DoppelDridex Delivered via Slack and Discord

Published: 2021-09-27 · Archived: 2026-04-05 21:22:52 UTC

Summary

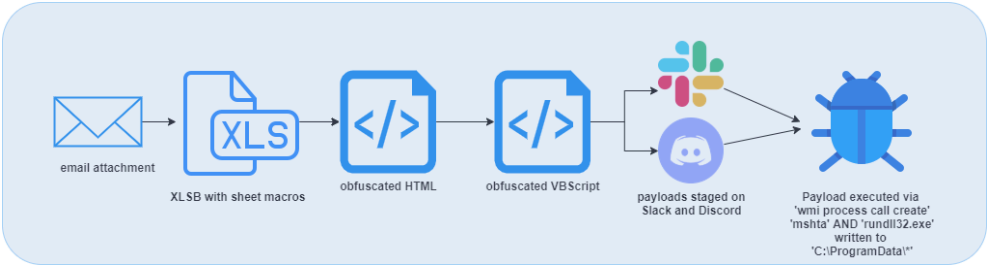
Several recent phishing campaigns have attempted to deliver a variant of the Dridex banking trojan via payloads staged on Slack and Discord CDNs. This is DoppelDridex, a modified variant of original Dridex malware. It is operated by the financially motivate eCrime adversary tracked as DOPPEL SPIDER. Additional tooling is often delivered as a secondary payload such as Cobalt Strike, which may be leveraged for further remote access, lateral movement, and preparation for deployment of Grief ransomware.

The recent campaigns delivering this malware variant have used a technique that leverages attachments with the Excel 4.0 sheet-style macros to fetch the initial payload that is hosted on domains of popular messaging CDNs such as discordapp[.]com and files.slack[.]com. These sites are likely attractive for threat actors to stage payloads because they may be trusted or allowlisted by proxies or other network-based controls. The maldocs in the phishing campaigns are also commonly built in the Microsoft Excel Binary Format (XLSB), which can cause problems for some tools designed for automated analysis.

In this blog, I will review a recent sample of a DoppelDridex Excel maldoc with .xlsb extension, and examine some analytical approaches to extracting useful information in the form of TTPs and IOCs.

Delivery and Infection Chain

The maldocs in these campaigns are delivered as attachments to emails that commonly leverage an invoice-based or tax themed social engineering lure. If the user enables contented, the sheet macro is executed. The macro code is contains series of two obfuscated HTML documents that execute embedded VBScript to retrieve the the DoppelDridex payloads from adversary-controlled infrastructure hosted by the Slack and Discord CDNs. Two files are written to the ProgramData directory. The first, is an embedded HTML document extracted from the sheet macro, which is written to ‘C:\ProgramData\[random name].rtf’. and ran via an mhta.exe process. This .rtf contains an obfuscated array, which decodes to another HTML document. The second HTML contains lightly obfuscated VBScript and is responsible for launching a shell object which then loads the main DoppelDridex payload–ultimately written to disk in ‘C:\ProgramData\defdoc.png’ and then executed by a rundll32.exe process.

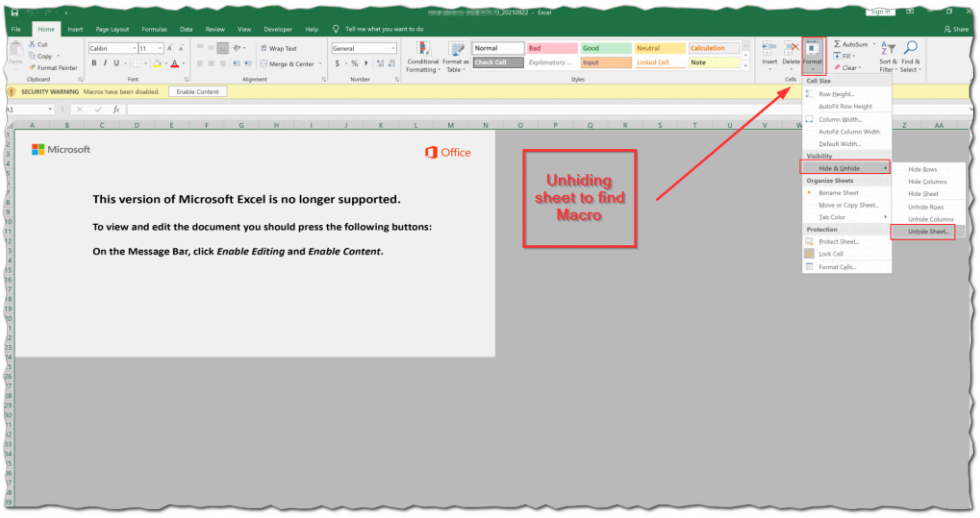


DoppelDridex infection chain

Static Sample Analysis

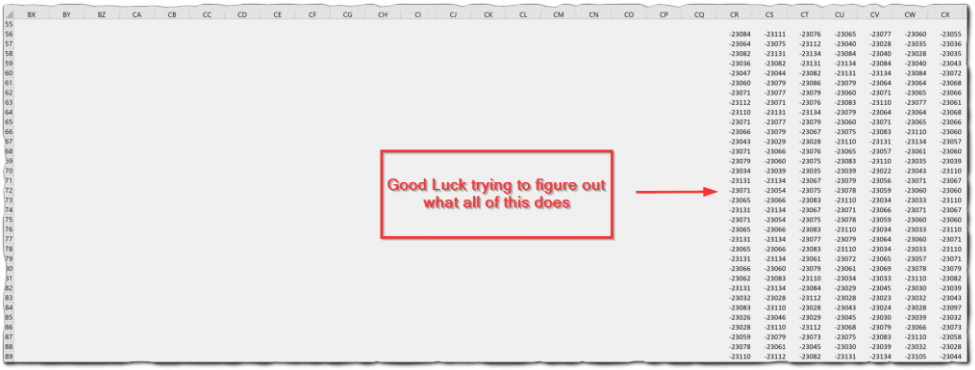
SHA256: 91164696edc4efba635e5246a48693e8fd75db2eef8e06e354848365b9fead55 (VT LINK)

The maldoc downloader is an example of an Excel document weaponized with Excel 4.0 (XLM) sheet-style macros–which have been popular for a couple of years now. This type of macro is an older standard by Microsoft that has been essentially deprecated in favor of VBA macros. However, all versions of Excel possess the capability of running Excel 4.0 macros, their use is simply discouraged. So, Excel 4.0 macros (a 20+ year standard) still work, and their functional use as a malware loader is intended for defense evasion. With sheet macros, instead of being contained in the OLE stream of a file, the code strings are simply broken up in various cells within the spreadsheet.



Hidden Sheet Macros

This can theoretically bypass detection mechanisms that are solely based on detecting compressed VBA within an OLE stream. When attempting to analyze the Excel 4.0 macro there are several options for extracting them. You could of course track the code execution throughout all of the cells, but this is not a practical as there a several cells that reference each other along with string/integer manipulation that would need to be processed. In these cases, it is not worth spending any additional time analyzing the document manually and automated tools should be used.



Sheet macro cells

I have two go-to tools that do not require opening and interacting with the file directly. The first I typically try running is the oledump.py tool from Didier Stevens. This tool has a plugin developed specifically for sheet macros that are stored in the more common .xls or .xlsx formats, which will recognize these files and extract the macros from the BIFF record (Binary Interchange File Format) inside the OLE “Workbook” stream. The BIFF record is a very old file format that pre-dates the XLS format, and the use of OLE binary data. By using the BIFF plugin, the tool will then dump all of the BIFF records in the stream. This works well on XLS format, but appears to be problematic with XLSB.

I also like Decalage‘s olevba for this type of analysis. Both tools are extremely useful for analyzing the OLE streams in documents weaponized with VBA macros. In this case, the XLSB caused me some problems with as the file format being XLSB there are literally no OLE steams to be analyzed, so the macros were not identified by my preferred tooling.

issues for automated analysis

Unfortunately, since this file is is .xlsb, neither tool are able to recognize the file or identify the macros. Of course in this case, we have the advantage of knowing there is definitely a macro contained in the maldoc. There are very many cell values and/or string values that perform malicious operations when a victim enables content.

Since both of my tried and true methods were not effective, I turned to alternative tool from DissectMalware called XLMMacroDeobfuscator. This tool uses an internal XLM emulator that is able to parse the macros without the need to actually run the code itself. Below you can see that deobfuscator not only identifies the macros, but interprets the code execution, effectively stripping out the obfuscation. This way, the URLs that are hosting the initial DoppelDridex payload on cdn.discordapp[.]com and files.slack[.]com can be easily extracted.

succesful extraction with XLMMacroDeobfuscator

The XLMMacroDeobfuscator also identifies another segment of the HTML file that is of particular interest for the next stage execution. Here, I was able to identify a large block of integers that had been assigned to an array. The key to decoding this block is by looping through the array and then subtracting “1022” from each integer. I was able to get this key from a line below the code block:

```
RKzEcSN = RKzEcSN & Chr(Round(VYITkd - 1022,0))
```

If you look closely the “VYITkd” variable is iterated through in the array via a For Each statement.

The good new is that we can use this same logic to decode this array quickly and safely. I whipped up a quick Python script to handle this operation as this was likely the fastest and easiest method. The script itself isn’t anything fancy, but it got the job done. I basically just needed to loop through the array, subtract “1022”, and then convert the resulting integer value from decimal to ascii format. Then by joining those results, I was able to get the second layer of HTML code. The Python script I used to decode the array can be found on my GitHub here:

decoding the array with Python

The final command here in the stage of the infection chain simply creates a new object and leverages a wmic process to launch rundll32 which loads the DoppelDridex DLL, which was previously downloaded as a PNG file. To recap, the commands that can be leveraged for detection are:

```
 wmic process call create 'mshta C:\ProgramData\[RANDOM].rtf'
wmic process call create "rundll32.exe C:\\ProgramData\defdoc.png"
```

More details on this maldoc can be found at the VT link provided above or the Joe Sandbox report here.

Conclusion

DoppelSpider has consistently leveraged both Discord and Slack to deliver DoppelDridex payloads to victims in recent weeks. Search for the following Dridex tags on URLhaus, and it is evident that the usage of Slack appears to ebb and flow, but Discord appears to be a preferred platform to stage their payloads. If your organization doesn't require connection to these CDNs, you might want to consider outright blocking them at your network perimeter if there is no business justification for those connections. These campaigns also consistently utilize the XLSB file format that may cause some problems for automation that relies on identifying malicious content in OLE steams. Despite this, static analysis can be accomplished with tools that can emulate the macros in the XLSB document type, which easily extract the embedded IOCs.

Technical controls at the mail gateway typically have very high success rates for defeating commodity malware delivered in opportunistic campaigns. The EXCEL 4.0/XLM macros in the maldocs with XLSB format may evade detection for similar reasons as noted above. The TTPs presented here can provide some additional detection opportunities for a layered defense strategy. I have also presented some analysis techniques that can be used in response efforts to quickly identify and extract IOCs when needed. This campaign is a few days old of the time of this writing, however, the TTPs should still be relevant.

IOCs

Delivery Maldoc SHA256: 91164696edc4efba635e5246a48693e8fd75db2eef8e06e354848365b9fead55

DoppelDridex DLL SHA256: acbcd5ce1579a43148eee9b867f035cd0bc16f237a4790322467a0dac23ce7c6

DoppelDridex DLL SHA256: a6aaa4ffb112d78aa20345821920ce6554d96303f7fb3facb5143de348cf2aae

hxxps[:]//cdn.discordapp[.]com/attachments/890212086519566369/890212261132636200/5_samsrv.dll.dll
hxxps[:]//cdn.discordapp[.]com/attachments/890212086519566369/890212251435425862/0_system.componentmodel.composition.registra
hxxps[:]//cdn.discordapp[.]com/attachments/890212591471824921/890212677559922708/9_dispex.dll.dll
hxxps[:]//files.slack[.]com/files-pri/T02F79UM6TT-F02F9AE9ZJ6/download/3_SmiEngine?pub_secret=4e9eeb9360

hxxps[:]//files.slack[.]com/files-pri/T02ERNYLC69-F02F9AG9CEN/download/6_hpzstw72?pub_secret=356a094b3b
hxxps[:]//files.slack[.]com/files-pri/T02EHM1BB19-F02FFGMT84C/download/6_hpzstw72?pub_secret=009a86b011

References

https://www.crowdstrike.com/blog/doppelpaymer-ransomware-and-dridex-2/

https://redcanary.com/blog/grief-ransomware/

https://www.virustotal.com/gui/file/91164696edc4efba635e5246a48693e8fd75db2eef8e06e354848365b9fead55/community

https://docs.microsoft.com/en-us/openspecs/office_file_formats/ms-xlsb/acc8aa92-1f02-4167-99f5-84f9f676b95a

https://support.microsoft.com/en-us/office/working-with-excel-4-0-macros-ba8924d4-e157-4bb2-8d76-2c07ff02e0b8?ocmsassetid=ha010336614&correlationid=2aa46e64-978f-4d6a-bf7d-950ab12599a1&ui=en-us&rs=en-us&ad=us

https://www.virustotal.com/gui/file/91164696edc4efba635e5246a48693e8fd75db2eef8e06e354848365b9fead55/community

https://www.decalage.info/python/olevba

https://github.com/DissectMalware

https://github.com/DissectMalware/XLMMacroDeobfuscator

https://github.com/Sec-Soup/Python-ToolBox/tree/master/array-decoder_2

https://www.joesandbox.com/analysis/488098/0/html

https://urlhaus.abuse.ch/browse/tag/Dridex/
