrusted source (such as the internet). This is the case for most Office applications as any content originating from an untrusted source is opened in a sandbox first. OneNote, however, is not enrolled in *Protected View* and will not trigger it when pulled from the internet.

If a user has OneNote installed, an attacker can embed a weaponized Excel spreadsheet into a OneNote file and send it to a victim via a weblink or an email attachment. When the user receives the OneNote file and opens the embedded spreadsheet, it will not open in *Protected View* and they will simply be presented with the DDE prompt (which you can tamper with as demonstrated above):



Demo of DDE execution from the internet (bypassing Protected View sandbox)

***It should be noted that the *Protected View* aspect was reported to MSRC on April 20th, 2017 and it was deemed not a security issue.**

So, what can you do? Well, at the time, the only mitigation was to completely kill embedding in OneNote. This was reported to Microsoft on October 10th of 2017 and on January 9th, 2018 they pushed out an update to all Office versions going back to 2007. The Excel update was added to the already existing Advisory ADV170021, in which that advisory now details how to implement mitigations for both Excel and Word (since it was previously only Word that was available). Additional documentation can be found here.

This update created a value you can add under Microsoft Excel’s security options in the registry. By setting “DisableDDEServerLaunch” to DWORD 1, DDE will effectively be neutered for Excel. This is important because

OneNote itself wasn't entirely interesting. It was the embedded Excel functionality that made this attack work. By adding mitigation options for Excel, users can protect themselves from this attack.

Additionally, you can employ Attack Surface Reduction (ASR) rules in Windows 10 1709 to prevent not only DDE attacks, but other attacks where an Office program is spawning a child process. You can read more on ASR here.

-Matt N.
