


☐

I'm not robot


reCAPTCHA

Continue

Winrar sfx build sfxrar32 release sfxrar.pdb

If your antivirus recognizes this page as malicious, it's likely because I pasted some scripts into the article. Don't be alarmed! Introduction recently I have (finally) set up proper VM malware analysis so I thought it would be super fun to download some samples from malware distributors on YouTube. I came across something interesting from which I learned and expanded my knowledge so I thought I'd share my analytical journey with you in the hope that you would learn something too! Author assigned level: - Community assigned level: Newbie Wannabe Hacker Wizard Guru required nothing unusual skills, these are completely optional and there is no need to understand the analysis. Visual Basic C# .NET reverse engineering forensics persistence and common sense analysis with all things hacking, reconnaissance should be the first step toward understanding the target. The first thing I do is upload the VirusTotal sample (VT) to see if it has been detected or not, or just to try and get an overview of what I might be dealing with. Okay, so there are some revelations but it seems pretty generic and it doesn't really help me much. pestudio - Firestorm.exe since VT was not helpful, I had to get a better understanding of what this file is and what it can do. So, using a pesto, maybe I can get some clues. We can see here there are quite a few flags raised, mainly: it is trying to appear as a Microsoft Startup IEXPLORE.exe which is Internet Explorer, it seems to have another file inside, it redirects file streams so that it may write to disk, it redirects encryption functions which may indicate some kind of debug file name complication is sfxrar.pdb which indicates a self-extracting archive, most likely WinRAR. If we enter the manifest, we can see that it tells us that it is WinRAR SFX: extracting files with WinRAR - Firestorm.exe Let's open this archive in WinRAR and see what it contains: on the right, we can see the SFX script which lists the path where these files extract, file execution after discharge, silent install and replace file. Of course making the file is what catches our attention so let's extract these files and have a look at them. pestudio - sfx.exe so the file name of the sfx.exe gives us a pretty good clue as to what it might be so let's confirm with pestudio: yes, as we expected. Can we get him out? Unfortunately, no. So let's get on with form1.vbe. Deobfuscating Visual Basic - form1.vbe When the file is opened, this is what we see: TZUO3TWPSQYCGRU=newX\$S;-lok <> <> <<af "eHbD><<UX2}AAEÑeEÄyC\$ÄZE"EIA[hwb~ S\$UAFSbw Y p;@A ä&MZIT...;HJZEÄA? LbFytp; \$UAFSbpyb;@EJZEUA? L Fw... - 1ZDf6D*oT...@HJZEUAGSbwSp;@JZ BUbFw ä-1AEtG"ijb--> Who received the Nobel Peace Prize, was awarded the Nobel Peace @L9i <it"af "eHbD>it"" <<><<af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU . CZUO3TWPSQYCGRU=len(PZUO3TWPSQYCGRU) : do until NZUO3TWPSQYCGRU=len(TZUO3TWPSQYCGRU) : NZUO3TWPSQYCGRU=NZUO3TWPSQYCGRU+1 : UZUO3TWPSQYCGRU=UZUO3TWPSQYCGRU+1 : if UZUO3TWPSQYCGRU=len(PZUO3TWPSQYCGRU) then UZUO3TWPSQYCGRU=1 WZUO3TWPSQYCGRU=WZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come as a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get. Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This := pzuo3twpsqycgru=ZUO3TWPSQYCGRU := czuo3twpsqycgru=len(PZUO3TWPSQYCGRU) := do= until= nzuo3twpsqycgru=len(TZUO3TWPSQYCGRU) := nzuo3twpsqycgru=NZUO3TWPSQYCGRU+1 := uzuo3twpsqycgru=UZUO3TWPSQYCGRU+1 := if= uzuo3twpsqycgru=len(PZUO3TWPSQYCGRU) then= uzuo3twpsqycgru=1 wzuo3twpsqycgru=WZUO3TWPSQYCGRU &= chrw(ascw(mid(tzuo3twpsqycgru,= nzuo3twpsqycgru,= 1))&= := "*= czuo3twpsqycgru)= := loop:= := execute= wzuo3twpsqycgru= and= what= a= mess= it= is= obfuscation= for= scripts= is= a= very= common= thing= so= it= should= not= come= as= a= surprise.= the= problem= for= us= is= to= deobfuscate= it= somehow.= we= can= see= that= there= is= a= string= which= may= contain= another= payload= once= it= has= been= deobfuscated= and= some= function= calls= but= it's= still= very= confusing= and= may= take= us= some= time= to= reverse= engineer= it.= especially= for= people= like= me= who= are= not= quite= familiar= with= visual= basic.= what's= a= simple= way= we= can= continue= without= too= much= effort?= on= the= end= of= the= script.= we= see= an= execute= function= on= the= wzuo3twpsqycgru= variable.= so= let's= be= smart= here= and= instead= of= having= execute= there= we= force= it= to= write= the= contents= out= to= disk!= after= a= quick= google= search.= it= seems= we= just= need= to= replace= it= with= wscript.echo.= then= call= the= script= using= cscript.exe.= (important!= otherwise= it= will= pop= up= a= message= box!).= here= is= the= output:= (the= following= script= has= sections= removed= to= prevent= av= detection)= dim= df= mohmrret=83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 end= if= mohar=MOHMRRET mohmr=SPLIT(mohar,CHRW(10 + 4)) end= if= execute= (flam)= end= if= yes= indeed= it= is= another= payload = again.= we= can= see= it's= obfuscated= but= this= time.= it's= makes= a= bit= more= sense = there= is= another= potential= payload= held= in= the= mohmrret= variable= which= looks= to= be= ascii= with= some= kind= of= character= woven= in.= we= can= see= the= split= function= using= mohar= (mohmrret)= and= a= character= 10= + 4= which= may= confirm= the= assumption.= instead= of= deobfuscating= this= file.= we= want= to= be= smart= so= similar= to= the= previous= script.= let's= replace= execute= with= wscript.echo= and= see= what= we= get.= microsoft= (r)= windows= script= host= version= 5.8= copyright= (c)= microsoft= corporation.= all= rights= reserved.= set= wshshell=WScript.CreateObject(WScript.Shell) wshshell.run= sfx.exe -pT*8+nArvEpTWEE# there= is= no= more= obfuscation= and= we= can= see= references= to= sfx.exe.= this=<</7B-EBÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU . CZUO3TWPSQYCGRU=len(PZUO3TWPSQYCGRU) : do until NZUO3TWPSQYCGRU=len(TZUO3TWPSQYCGRU) : NZUO3TWPSQYCGRU=NZUO3TWPSQYCGRU+1 : UZUO3TWPSQYCGRU=UZUO3TWPSQYCGRU+1 : if UZUO3TWPSQYCGRU=len(PZUO3TWPSQYCGRU) then UZUO3TWPSQYCGRU=1 WZUO3TWPSQYCGRU=WZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get: Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This ><</af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get: Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This ><</af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get: Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This ><</af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get: Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This ><</af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get: Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This ><</af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get: Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This ><</af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get: Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This ><</af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get: Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This ><</af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get: Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This ><</af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQYCGRU, NZUO3TWPSQYCGRU, 1)) - UZUO3TWPSQYCGRU * CZUO3TWPSQYCGRU) : loop : execute WZUO3TWPSQYCGRU and what a mess it is. Obfuscation for scripts is a very common thing so it should not come a surprise. The problem for us is to deobfuscate it somehow. We can see that there is a string which may contain another payload once it has been deobfuscated and some function calls but it's still very confusing and may take us some time to reverse engineer it, especially for people like me who are not quite familiar with Visual Basic. What's a simple way we can continue without too much effort? On the end of the script, we see an execute function on the WZUO3TWPSQYCGRU variable. So let's be smart here and instead of having execute there we force it to write the contents out to disk! After a quick Google search, it seems we just need to replace it with WScript.Echo, then call the script using cscript.exe (important! otherwise it will pop up a message box!). Here is the output: (The following script has sections removed to prevent AV detection) DIM DF MOHMRRET = 83101116328711510483104101108108326132878399114105112116466711410197116101799810610199116403487839911410511211646831041011081083441131087115104831041011081084682117110323411510212-04610112010132324511284945643110658211869112841198769693534 END IF mohar = MOHMRRET MOHMRT = SPLIT(mohar,CHRW(10 + 4)) END IF EXECUTE (FLAM) END IF Yes, indeed it is another payload. Again, we can see it's obfuscated but this time, it's makes a bit more sense. There is another potential payload held in the MOHMRRET variable which looks to be ASCII with some kind of character woven in. We can see the split function using mohar (MOHMRRET) and a character 10 + 4 which may confirm the assumption. Instead of deobfuscating this file, we want to be smart so similar to the previous script, let's replace execute with WScript.Echo and see what we get: Microsoft (R) Windows Script Host Version 5.8 Copyright (C) Microsoft Corporation. All rights reserved. Set WshShell = WScript.CreateObject(WScript.Shell) WshShell.Run sfx.exe -pT*8+nArvEpTWEE# There is no more obfuscation and we can see references to sfx.exe. This ><</af "eHbD><<<<<<<<<7B-EDÖö&EÖrZr;»> }YCEä...r\$} Dâ\$™œCŦ. YJ<, aljBp%ä...-cŦ™œ . PZUO3TWPSQYCGRU=ZUO3TWPSQYCGRU & ChrW(AscW(Mid(TZUO3TWPSQY