



Assignment 4: Research Assignment: Future PowerShell Use

AUTOMATING SECURITY AND ACTIVE DIRECTORY USER MANAGEMENT



Harshul Hareshkumar Shukla (8986048)

Applied Network Infrastructure and System Administration

Prof.Serge Trunkin

NTWK8056: Scripting with PowerShell

CONTENTS

Research Task	3
Introduction	3
1. Manual User Management	3
Challenges presented because of manual user management	4
2. Security Management	4
Challenges presented because of manual security management	5
Research Solution.....	6
PowerShell	6
Work-Flow	9
PowerShell Automation Presentation of Working script	10
Security management.....	10
User Management	13
Task Management	20
Script Functions Information Table	23
PowerShell Code.....	24
References.....	47
Appendix.....	48

RESEARCH TASK

INTRODUCTION

As a system administrator managing user accounts within Active Directory is a crucial task but also takes a lot of time. As the company or organizations grow, the number of users also increases, making manual user management difficult and increasing the chances of errors. In this, we will explore the automation of Active Directory user management using PowerShell which will focus on tasks such as adding, updating, enabling disabling, and deleting user accounts and sending an E-mail to a related person stating that the requested task has been done.

Additionally, there are many other system security-related tasks such as malware scans, managing firewall rules, performing system audits, monitoring system logs, updating the system, taking backups, optimization of resource allocation, preventing system downtime, checking for software updates, tuning the system performance and tech support - troubleshooting. All these tasks are essential for maintaining the security and integrity of the system and take a lot of time. we can manage all that with the use of PowerShell and automating them can ensure that they are performed on time regularly and with consistency.

This research task shows the difficulties in managing AD and suggests that PowerShell scripting can be used to effectively automate and optimize these procedures as discussed by **(Company, 2023)**.

1. MANUAL USER MANAGEMENT

Manual user management in AD involves going through many admin tools and interfaces like ADUC or PowerShell cmdlets, system admin must perform tasks such as creating, updating, and enabling new user accounts and modifying related attributes, and the offboarding process involves disabling or deleting accounts as per the requirements of the organization and notifying HR about these changes.

CHALLENGES PRESENTED BECAUSE OF MANUAL USER MANAGEMENT

- Time-consuming: as organizations grow number of users increases and adding them one by one is a noticeably big challenge due to the time it takes to do user-related tasks.
- Error-possibilities: with manual work human error happens such as typing wrong e-mail addresses or incorrect entries which can lead to issues such as misconfigured user accounts or service disruption, Issues such as unauthorized access can also arise from it.
- Inconsistency: Manual processes might lead to inconsistencies in user configuration across the AD environment, leading to operational issues such as deleting or disabling the wrong user account or adding data to different users with the same name.
- Scalability problems: for a large-scale deployment manual user management becomes impractical it is also difficult for organizations with frequent staff changes.
- Sending E-mail: Once a new user has been added, updated, or deleted from ADUC as per requirements system admin also has to inform the related authority that the required task has been done and if you are adding 2-3 users then it might not feel like an issue but if you are adding many users in one day then sending so many emails can be a time-consuming and challenging task.

2. SECURITY MANAGEMENT

It takes time and effort to perform security-related, optimization-related, or compliance-checking chores manually. In addition, there's a chance the system administrator forgot to complete these duties or was too preoccupied with other work at the time. Users must be informed about several of these duties that may have an impact on them. Tasks including system audits, firewalls, log monitoring, data backups, update management, preventing system outages, infrastructure scalability, regulatory compliance, optimizing system performance, and troubleshooting can also be included.

CHALLENGES PRESENTED BECAUSE OF MANUAL SECURITY MANAGEMENT

- Malware Scanning: Malware scanning is essential for maintaining security, and manually doing it lacks consistency which may result in issues later on.
- Firewall Management: Firewalls are the most critical components of network security; manually managing firewall rules to block incoming traffic on a specific port creates a security issue.
- System Auditing and Log Monitoring: Regular system audits and monitoring can help detect unusual activity that can catch security breaches and log monitoring can provide information about system activities and help identify incidents manually doing these things can take time and mistakes might lead to security risks.
- System Updates: keeping the system updated is most important for security as updates include security patches and manually doing them leads to issues related to security as the system admin can miss if too busy or out of the office.
- Data-Protection: Regular backups can help ensure that important data can be recovered during an incident and doing this manually creates huge risks.
- Prevent system downtime: The system admin has to manually add information in the task scheduler and it's also time-consuming to send emails to users who might be affected.
- Regulation compliance: The system admin also needs to make sure that everything is as per regulations, checking all of the tasks requires a lot of time.
- Performance Tuning and Resource optimization: This involves checking if all the resources are used correctly and manually checking it takes time.
- Technical support and Troubleshooting: There are many critical issues the system admin has to face in troubleshooting, there are many services like printers and servers, and manually checking everything is time-consuming.

In Summary, Automation will be able to improve the accuracy, efficiency, and consistency of user management and security management, and also improve and free system admins to focus on other tasks.

RESEARCH SOLUTION

POWERSHELL

This research solution automates tasks with PowerShell, optimizes workflows to boost output, fortifies security, and ensures legal compliance. A method for email automation is also included in the solution, and it's utilized to inform relevant parties about the progress of different activities. System administrators find PowerShell to be an indispensable tool due to the substantial time savings, fewer errors, and enhanced system performance and stability that result from automating these activities.

User Management: It is possible to automate user administration tasks with PowerShell cmdlets. For instance, use the New-ADUser cmdlet to add a new user to Active Directory (AD). Options like -Name, -GivenName, -Surname, -EmailAddress, and others are accepted. -Explanation, -UserPrincipalName, -SamAccountName, -Office, -Workplace Phone -Country of Work, Use -Enabled, and -AccountPassword to fully configure the new user.

Security Management: Security tasks such as malware scanning can be automated using the Start-MpScan cmdlet, which initiates a malware scan. Firewall rules can be managed using the New-NetFirewallRule cmdlet.

Email Automation Using the Sendemailuseradd function, an automatic email is sent to HR following the completion of user-related actions. The Net.Mail is used by this function. Send an email using the SmtpClient class. The SMTP server name, port, and the EnableSsl attribute are used to instantiate the SmtpClient class, and it is set to true.

A PSCredential object, which is generated with the sender's email address and an app-specific password, is what the Credentials field of the SmtpClient object is set to. The ConvertTo-SecureString cmdlet is used to transform the app-specific password into a secure string.

Data Protection: Data can be copied between locations using the Copy-Item cmdlet, which essentially creates a backup of the original file. The parameters -Path, -Destination, -Recurse, and -Force are accepted by this cmdlet.

Resource Management: System resources, such as memory, can be monitored to optimize resource allocation. The Win32_OperatingSystem class and the Get-WmiObject cmdlet can be used to accomplish this, retrieving details about the operating system, including free physical memory.

System Downtime Prevention: The Register-ScheduledTask cmdlet can be used to schedule system reboots. The -Action, -Trigger, -TaskName, and -Description options are accepted when using this cmdlet to create a new scheduled task.

Software Updates Management: To find out what software updates are available, use the Get-WindowsUpdate cmdlet. These updates can then be installed using the Install-WindowsUpdate cmdlet. The PSWindowsUpdate contains these cmdlets.

Infrastructure Scalability Planning: By keeping an eye on system resources like CPU utilization, infrastructure scalability may be planned. To obtain performance counter information, including processor time, use the Get-Counter cmdlet.

Compliance Checks: Encrypting data allows one to verify compliance with requirements. Although there isn't a built-in cmdlet for this in PowerShell, it is possible to write a custom function to verify if data at rest is encrypted.

Performance Tuning: Process priority can be changed to fine-tune system performance. A list of all processes that are currently executing can be obtained using the Get-Process cmdlet, and each process PriorityClass attribute can be changed as necessary.

Technical Support and Troubleshooting: By using the Get-Service cmdlet to provide information about a service and its dependencies and the Get-WinEvent cmdlet to search event logs for related occurrences, technical help and troubleshooting may be automated.

In conclusion, PowerShell offers a stable basis for automating several kinds of system administration work. Administrators can increase security, check regulatory compliance, optimize workflows, and increase efficiency by utilizing particular PowerShell cmdlets. Significant time savings, a decrease in errors, and enhanced system stability and performance can all result from this.

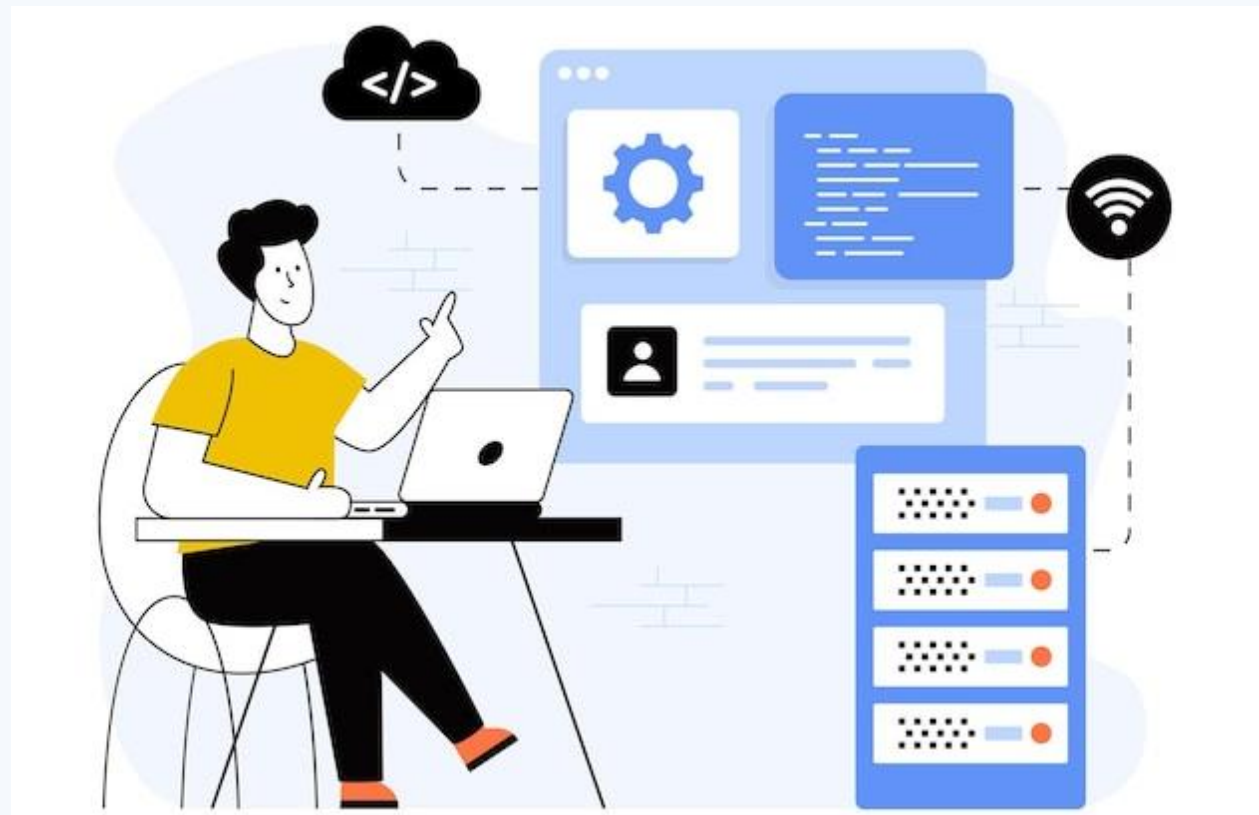


Figure 1(Premium Vector | System Administrator, 2021)

WORK-FLOW

The flow chart demonstrates the working of the PowerShell script that solves problems by automating the described tasks which saves time and reduces errors.

PowerShell Solution Flowchart

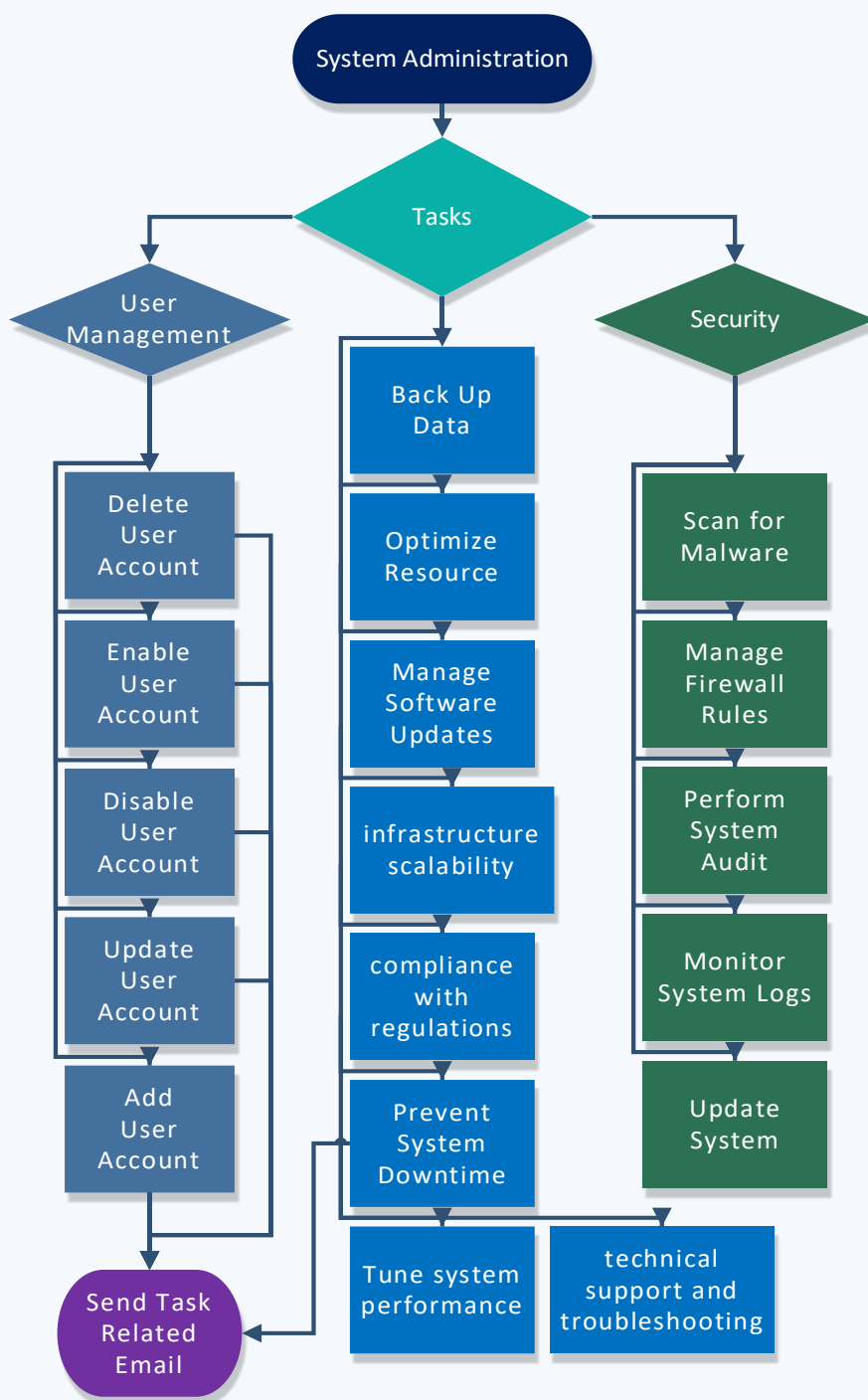


Figure 2 PowerShell code working flowchart.

POWERSHELL AUTOMATION PRESENTATION OF WORKING SCRIPT

SECURITY MANAGEMENT

Once the script is executed it will import all the necessary modules and install modules as per requirements. The script will show a list of actions that can be achieved by running it.

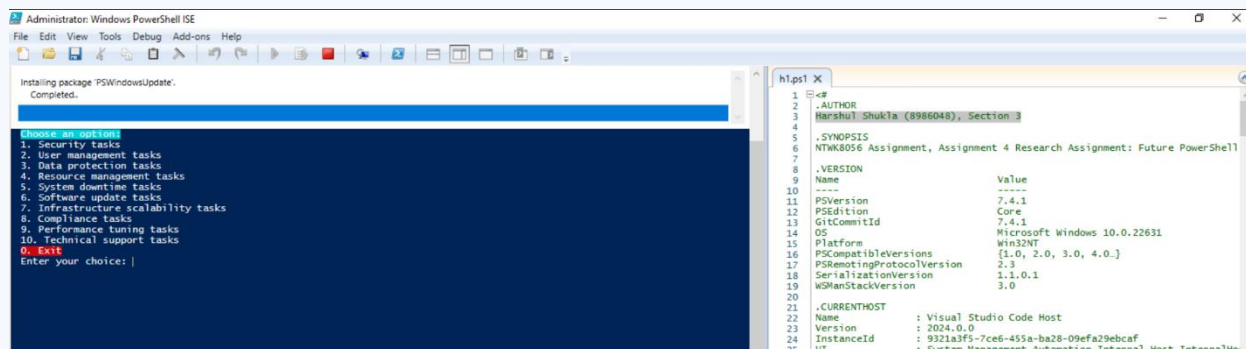


Figure 3 Installing modules and showing a list of actions.

Malware Scan: The first option is security management, which includes malware scans. We can execute it by just choosing no 1 and pressing enter.

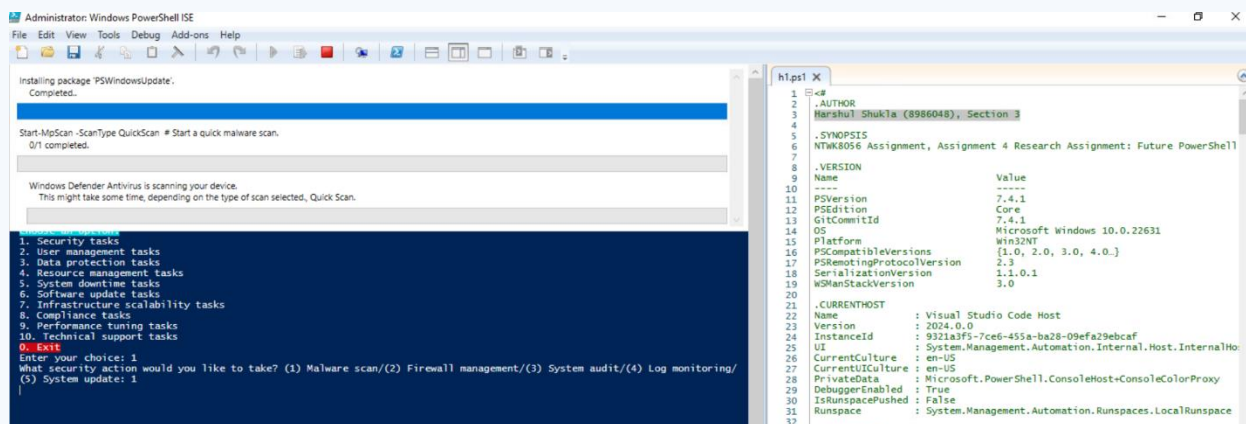


Figure 4 Malware scanning in the process.

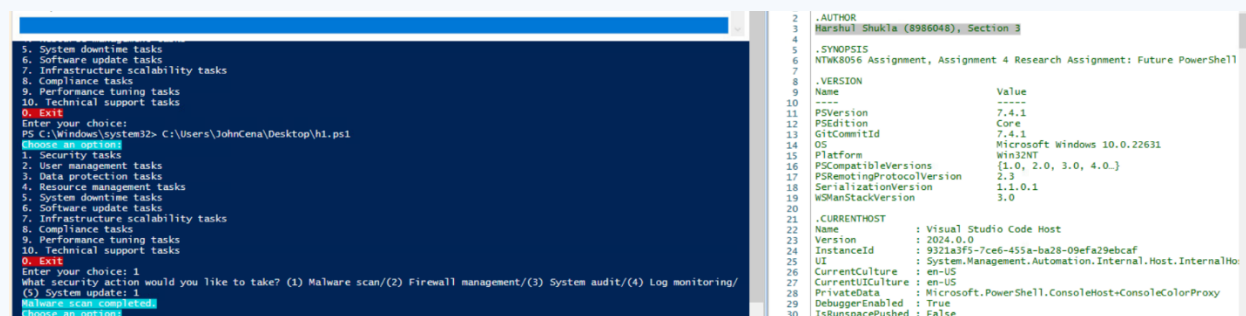
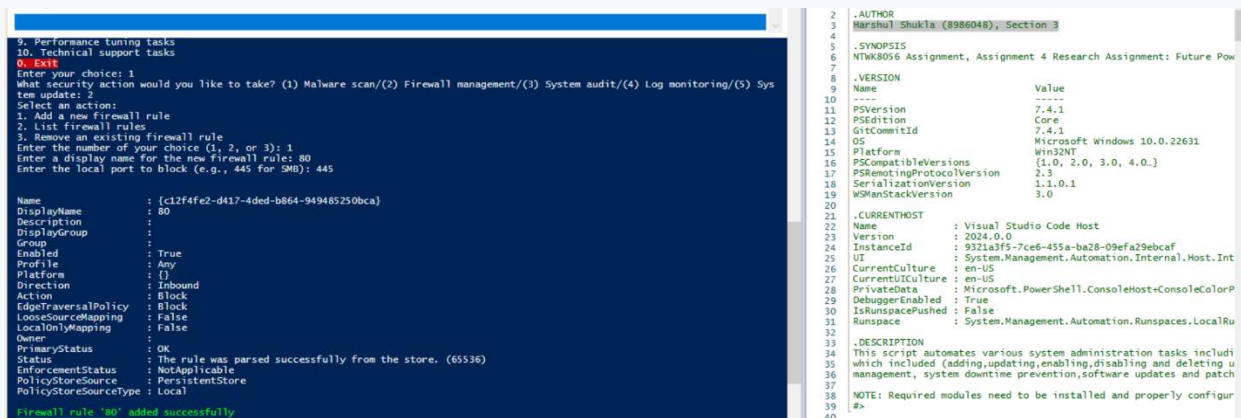


Figure 5 Malware scan complete.

Firewall Management: The second option is firewall rules management we can set the rules and running the script would set it as per requirements. We can block and unblock traffic using this function.

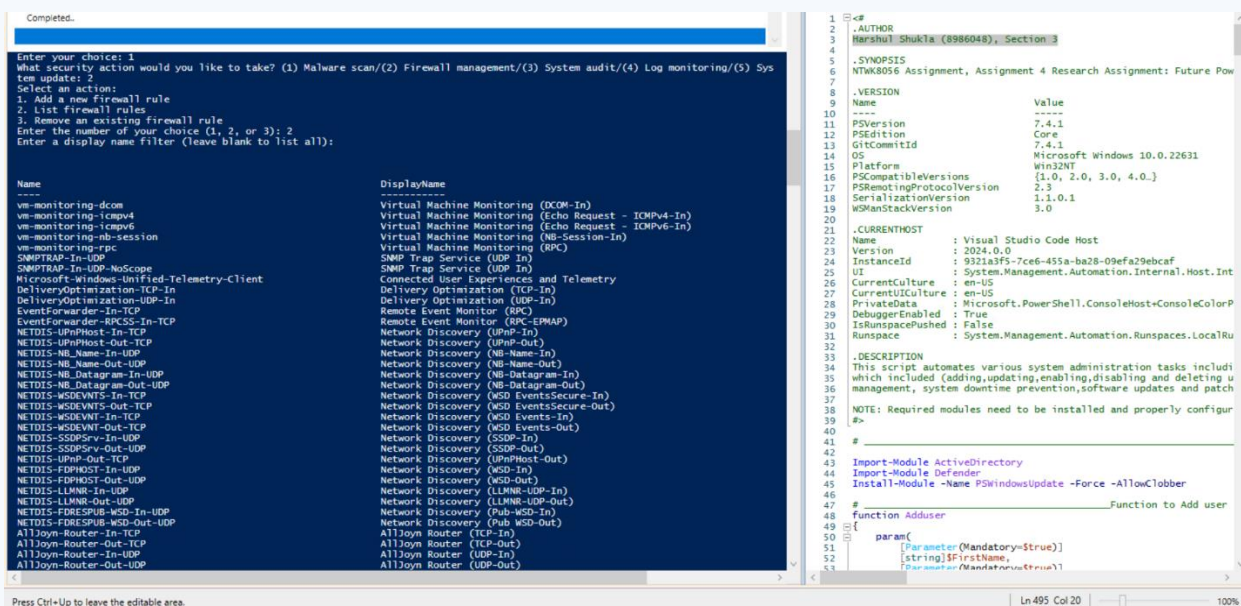


```

1 2
3 .AUTHOR
4 Harshul Shukla (@986048), Section 3
5
6 .SYNOPSIS
7 NTWK8056 Assignment, Assignment 4 Research Assignment: Future Pow
8
9 .VERSION
10 Name Value
11 -----
12 PSVersion 7.4.1
13 PSEdition Core
14 GitCommitId 7.4.1
15 OS Microsoft Windows 10.0.22631
16 Platform Win32NT
17 PSCompatibleVersions {1.0, 2.0, 3.0, 4.0}
18 PSRemotingProtocolVersion 2.3
19 SerializationVersion 1.1.0.1
20 WSManStackVersion 3.0
21
22 .CURRENTHOST
23 Name : Visual Studio Code Host
24 Version : 2024.0.0
25 InstanceId : 9321a3f5-7ce6-455a-ba28-09efa29ebcaf
26 UI : System.Management.Automation.Internal.Host.Int
27 CurrentCulture : en-US
28 CurrentUICulture : en-US
29 PrivateData : Microsoft.PowerShell.ConsoleHost\ConsoleColorP
30 DebuggerEnabled : True
31 IsRunspacePushed : False
32 Runspace : System.Management.Automation.Runspaces.LocalRu
33
34 .DESCRIPTION
35 This script automates various system administration tasks includi
36 which included (adding, updating, enabling, disabling and deleting u
37 management, system downtime prevention, software updates and patch
38 NOTE: Required modules need to be installed and properly configur
39 #
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 6 Add Firewall Rule

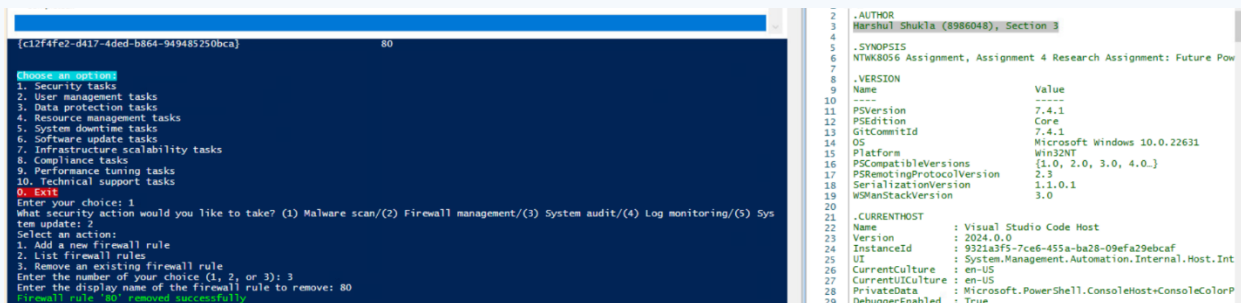


```

1 2
3 .AUTHOR
4 Harshul Shukla (@986048), Section 3
5
6 .SYNOPSIS
7 NTWK8056 Assignment, Assignment 4 Research Assignment: Future Pow
8
9 .VERSION
10 Name Value
11 -----
12 PSVersion 7.4.1
13 PSEdition Core
14 GitCommitId 7.4.1
15 OS Microsoft Windows 10.0.22631
16 Platform Win32NT
17 PSCompatibleVersions {1.0, 2.0, 3.0, 4.0}
18 PSRemotingProtocolVersion 2.3
19 SerializationVersion 1.1.0.1
20 WSManStackVersion 3.0
21
22 .CURRENTHOST
23 Name : Visual Studio Code Host
24 Version : 2024.0.0
25 InstanceId : 9321a3f5-7ce6-455a-ba28-09efa29ebcaf
26 UI : System.Management.Automation.Internal.Host.Int
27 CurrentCulture : en-US
28 CurrentUICulture : en-US
29 PrivateData : Microsoft.PowerShell.ConsoleHost\ConsoleColorP
30 DebuggerEnabled : True
31 IsRunspacePushed : False
32 Runspace : System.Management.Automation.Runspaces.LocalRu
33
34 .DESCRIPTION
35 This script automates various system administration tasks includi
36 which included (adding, updating, enabling, disabling and deleting u
37 management, system downtime prevention, software updates and patch
38 NOTE: Required modules need to be installed and properly configur
39 #
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 7 List Firewall Rules



```

1 2
3 .AUTHOR
4 Harshul Shukla (@986048), Section 3
5
6 .SYNOPSIS
7 NTWK8056 Assignment, Assignment 4 Research Assignment: Future Pow
8
9 .VERSION
10 Name Value
11 -----
12 PSVersion 7.4.1
13 PSEdition Core
14 GitCommitId 7.4.1
15 OS Microsoft Windows 10.0.22631
16 Platform Win32NT
17 PSCompatibleVersions {1.0, 2.0, 3.0, 4.0}
18 PSRemotingProtocolVersion 2.3
19 SerializationVersion 1.1.0.1
20 WSManStackVersion 3.0
21
22 .CURRENTHOST
23 Name : Visual Studio Code Host
24 Version : 2024.0.0
25 InstanceId : 9321a3f5-7ce6-455a-ba28-09efa29ebcaf
26 UI : System.Management.Automation.Internal.Host.Int
27 CurrentCulture : en-US
28 CurrentUICulture : en-US
29 PrivateData : Microsoft.PowerShell.ConsoleHost\ConsoleColorP
30 DebuggerEnabled : True
31 IsRunspacePushed : False
32 Runspace : System.Management.Automation.Runspaces.LocalRu
33
34 .DESCRIPTION
35 This script automates various system administration tasks includi
36 which included (adding, updating, enabling, disabling and deleting u
37 management, system downtime prevention, software updates and patch
38 NOTE: Required modules need to be installed and properly configur
39 #
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
83
```

System Audit: The third option is a system audit, going through every single file to check for permissions and for the creation of new user accounts, which takes around 30 minutes to a few hours depending on the system.

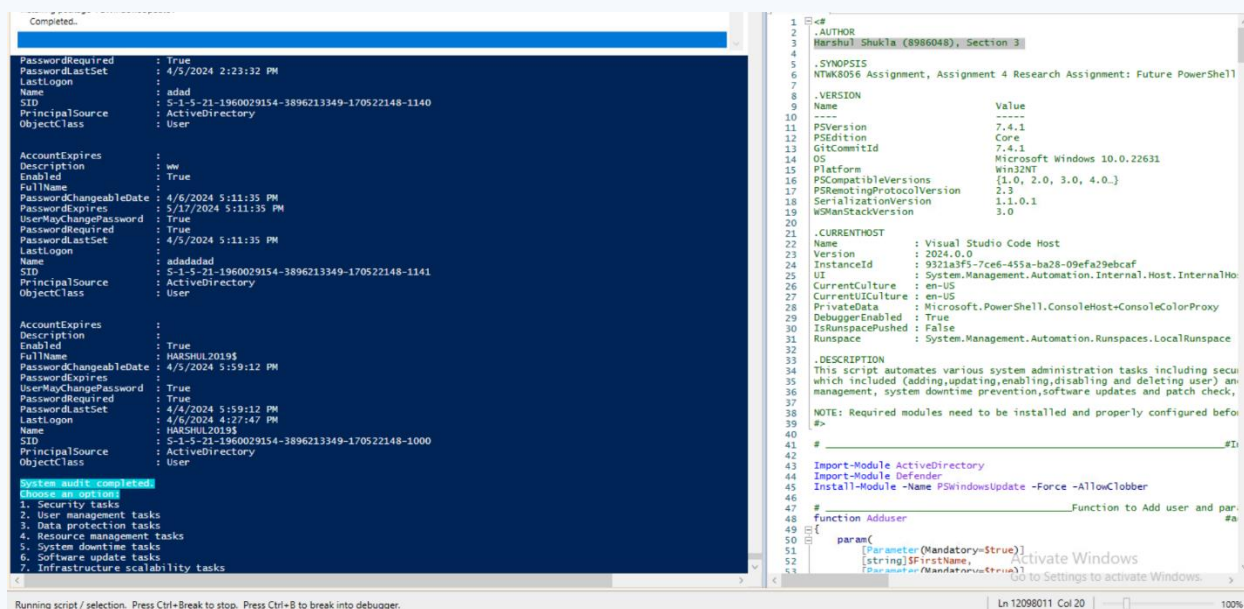


Figure 9 System Audit

Monitor System Logs: Checks for security threats and unauthorized logins.

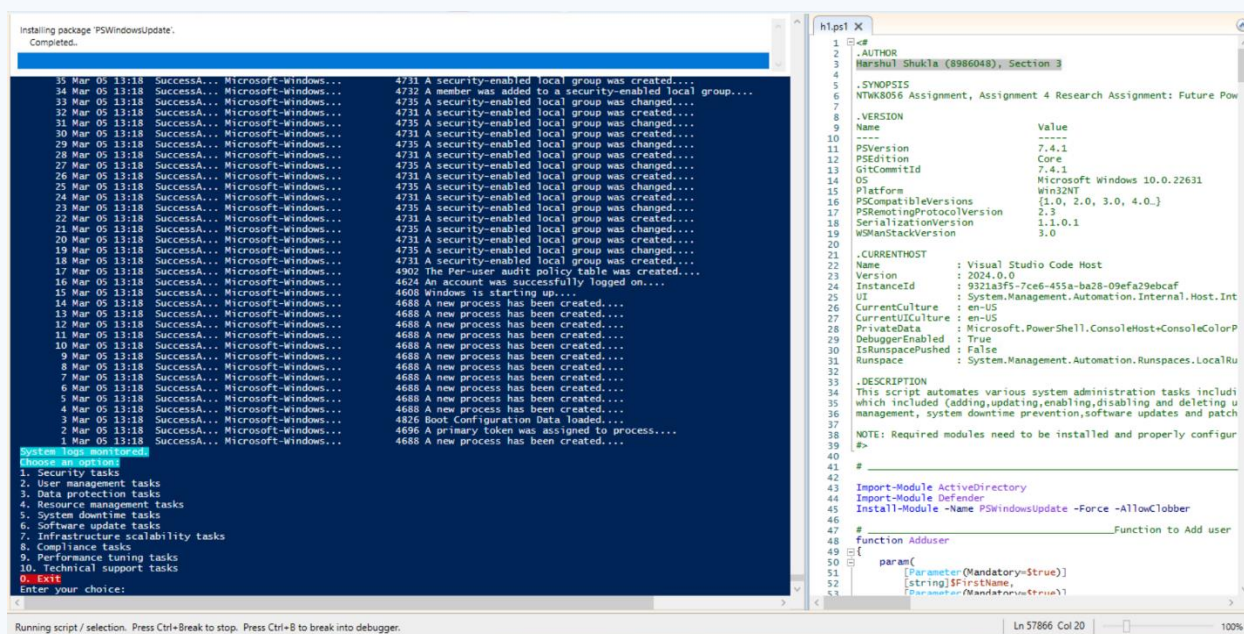


Figure 10 Monitoring System logs.

System Update: Checks for system updates and reboots if necessary

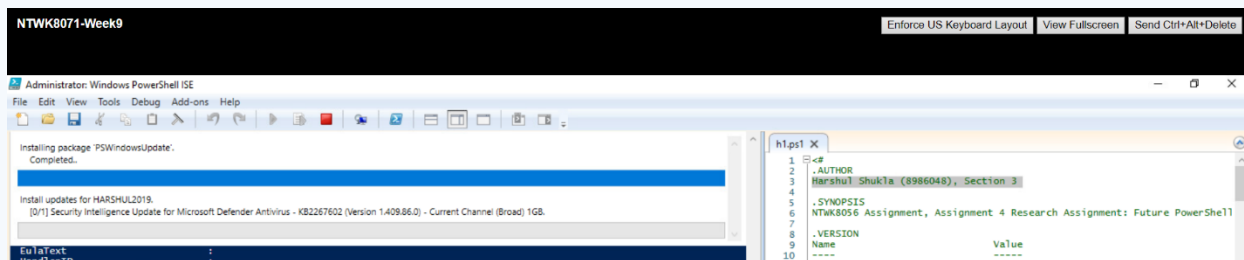


Figure 11 System Update in Process

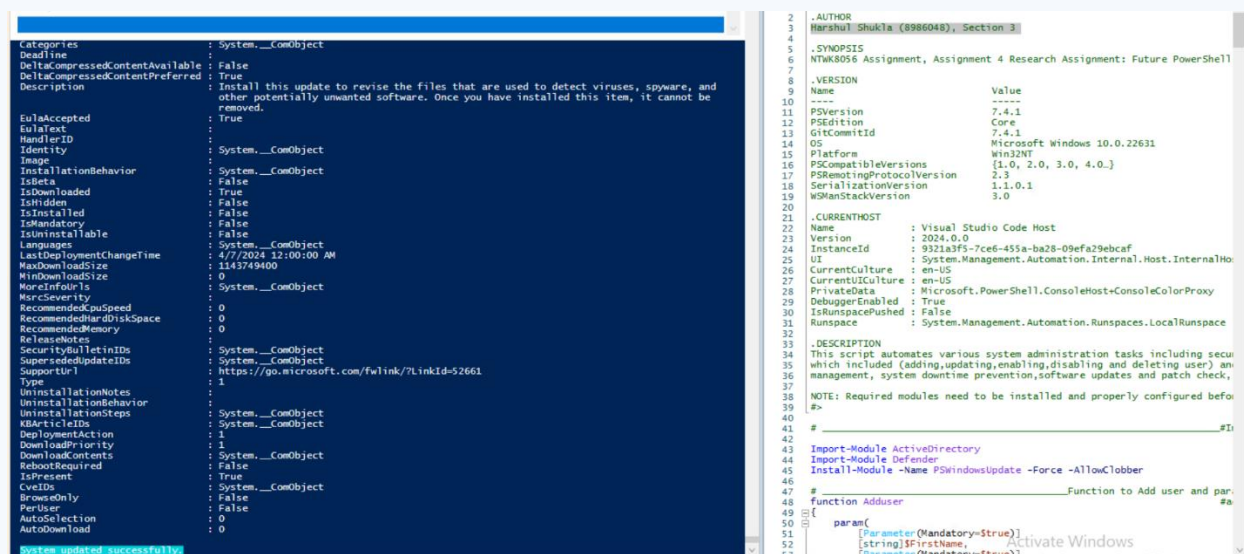


Figure 12 System Updated

USER MANAGEMENT

The script will prompt the user to choose from user-related options and will automatically send an email as soon as the function is executed.

Add User: The script will ask to enter new user details and check if the information is up to standard or not like email and password.

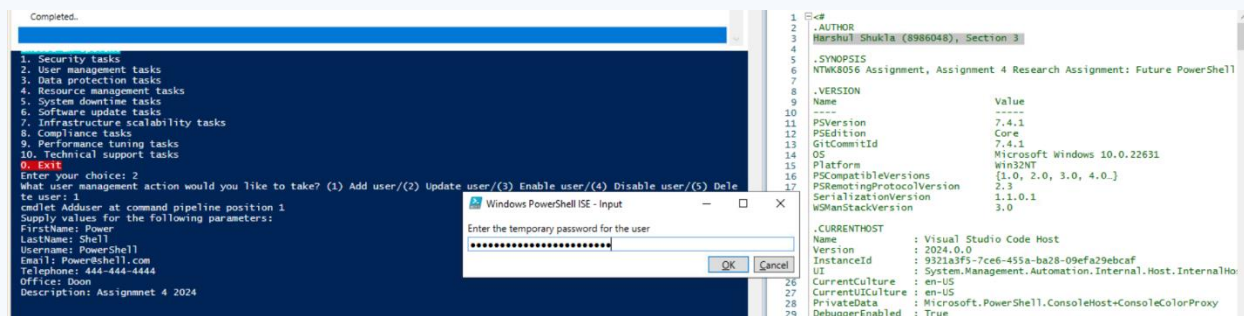


Figure 13 Add user information.

Send Email: Once the user is created email will be sent to HR with the details.

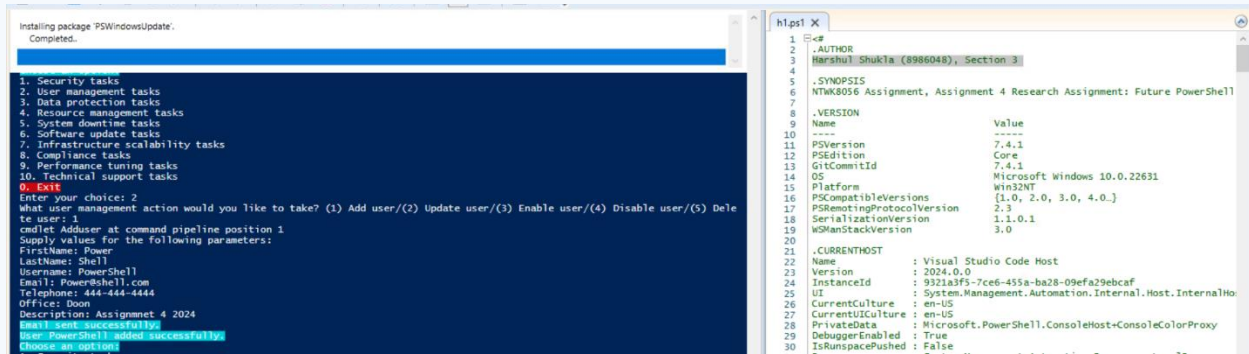


Figure 14 The user-added, and an email was sent automatically.

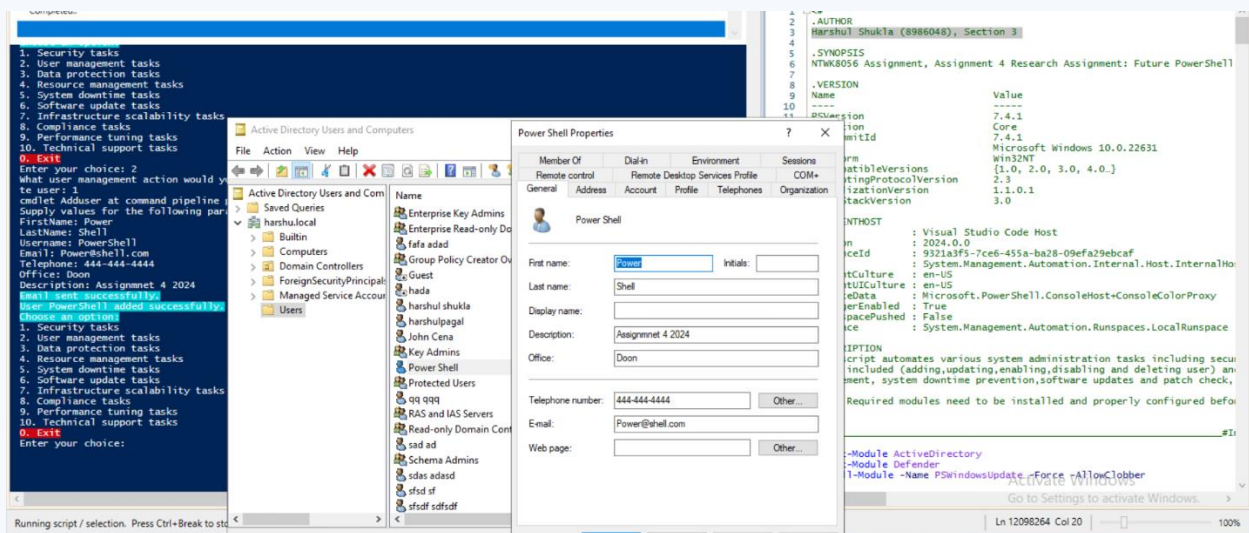


Figure 15 User added in ADUC.

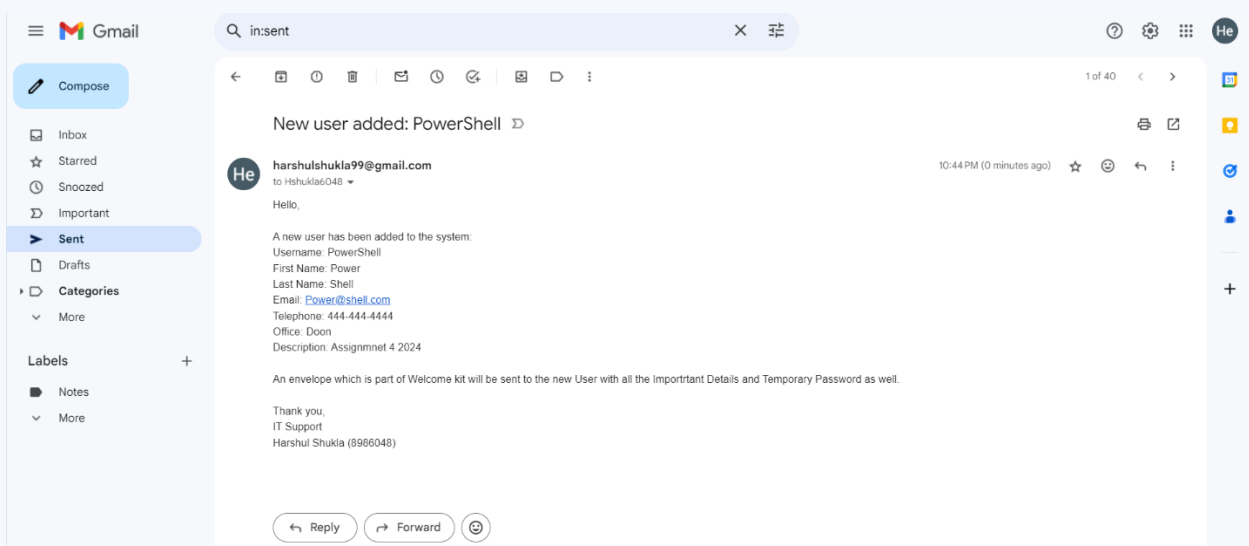


Figure 16 Email sent about the added user.

Automating Security and Active Directory User Management with PowerShell

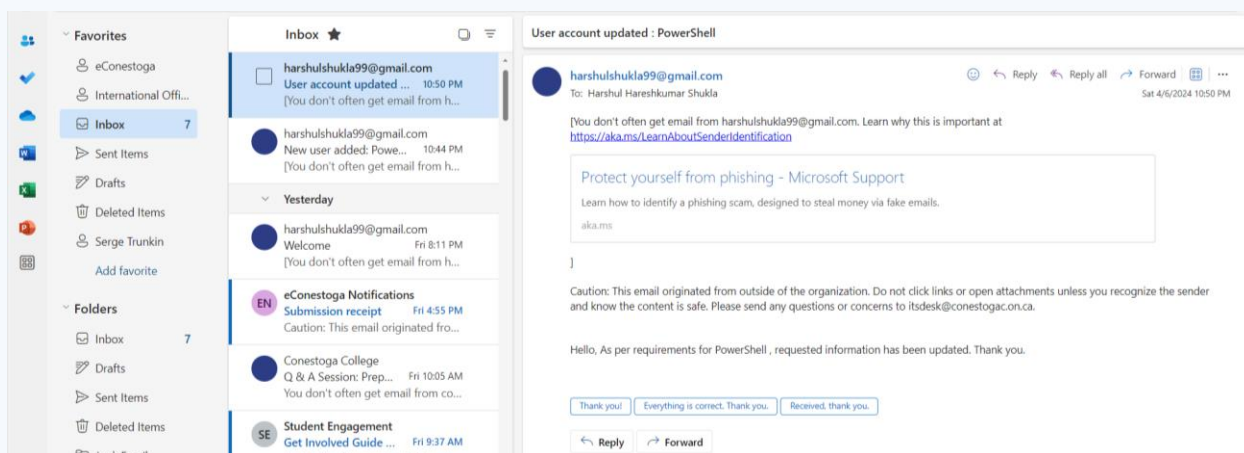


Figure 20 Email received about the updated user.

Disable User: The script will ask you to enter user details and once you enter the details user account will be disabled and an automatic email will be sent to HR.

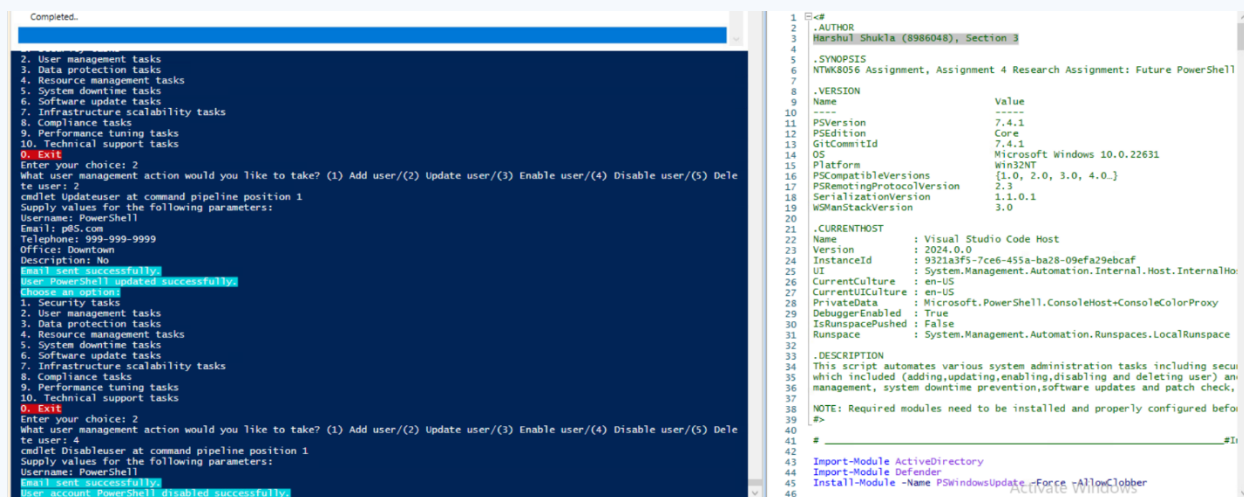


Figure 21 The user account is disabled.

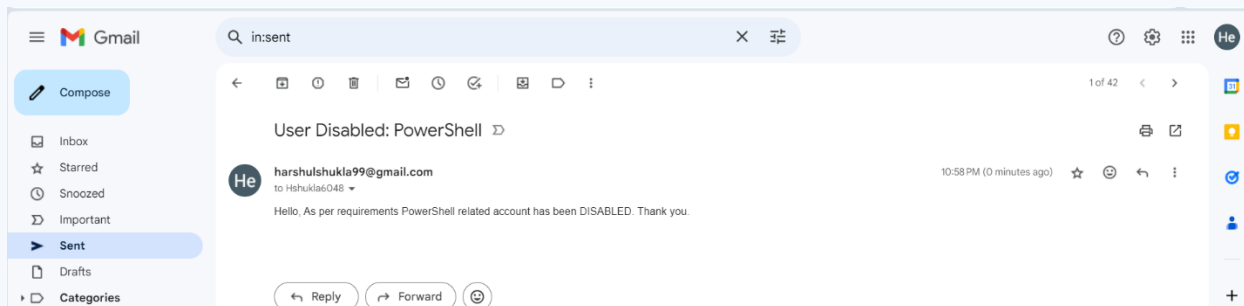


Figure 22 Email sent about the disabled user.

Automating Security and Active Directory User Management with PowerShell

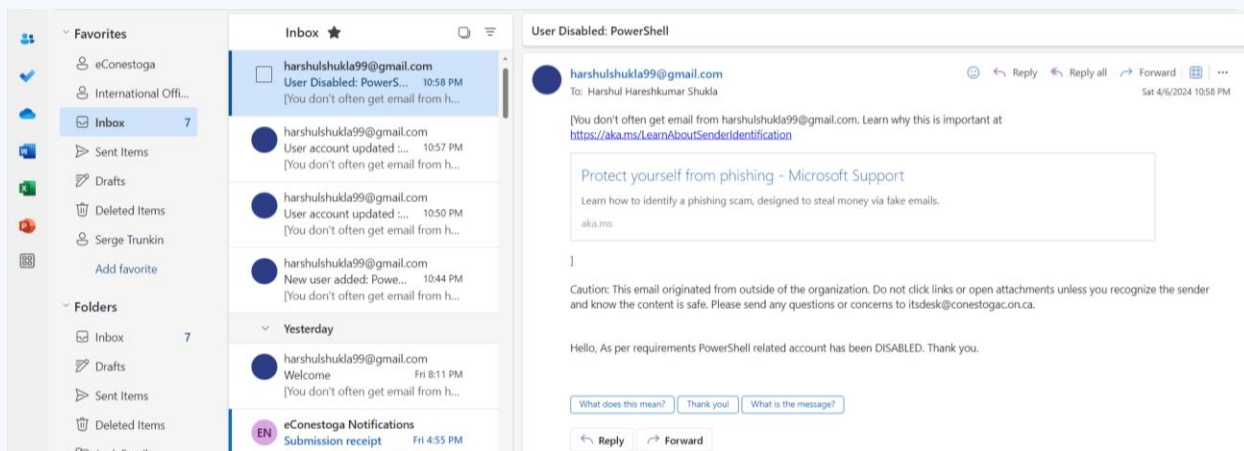


Figure 23 Email received about the disabled user.

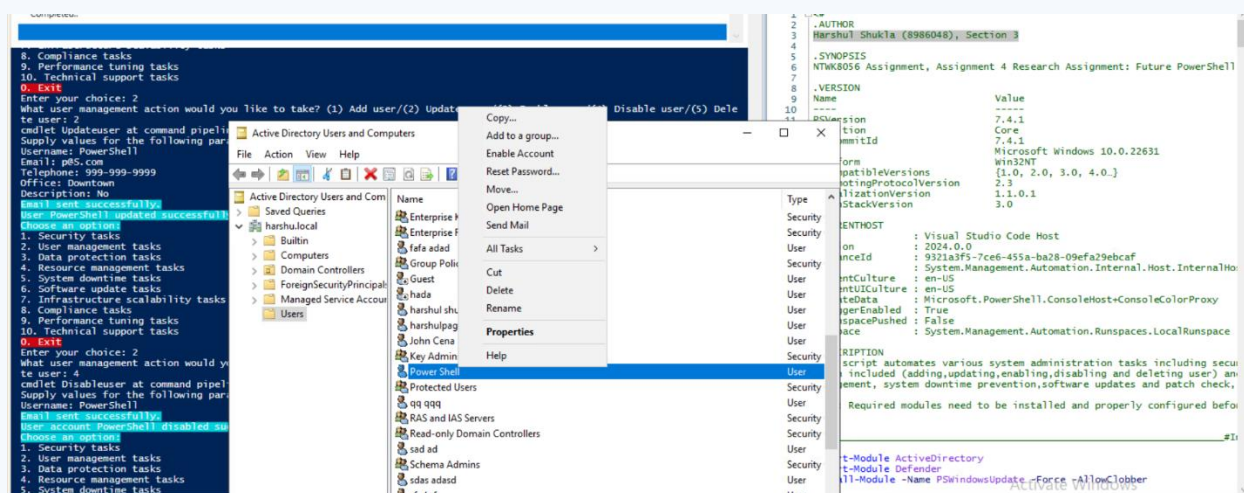


Figure 24 options to enable means the user is disabled.

Enable User: The script will ask you to enter user details and once you enter the details user account will be Enabled and an automatic email will be sent to HR.

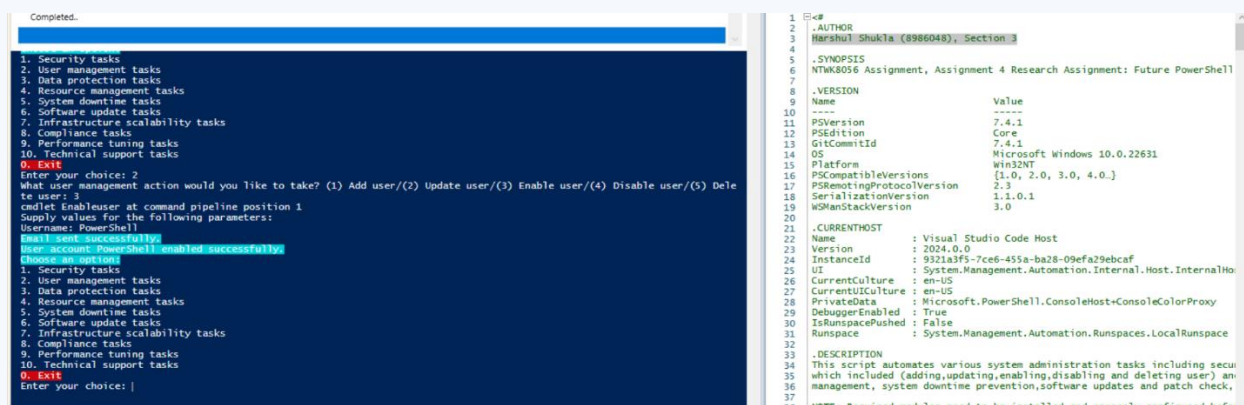


Figure 25 User account enabled.

Automating Security and Active Directory User Management with PowerShell

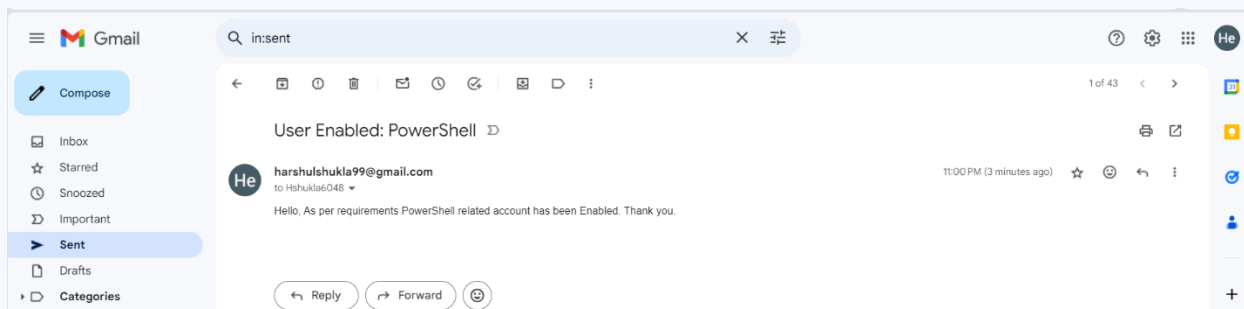


Figure 26 Email Sent about the enabled user.

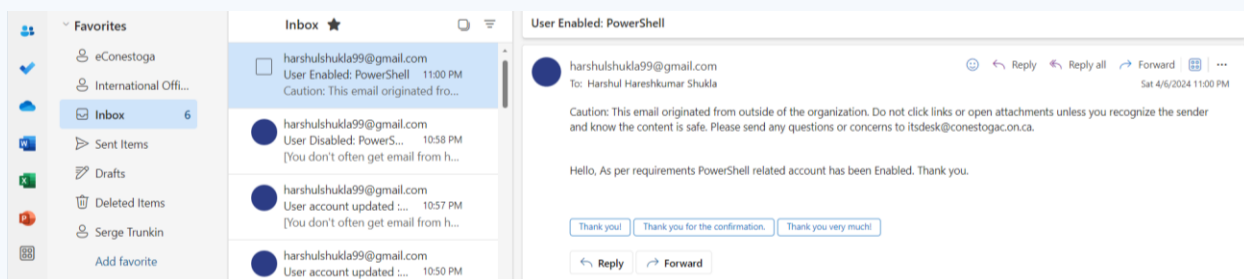


Figure 27 Email received about the enabled user.

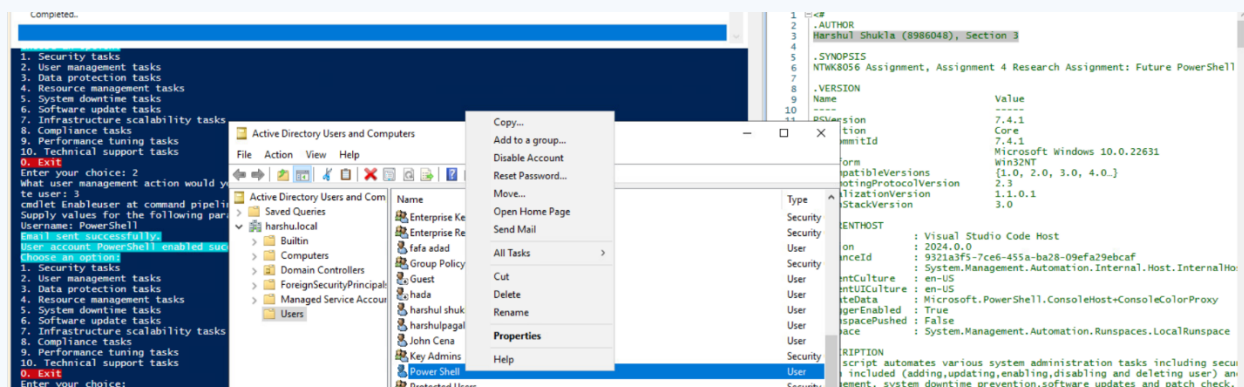


Figure 28 options to disable is showing means the user is enabled.

Delete User: The user account will be disabled, and an email will be sent to HR.

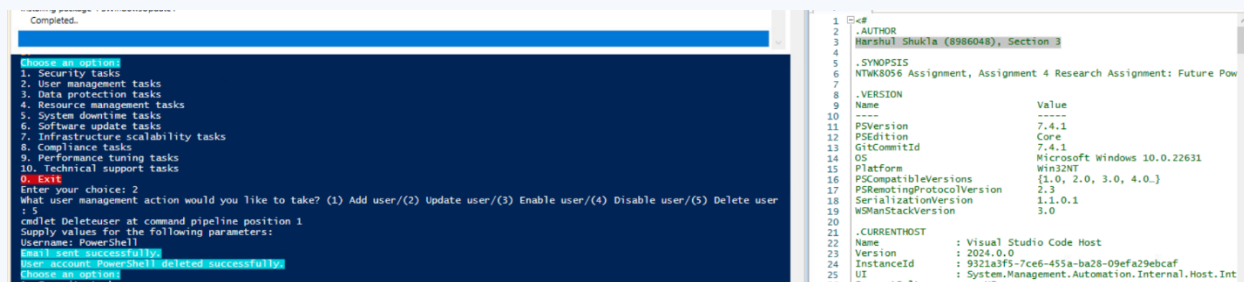


Figure 29 The user account is deleted.

Automating Security and Active Directory User Management with PowerShell

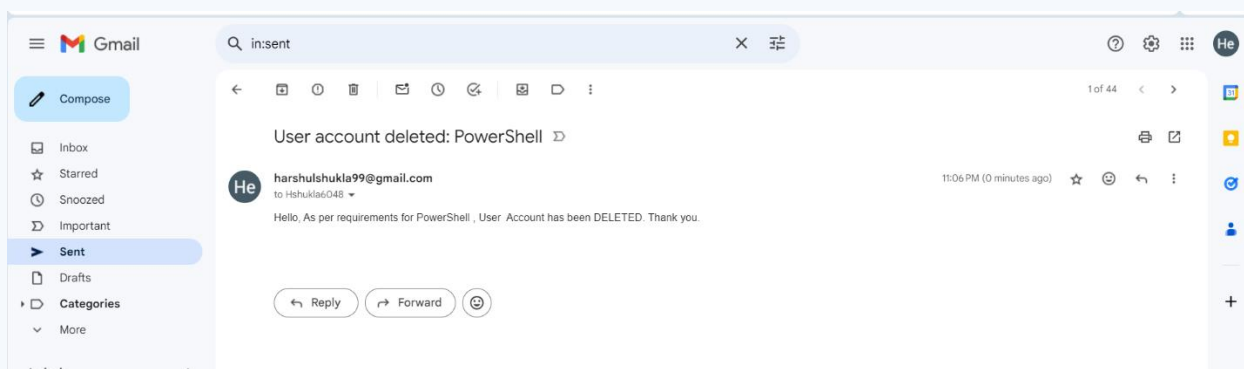


Figure 30 Email sent about the enabled user.

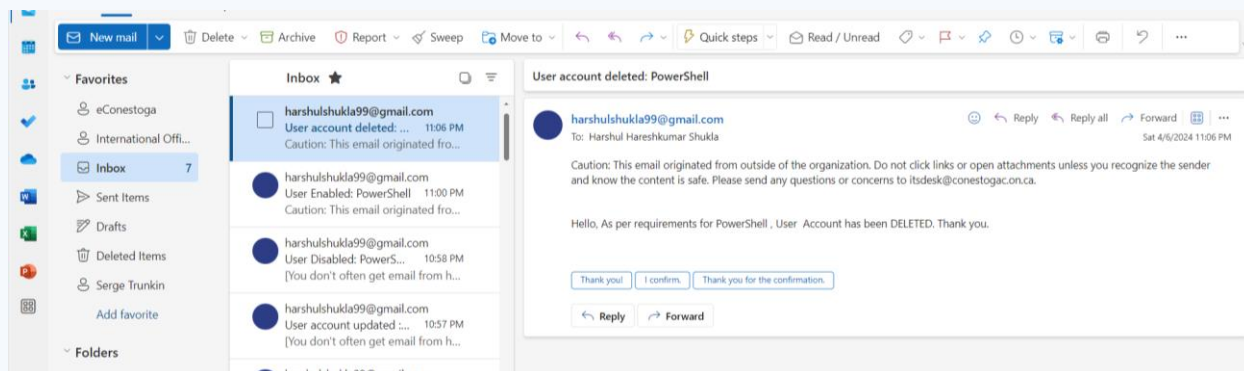


Figure 31 Email received about the deleted user.

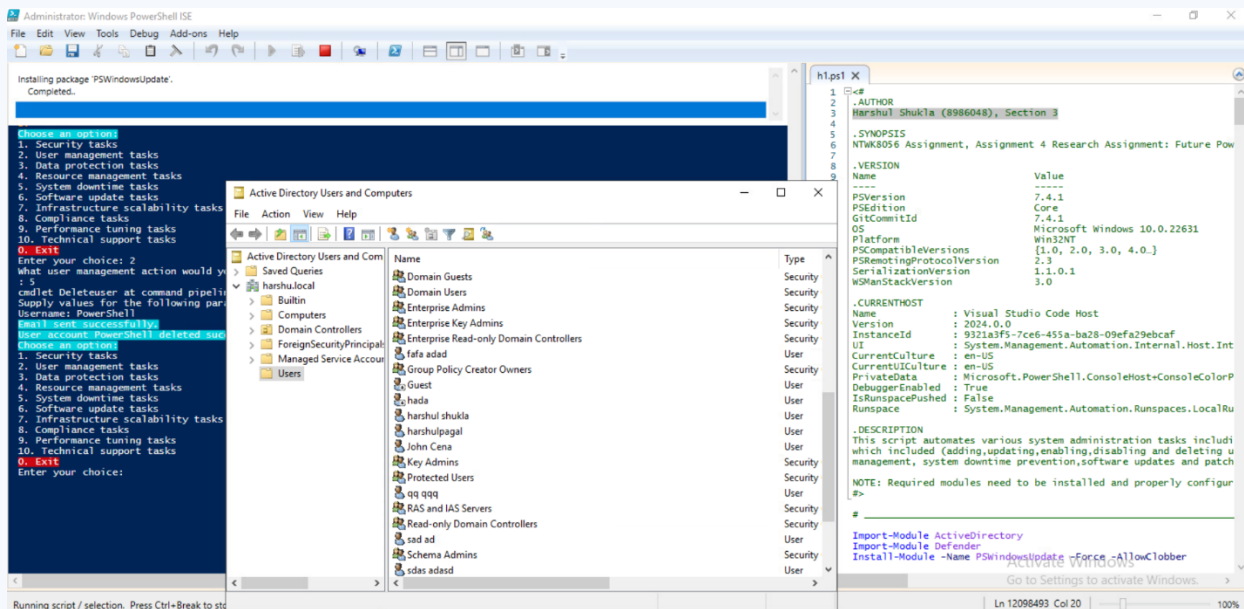


Figure 32 The user account is deleted as it can't be found in ADUC.

TASK MANAGEMENT

Data Protection: Backup will be taken immediately based on the source and the destination of the files.

```

1. Security tasks
2. User management tasks
3. Data protection tasks
4. Resource management tasks
5. System downtime tasks
6. Software update tasks
7. Infrastructure scalability tasks
8. Compliance tasks
9. Performance tuning tasks
10. Technical support tasks
0. Exit
Enter your choice: 3
What data protection action would you like to take? (1) Backup data: 1
Enter the source path for backup: C:\abc
Enter the destination path for backup: C:\xyz
Data backed up successfully

```

Figure 33 Data protection.

Optimize resource allocation: Checks memory usage and provides details.

```

1. Security tasks
2. User management tasks
3. Data protection tasks
4. Resource management tasks
5. System downtime tasks
6. Software update tasks
7. Infrastructure scalability tasks
8. Compliance tasks
9. Performance tuning tasks
10. Technical support tasks
0. Exit
Enter your choice: 4
What resource management action would you like to take? (1) Optimize resource allocation: 1
Low memory, please try and close unnecessary apps

```

Figure 34 Resource optimization.

Prevent system downtime: schedule a reboot and send an email regarding it.

```

1. Security tasks
2. User management tasks
3. Data protection tasks
4. Resource management tasks
5. System downtime tasks
6. Software update tasks
7. Infrastructure scalability tasks
8. Compliance tasks
9. Performance tuning tasks
10. Technical support tasks
0. Exit
Enter your choice: 5
What system downtime action would you like to take? (1) Prevent system downtime: 1
The scheduled task for system downtime prevention WeeklyReboot already exists. No action taken.
Choose an option:
1. Security tasks
2. User management tasks
3. Data protection tasks
4. Resource management tasks
5. System downtime tasks
6. Software update tasks
7. Infrastructure scalability tasks
8. Compliance tasks
9. Performance tuning tasks
10. Technical support tasks
0. Exit
Enter your choice: 1
PS C:\Windows\system32> C:\Users\JohnCena\Desktop\hl.ps1
Choose an option:
1. Security tasks
2. User management tasks
3. Data protection tasks
4. Resource management tasks
5. System downtime tasks
6. Software update tasks
7. Infrastructure scalability tasks
8. Compliance tasks
9. Performance tuning tasks
10. Technical support tasks
0. Exit
Enter your choice: 5
What system downtime action would you like to take? (1) Prevent system downtime: 1
TaskPath          TaskName          State
-----
WeeklyReboot      Ready
Email sent successfully
System downtime prevention scheduled.

```

Figure 35 A task is scheduled, and an email is sent, if a task exists it will notify, and no email will be sent.

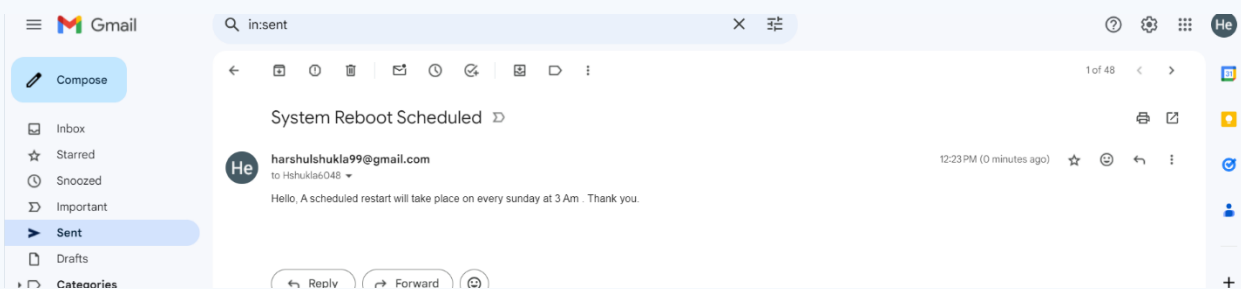


Figure 36 Email sent regarding the reboot.

The screenshot shows the Outlook interface. On the left is the navigation pane with 'Inbox' selected. The main pane displays a list of emails. The selected email is from 'harshulshukla99@gmail.com' with the subject 'System Reboot Sched...' and a timestamp of '12:23 PM'. The email body contains a warning: 'Caution: This email originated from outside of the organization. Do not click links or open attachments unless you recognize the sender and the content is safe. Please send any questions or concerns to ltsdesk@conestogac.on.ca.' Below the email list, a preview of the selected email is visible, showing the same warning and a message: 'Hello, A scheduled restart will take place on every sunday at 3 Am , Thank you.' At the bottom of the preview, there are buttons for 'Confirmed, thank you.', 'Thank you!', and 'Ok, thanks for letting me know.'.

The screenshot displays the Windows Task Scheduler interface and a PowerShell console window.

Task Scheduler Library:

Name	Status	Triggers
CreateExplor...	Ready	When the task is created or modified
GoogleUpda...	Ready	Multiple triggers defined
GoogleUpda...	Ready	At 4:03 PM every day - After triggered, repeat every 1 hour for a dura...
MicrosoftEd...	Ready	Multiple triggers defined
MicrosoftEd...	Ready	At 8:00 AM every day - After triggered, repeat every 1 hour for a dura...
User_Feed_S...	Ready	At 3:20 PM every day - Trigger expires at 4/7/2034 3:20:50 PM.
WeeklyReboot	Ready	At 3:00 AM every Sunday of every week, starting 4/7/2024

Task Properties (WeeklyReboot):

- Name: CreateExplorerShellUnelevatedTask
- Location: \
- Author: ExplorerShellUnelevated
- Description:
- Security options:

Task Actions:

- Run: Run
- Disable
- Export...
- DebuggerEnabled
- IsRunspacePushed
- Runspace
- Delete

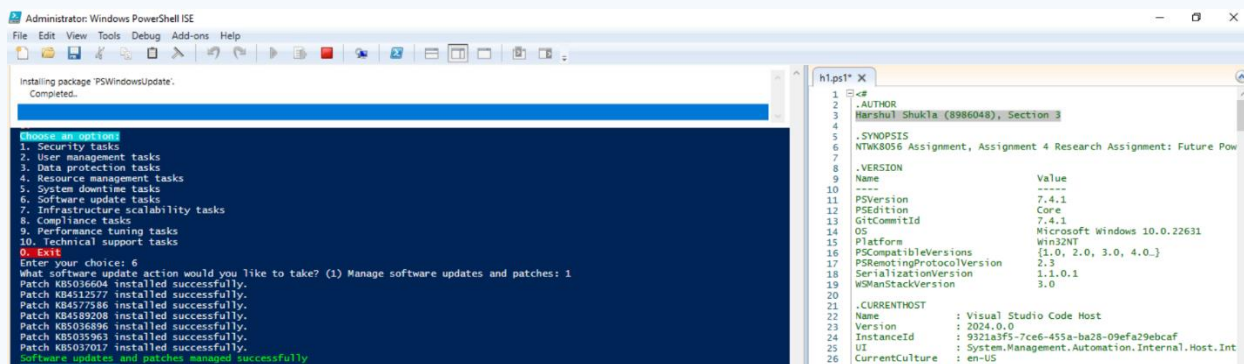
Task Status: WeeklyReboot Ready

PowerShell Console Output:

```

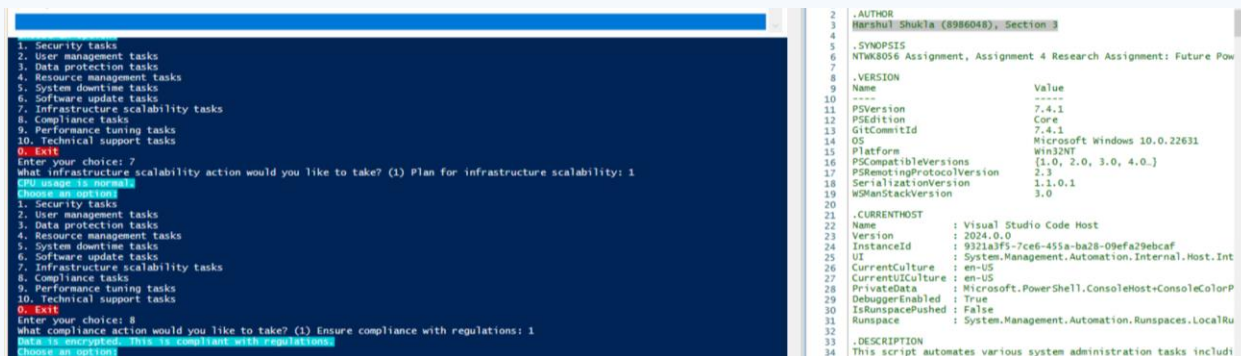
Mail sent successfully.
System downtime prevention scheduled.
Choose an action:
1. Security tasks
2. User management tasks
3. Data protection tasks
4. Resource management tasks
5. System downtime tasks
6. Add new tasks
7. Remove tasks
8. Refresh
9. Help
10. Exit
11. Cancel
12. Back
13. Forward
14. Home
15. Search
16. Help
17. About
18. Exit
19. Cancel
20. Back
21. Forward
22. Home
23. Search
24. Help
25. About
26. Exit
27. Cancel
28. Back
29. Forward
30. Home
31. Search
32. Help
33. About
34. Exit
35. Cancel
36. Back
37. Forward
38. Home
39. Search
40. Help
41. About
42. Exit
43. Cancel
44. Back
45. Forward
46. Home
47. Search
48. Help
49. About
50. Exit
51. Cancel
52. Back
53. Forward
54. Home
55. Search
56. Help
57. About
58. Exit
59. Cancel
60. Back
61. Forward
62. Home
63. Search
64. Help
65. About
66. Exit
67. Cancel
68. Back
69. Forward
70. Home
71. Search
72. Help
73. About
74. Exit
75. Cancel
76. Back
77. Forward
78. Home
79. Search
80. Help
81. About
82. Exit
83. Cancel
84. Back
85. Forward
86. Home
87. Search
88. Help
89. About
90. Exit
91. Cancel
92. Back
93. Forward
94. Home
95. Search
96. Help
97. About
98. Exit
99. Cancel
100. Back
101. Forward
102. Home
103. Search
104. Help
105. About
106. Exit
107. Cancel
108. Back
109. Forward
110. Home
111. Search
112. Help
113. About
114. Exit
115. Cancel
116. Back
117. Forward
118. Home
119. Search
120. Help
121. About
122. Exit
123. Cancel
124. Back
125. Forward
126. Home
127. Search
128. Help
129. About
130. Exit
131. Cancel
132. Back
133. Forward
134. Home
135. Search
136. Help
137. About
138. Exit
139. Cancel
140. Back
141. Forward
142. Home
143. Search
144. Help
145. About
146. Exit
147. Cancel
148. Back
149. Forward
150. Home
151. Search
152. Help
153. About
154. Exit
155. Cancel
156. Back
157. Forward
158. Home
159. Search
160. Help
161. About
162. Exit
163. Cancel
164. Back
165. Forward
166. Home
167. Search
168. Help
169. About
170. Exit
171. Cancel
172. Back
173. Forward
174. Home
175. Search
176. Help
177. About
178. Exit
179. Cancel
180. Back
181. Forward
182. Home
183. Search
184. Help
185. About
186. Exit
187. Cancel
188. Back
189. Forward
190. Home
191. Search
192. Help
193. About
194. Exit
195. Cancel
196. Back
197. Forward
198. Home
199. Search
200. Help
201. About
202. Exit
203. Cancel
204. Back
205. Forward
206. Home
207. Search
208. Help
209. About
210. Exit
211. Cancel
212. Back
213. Forward
214. Home
215. Search
216. Help
217. About
218. Exit
219. Cancel
220. Back
221. Forward
222. Home
223. Search
224. Help
225. About
226. Exit
227. Cancel
228. Back
229. Forward
230. Home
231. Search
232. Help
233. About
234. Exit
235. Cancel
236. Back
237. Forward
238. Home
239. Search
240. Help
241. About
242. Exit
243. Cancel
244. Back
245. Forward
246. Home
247. Search
248. Help
249. About
250. Exit
251. Cancel
252. Back
253. Forward
254. Home
255. Search
256. Help
257. About
258. Exit
259. Cancel
260. Back
261. Forward
262. Home
263. Search
264. Help
265. About
266. Exit
267. Cancel
268. Back
269. Forward
270. Home
271. Search
272. Help
273. About
274. Exit
275. Cancel
276. Back
277. Forward
278. Home
279. Search
280. Help
281. About
282. Exit
283. Cancel
284. Back
285. Forward
286. Home
287. Search
288. Help
289. About
290. Exit
291. Cancel
292. Back
293. Forward
294. Home
295. Search
296. Help
297. About
298. Exit
299. Cancel
300. Back
301. Forward
302. Home
303. Search
304. Help
305. About
306. Exit
307. Cancel
308. Back
309. Forward
310. Home
311. Search
312. Help
313. About
314. Exit
315. Cancel
316. Back
317. Forward
318. Home
319. Search
320. Help
321. About
322. Exit
323. Cancel
324. Back
325. Forward
326. Home
327. Search
328. Help
329. About
330. Exit
331. Cancel
332. Back
333. Forward
334. Home
335. Search
336. Help
337. About
338. Exit
339. Cancel
340. Back
341. Forward
342. Home
343. Search
344. Help
345. About
346. Exit
347. Cancel
348. Back
349. Forward
350. Home
351. Search
352. Help
353. About
354. Exit
355. Cancel
356. Back
357. Forward
358. Home
359. Search
360. Help
361. About
362. Exit
363. Cancel
364. Back
365. Forward
366. Home
367. Search
368. Help
369. About
370. Exit
371. Cancel
372. Back
373. Forward
374. Home
375. Search
376. Help
377. About
378. Exit
379. Cancel
380. Back
381. Forward
382. Home
383. Search
384. Help
385. About
386. Exit
387. Cancel
388. Back
389. Forward
390. Home
391. Search
392. Help
393. About
394. Exit
395. Cancel
396. Back
397. Forward
398. Home
399. Search
400. Help
401. About
402. Exit
403. Cancel
404. Back
405. Forward
406. Home
407. Search
408. Help
409. About
410. Exit
411. Cancel
412. Back
413. Forward
414. Home
415. Search
416. Help
417. About
418. Exit
419. Cancel
420. Back
421. Forward
422. Home
423. Search
424. Help
425. About
426. Exit
427. Cancel
428. Back
429. Forward
430. Home
431. Search
432. Help
433. About
434. Exit
435. Cancel
436. Back
437. Forward
438. Home
439. Search
440. Help
441. About
442. Exit
443. Cancel
444. Back
445. Forward
446. Home
447. Search
448. Help
449. About
450. Exit
451. Cancel
452. Back
453. Forward
454. Home
455. Search
456. Help
457. About
458. Exit
459. Cancel
460. Back
461. Forward
462. Home
463. Search
464. Help
465. About
466. Exit
467. Cancel
468. Back
469. Forward
470. Home
471. Search
472. Help
473. About
474. Exit
475. Cancel
476. Back
477. Forward
478. Home
479. Search
480. Help
481. About
482. Exit
483. Cancel
484. Back
485. Forward
486. Home
487. Search
488. Help
489. About
490. Exit
491. Cancel
492. Back
493. Forward
494. Home
495. Search
496. Help
497. About
498. Exit
499. Cancel
500. Back
501. Forward
502. Home
503. Search
504. Help
505. About
506. Exit
507. Cancel
508. Back
509. Forward
510. Home
511. Search
512. Help
513. About
514. Exit
515. Cancel
516. Back
517. Forward
518. Home
519. Search
520. Help
521. About
522. Exit
523. Cancel
524. Back
525. Forward
526. Home
527. Search
528. Help
529. About
530. Exit
531. Cancel
532. Back
533. Forward
534. Home
535. Search
536. Help
537. About
538. Exit
539. Cancel
540. Back
541. Forward
542. Home
543. Search
544. Help
545. About
546. Exit
547. Cancel
548. Back
549. Forward
550. Home
551. Search
552. Help
553. About
554. Exit
555. Cancel
556. Back
557. Forward
558. Home
559. Search
560. Help
561. About
562. Exit
563. Cancel
564. Back
565. Forward
566. Home
567. Search
568. Help
569. About
570. Exit
571. Cancel
572. Back
573. Forward
574. Home
575. Search
576. Help
577. About
578. Exit
579. Cancel
580. Back
581. Forward
582. Home
583. Search
584. Help
585. About
586. Exit
587. Cancel
588. Back
589. Forward
590. Home
591. Search
592. Help
593. About
594. Exit
595. Cancel
596. Back
5
```

Manage software updates and patches: Update and Create logs.



CONESTOGA
Connect Life and Learning

Infrastructure scalability and compliance with regulations: Checks CPU usage and notifies the user and also checks data in compliance with regulations.



```

1. Security tasks
2. User management tasks
3. Data protection tasks
4. Resource management tasks
5. System downtime tasks
6. Software update tasks
7. Infrastructure scalability tasks
8. Compliance tasks
9. Performance tuning tasks
10. Technical support tasks
0. EXIT
Enter your choice: 7
What infrastructure scalability action would you like to take? (1) Plan for infrastructure scalability: 1
CPU usage is normal
Choose an option:
1. Security tasks
2. User management tasks
3. Data protection tasks
4. Resource management tasks
5. System downtime tasks
6. Software update tasks
7. Infrastructure scalability tasks
8. Compliance tasks
9. Performance tuning tasks
10. Technical support tasks
0. EXIT
Enter your choice: 8
What compliance action would you like to take? (1) Ensure compliance with regulations: 1
Data is compliant, this is compliant with regulations
Choose an option:

```

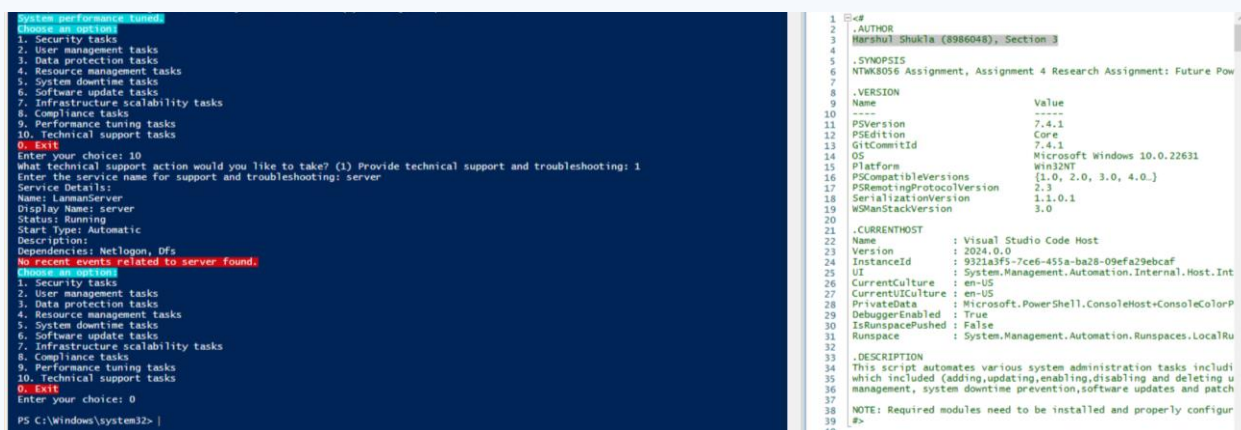
```

2 .AUTHOR
3 Harshul Shukla (8986048), Section 3
4
5 .SYNOPSIS
6 NTWK8056 Assignment, Assignment 4 Research Assignment: Future Pow
7
8 .VERSION
9 ----
10 Name Value
11 PSVersion 7.4.1
12 PSDiption Core
13 GitCommitId 7.4.1
14 OS Microsoft Windows 10.0.22631
15 Platform Win32NT
16 PSCompatibleVersions {1.0, 2.0, 3.0, 4.0.}
17 PSRemotingProtocolVersion 2.3
18 SerializationVersion 1.1.0.1
19 WSManStackVersion 3.0
20
21 .CURRENTHOST
22 Name : Visual Studio Code Host
23 Version : 2024.0.0
24 InstanceId : 9321a3f5-7ce6-455a-ba28-09efa29ebcaf
25 UI : System.Management.Automation.Internal.Host.Int
26 CurrentCulture : en-US
27 CurrentUICulture : en-US
28 PrivateData : Microsoft.PowerShell.ConsoleHost+ConsoleColorP
29 DebuggerEnabled : True
30 IsRunspacePushed : False
31 Runspace : System.Management.Automation.Runspaces.LocalRu
32
33 .DESCRIPTION
34 This script automates various system administration tasks includi
35

```

Figure 40 Infrastructure scalability and compliance with regulations.

Tech support and troubleshooting: enter the service name for support and troubleshooting and it shows details.



```

System performance tuned.
Choose an option:
1. Security tasks
2. User management tasks
3. Data protection tasks
4. Resource management tasks
5. System downtime tasks
6. Software update tasks
7. Infrastructure scalability tasks
8. Compliance tasks
9. Performance tuning tasks
10. Technical support tasks
0. EXIT
Enter your choice: 10
What technical support action would you like to take? (1) Provide technical support and troubleshooting: 1
Enter the service name for support and troubleshooting: server
Service Details:
Name: LanmanServer
Display Name: server
Status: Running
Start Type: Automatic
Description:
Dependencies: Netlogon, Dfs
No recent events related to server found.
Choose an option:
1. Security tasks
2. User management tasks
3. Data protection tasks
4. Resource management tasks
5. System downtime tasks
6. Software update tasks
7. Infrastructure scalability tasks
8. Compliance tasks
9. Performance tuning tasks
10. Technical support tasks
0. EXIT
Enter your choice: 0
PS C:\Windows\system32> |

```

```

1 <#
2 .AUTHOR
3 Harshul Shukla (8986048), Section 3
4
5 .SYNOPSIS
6 NTWK8056 Assignment, Assignment 4 Research Assignment: Future Pow
7
8 .VERSION
9 ----
10 Name Value
11 PSVersion 7.4.1
12 PSDiption Core
13 GitCommitId 7.4.1
14 OS Microsoft Windows 10.0.22631
15 Platform Win32NT
16 PSCompatibleVersions {1.0, 2.0, 3.0, 4.0.}
17 PSRemotingProtocolVersion 2.3
18 SerializationVersion 1.1.0.1
19 WSManStackVersion 3.0
20
21 .CURRENTHOST
22 Name : Visual Studio Code Host
23 Version : 2024.0.0
24 InstanceId : 9321a3f5-7ce6-455a-ba28-09efa29ebcaf
25 UI : System.Management.Automation.Internal.Host.Int
26 CurrentCulture : en-US
27 CurrentUICulture : en-US
28 PrivateData : Microsoft.PowerShell.ConsoleHost+ConsoleColorP
29 DebuggerEnabled : True
30 IsRunspacePushed : False
31 Runspace : System.Management.Automation.Runspaces.LocalRu
32
33 .DESCRIPTION
34 This script automates various system administration tasks includi
35 which included (adding, updating, enabling, disabling and deleting u
36 management, system downtime prevention, software updates and patch
37
38 NOTE: Required modules need to be installed and properly configur
39
40 #>

```

Figure 41 Troubleshooting.

SCRIPT FUNCTIONS INFORMATION TABLE

The function is a block of code that performs a specific task when you execute it.

Table 1. Function Information

Function Name	Purpose	Input Parameters	Output
Adduser	Adds a new user to the Active Directory	FirstName, LastName, Username, Email, Telephone, Office, Description	Confirmation of user addition
Updateuser	Updates an existing user's details in Active Directory	Username, Email, Telephone, Office, Description	Confirmation of user update
Enableuser	Enables a disabled user account in the Active Directory	Username	Confirmation of user enablement
Disableuser	Disables an active user account in the Active Directory	Username	Confirmation of user disablement
Deleteuser	Deletes a user account from the Active Directory	Username	Confirmation of user deletion
Scanmalware	Initiates a malware scan	None	Confirmation of scan completion
Firewallrules	Manages firewall rules	Port	Confirmation of rule management
Systemaudit	Performs a system audit	None	Confirmation of audit completion
MonitorSystemlogs	Monitors system logs	None	Confirmation of log monitoring
Updatesystem	Updates the system	None	Confirmation of system update
Backupdata	Backs up data	None	Confirmation of data backup
Optimizeresource	Optimizes resource allocation	None	Resource allocation status
Preventssystemdowntime	Schedules system reboots to prevent system downtime	None	Confirmation of scheduled reboot
ManageSoftwareUpdates	Manages software updates and patches	None	Confirmation of software update management
Infrastructurescalability	Plans for infrastructure scalability	None	Scalability status
Compliancewithregulations	Ensures compliance with regulations	None	Compliance status
Tunesystemperformance	Tunes system performance	None	Confirmation of performance tuning
TechnicalsupportAndtroubleshooting	Provides technical support and troubleshooting	None	Confirmation of support provision
All Email functions	Sends Email	None (Inbuilt in code)	Confirmation of email sent

POWERSHELL CODE

Script Guide: Code should be able to run given that required modules are installed and with admin rights, just make sure that in the process of sending email, Mail to and Mail from being changed accordingly, and most importantly make sure 16-digit (xxxx xxxx xxxx xxxx) app-specific password has been generated by you inside your Gmail from which you are sending your email.

Currently code uses my Gmail app-specific password, to check the code you can only change TO which email address it sends email and keep from as it is, it will send email to your email address from my email address.

Change the path to your liking where paths are assigned.

```
<#
.AUTHOR
Harshul Shukla (8986048), Section 3

.SYNOPSIS
NTWK8056 Assignment, Assignment 4 Research Assignment: Future PowerShell Use , Title:
Automating Security and Active Directory User Management with PowerShell

.VERSION
Name                                Value
----                                -
PSVersion                          7.4.1
PSEdition                          Core
GitCommitId                        7.4.1
OS                                  Microsoft Windows 10.0.22631
Platform                           Win32NT
PSCompatibleVersions               {1.0, 2.0, 3.0, 4.0...}
PSRemotingProtocolVersion          2.3
SerializationVersion               1.1.0.1
WSManStackVersion                  3.0

.CURRENTHOST
Name                                : Visual Studio Code Host
Version                            : 2024.0.0
InstanceId                         : 9321a3f5-7ce6-455a-ba28-09efa29ebcaf
UI                                  : System.Management.Automation.Internal.Host.InternalHostUserInterface
CurrentCulture                     : en-US
CurrentUICulture                   : en-US
PrivateData                        : Microsoft.PowerShell.ConsoleHost+ConsoleColorProxy
DebuggerEnabled                    : True
IsRunspacePushed                   : False
Runspace                           : System.Management.Automation.Runspaces.LocalRunspace
```


.DESCRIPTION

This script automates various system administration tasks including security like (scanning for malware, audit, check logs and updates) script also provides user management which included (adding, updating, enabling, disabling and deleting user) and sending email to HR directly from here, it also provides things like data protection, resource management, system downtime prevention, software updates and patch check, infrastructure scalability, compliance, performance tuning, and technical support.

NOTE: Required modules need to be installed and properly configured before running this script and admin access is required.

#>

#Import and install required modules

Import-Module ActiveDirectory

Import-Module Defender

Install-Module -Name PSWindowsUpdate -Force -AllowClobber

Function to Add user and parameters and send Email to HR and a function to validate email (JasonGerend, n.d.)

function Adduser

{

param(

[Parameter(Mandatory=\$true)]

[string]\$FirstName,

[Parameter(Mandatory=\$true)]

[string]\$LastName,

[Parameter(Mandatory=\$true)]

[string]\$Username,

[Parameter(Mandatory=\$true)]

[string]\$Email,

[Parameter(Mandatory=\$true)]

[string]\$Telephone,

[Parameter(Mandatory=\$true)]

[string]\$Office,

[Parameter(Mandatory=\$true)]

[string]\$Description

)

#Parameters

try

{

#Validate email address format

Validatemail -Email \$Email

#Generate a random ID number

\$IDNumber = Get-Random -Minimum 1000 -Maximum 9999

#Ask user for password

\$Password = Read-Host "Enter the temporary password for the user" -AsSecureString

```

#Create user parameters this info will be added in ADUC.
$userParams = @{
    SamAccountName = $Username
    UserPrincipalName = $Email
    Name = "$FirstName $LastName"
    GivenName = $FirstName
    Surname = $LastName
    EmailAddress = $Email
    Description = $Description
    Office = $Office
    OfficePhone = $Telephone
    EmployeeNumber = $IDNumber
    Enabled = $true
    AccountPassword = $Password
}

#Add new user in Active Directory
New-ADUser @userParams
#Send email to HR
Sendemailuseradd -Subject $subject -Body $body
#User added text
Write-Host "User $Username added successfully" -BackgroundColor Cyan
}
catch
{
    #failed to add user text
    Write-Host "Failed to add user: $_" -BackgroundColor Red
}
}

#Function to send Email to HR(Yung, 2024)
function Sendemailuseradd
#Email details
{
    $EmailFrom = "harshulshukla99@gmail.com"
    $EmailTo = "Hshukla6048@conestogac.on.ca"
    $subject = "New user added: $Username"
    #Body paragraph of the email to HR
    $body = @"
Hello,

A new user has been added to the system:
Username: $Username
First Name: $FirstName
Last Name: $LastName
Email: $Email

```

Telephone: \$Telephone
 Office: \$Office
 Description: \$Description

An envelope which is part of Welcome kit will be sent to the new User with all the Important Details and Temporary Password as well.

Thank you,
 IT Support
 Harshul Shukla (8986048)

"@

#Gmail app-specific password, this has to be unique for every sender, this is app specific password for harshulshukla99@gmail.com which can be generated in email settings

`$Appspecificpassword = "pier ptoq xdue ojgz" | ConvertTo-SecureString -AsPlainText -Force`

#Create the credential object

`$Credential = New-Object System.Management.Automation.PSCredential($EmailFrom, $Appspecificpassword)`

#Create the SMTP client

`$SMTPServer = "smtp.gmail.com"`

#smtp server 587 for gmail

`$SMTPClient = New-Object Net.Mail.SmtpClient($SMTPServer, 587)`

#Enable SSL

`$SMTPClient.EnableSsl = $true`

#get network credential

`$SMTPClient.Credentials = $Credential.GetNetworkCredential()`

#Send the email

`try`

`{`

`$SMTPClient.Send($EmailFrom, $EmailTo, $Subject, $Body)`

`Write-Host "Email sent successfully" -BackgroundColor Cyan`

`}`

`catch`

`{`

`Write-Host "Failed to send email: $_" -BackgroundColor Red #Error`

`}`

`}`

#Function to validate email address

`function Validatemail`

`{`

`#parameter`

`param(`

`[Parameter(Mandatory=$true)]`

`[string]$Email`

`)`

`#condition`

```

    if ($Email -notmatch '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$')
    {
        throw "Please provide a valid email address"
    }
}

#Function to enable user account and send Email to HR(JasonGerend, n.d.)
function Enableuser
{
    param(
        [Parameter(Mandatory=$true)]
        [string]$Username
    )
    try
    {
        #Enable the user account
        Enable-ADAccount -Identity $Username
        #Send email to HR
        Sendmailenable -Subject $subject -Body $body
        #text indicating that account is enabled
        Write-Host "User account $Username enabled successfully" -BackgroundColor Cyan
    }
    catch
    {
        Write-Host "Failed to enable user account: $_" -BackgroundColor Red
    }
}

#Function to send Email that it has been enabled(Yung, 2024)
function Sendmailenable
{
    #Gmail credentials
    $EmailFrom = "harshulshukla99@gmail.com"
    $EmailTo = "Hshukla6048@conestogac.on.ca"
    $Subject = "User Enabled: $Username"
    $Body = "Hello, As per requirements $username related account has been Enabled. Thank you."
    #app-specific password
    $Appspecificpassword = "pier ptoq xdue ojgz" | ConvertTo-SecureString -AsPlainText -Force
    # Create the credential object
    $Credential = New-Object System.Management.Automation.PSCredential($EmailFrom,
    $Appspecificpassword)
    # Create the SMTP client
    $SMTPServer = "smtp.gmail.com"
    $SMTPClient = New-Object Net.Mail.SmtpClient($SMTPServer, 587)
    $SMTPClient.EnableSsl = $true
    $SMTPClient.Credentials = $Credential.GetNetworkCredential()
    # Send the email

```

```

try
{
    $SMTPClient.Send($EmailFrom, $EmailTo, $Subject, $Body)
    Write-Host "Email sent successfully." -BackgroundColor Cyan
}
catch
{
    Write-Host "Failed to send email: $_" -BackgroundColor Red
}
}

# Function to update user and send Email to HR(JasonGerend, n.d.)
function Updateuser
{
    param(
        [Parameter(Mandatory=$true)]
        [string]$Username,
        [Parameter(Mandatory=$true)]
        [string]$Email,
        [Parameter(Mandatory=$true)]
        [string]$Telephone,
        [Parameter(Mandatory=$true)]
        [string]$Office,
        [Parameter(Mandatory=$true)]
        [string]$Description
    )
    try
    {
        # Validate email address format
        Validatemail -Email $Email
        #Update user parameters
        $userParams = @{
            EmailAddress = $Email
            Office = $Office
            OfficePhone = $Telephone
            Description = $Description
        }
        #Update the user in AD
        Set-ADUser -Identity $Username @userParams
        #Send email to HR
        Sendemailupdate -Subject $subject -Body $body
        #updated
        Write-Host "User $Username updated successfully." -BackgroundColor Cyan
    }
    catch
    {
        Write-Host "Failed to update user: $_" -BackgroundColor Red
    }
}

```

```

    }
}
#send email regarding update function(Yung, 2024)
function Sendemailupdate
{
    # Define credentials
    $EmailFrom = "harshulshukla99@gmail.com"
    $EmailTo = "Hshukla6048@conestogac.on.ca"
    $Subject = "User account updated : $Username"
    $Body = "Hello, As per requirements for $username , requested information has been updated.
    Thank you."
    #Gmail app-specific password
    $Appspecificpassword = "pier ptoq xdue ojgz" | ConvertTo-SecureString -AsPlainText -Force
    #Create the credential object
    $Credential = New-Object System.Management.Automation.PSCredential($EmailFrom,
    $Appspecificpassword)
    # Create the SMTP client
    $SMTPServer = "smtp.gmail.com"
    $SMTPClient = New-Object Net.Mail.SmtpClient($SMTPServer, 587)
    $SMTPClient.EnableSsl = $true
    $SMTPClient.Credentials = $Credential.GetNetworkCredential()
    #Send the email
    try
    {
        $SMTPClient.Send($EmailFrom, $EmailTo, $Subject, $Body)
        Write-Host "Email sent successfully." -BackgroundColor Cyan
    }
    catch
    {
        Write-Host "Failed to send email: $_" -BackgroundColor Red
    }
}
#Function to disable user account and send Email to HR(JasonGerend, n.d.)
function Disableuser
{
    param(
        [Parameter(Mandatory=$true)]
        [string]$Username
    )
    try
    {
        # Disable the user account
        Disable-ADAccount -Identity $Username
        #Send email to HR
        Sendemaildisable -Subject $subject -Body $body
        Write-Host "User account $Username disabled successfully." -BackgroundColor Cyan
    }
}

```

```

    }
    catch
    {
        Write-Host "Failed to disable user account: $_" -BackgroundColor Red
    }
}

#Function to send email to HR(Yung, 2024)
function Sendemaildisable
{
    # Define your Gmail credentials
    $EmailFrom = "harshulshukla99@gmail.com"
    $EmailTo = "Hshukla6048@conestogac.on.ca"
    $Subject = "User Disabled: $Username"
    $Body = "Hello, As per requirements $username related account has been DISABLED. Thank you."
    #Gmail app-specific password
    $Appspecificpassword = "pier ptoq xdue ojgz" | ConvertTo-SecureString -AsPlainText -Force
    # Create the credential object
    $Credential = New-Object System.Management.Automation.PSCredential($EmailFrom,
    $Appspecificpassword)
    #Create the SMTP client
    $SMTPServer = "smtp.gmail.com"
    $SMTPClient = New-Object Net.Mail.SmtpClient($SMTPServer, 587)
    $SMTPClient.EnableSsl = $true
    $SMTPClient.Credentials = $Credential.GetNetworkCredential()
    #Send the email
    try
    {
        $SMTPClient.Send($EmailFrom, $EmailTo, $Subject, $Body)
        Write-Host "Email sent successfully." -BackgroundColor Cyan
    }
    catch
    {
        Write-Host "Failed to send email: $_" -BackgroundColor Red
    }
}

#Function to delete user account and send Email to HR(JasonGerend, n.d.)
function Deleteuser
{
    param(
        [Parameter(Mandatory=$true)]
        [string]$Username
    )
    # Delete the user account
    try
    {

```

```

    Remove-ADUser -Identity $Username -Confirm:$false
    #Send email to HR
    Sendemaildelete -Subject $subject -Body $body
    Write-Host "User account $Username deleted successfully." -BackgroundColor Cyan
}
catch
{
    Write-Host "Failed to delete user account: $_" -BackgroundColor Red
}
}

#Function to send email(Yung, 2024)
function Sendemaildelete
{
    #Define your Gmail credentials
    $EmailFrom = "harshulshukla99@gmail.com"
    $EmailTo = "Hshukla6048@conestogac.on.ca"
    $Subject = "User account deleted: $Username"
    $Body = "Hello, As per requirements for $username , User Account has been DELETED. Thank you."
    #Gmail app-specific password
    $Appspecificpassword = "pier ptoq xdue ojgz" | ConvertTo-SecureString -AsPlainText -Force
    #Create the credential object
    $Credential = New-Object System.Management.Automation.PSCredential($EmailFrom, $Appspecificpassword)
    # Create the SMTP client
    $SMTPServer = "smtp.gmail.com"
    $SMTPClient = New-Object Net.Mail.SmtpClient($SMTPServer, 587)
    $SMTPClient.EnableSsl = $true
    $SMTPClient.Credentials = $Credential.GetNetworkCredential()
    #Send the email
    try
    {
        $SMTPClient.Send($EmailFrom, $EmailTo, $Subject, $Body)
        Write-Host "Email sent successfully." -BackgroundColor Cyan
    }
    catch
    {
        Write-Host "Failed to send email: $_" -BackgroundColor Red
    }
}

#Function to scan for malware(JasonGerend, n.d.-b)
function Scanmalware
{
    try

```



```

{ # Start a quick malware scan
  Start-MpScan -ScanType QuickScan

  Write-Host "Malware scan completed" -BackgroundColor Cyan
}
catch
{
  Write-Host "Failed to scan for malware: $_" -BackgroundColor Red
}
}

#Function to manage firewall rules (Paolomatarazzo, 2023)
function Firewallrules
{
  #Set execution policy to allow running scripts
  Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
  #Display menu
  Write-Host "Select an action:"
  Write-Host "1. Add a new firewall rule"
  Write-Host "2. List firewall rules"
  Write-Host "3. Remove an existing firewall rule"
  #ask to enter number
  $action = Read-Host "Enter the number of your choice (1, 2, or 3)"
  #Perform action base on user choice
  switch ($action)
  {
    '1'
    { #ask the user to enter a display name and local port for the new firewall rule.
      $display = Read-Host "Enter a display name for the new firewall rule"
      $local = Read-Host "Enter the local port to block"
      try #create a new firewall rule with the parameters
      {
        New-NetFirewallRule -DisplayName $display -Direction Inbound -LocalPort $local -
Protocol TCP -Action Block
        Write-Host "Firewall rule '$display' added successfully" -ForegroundColor Green
      }
      catch
      {
        Write-Host "Failed to add firewall rule: $_" -ForegroundColor Red
      }
    }
    '2'
    {
      # ask user to enter info for a firewall
      $displayfilter = Read-Host "Press enter to see the list or enter info"
    }
  }
}

```

```

        Get-NetFirewallRule | Where-Object { $_.Display -like "$displayfilter*" } | Format-
Table -AutoSize
    }
    '3'
    { # ask the user to enter the display name of the rule to remove rule
        $display = Read-Host "Enter the display name of the firewall rule to remove"
        $Remove = Get-NetFirewallRule -DisplayName $display -ErrorAction SilentlyContinue
        if ($null -ne $Remove)
        {
            try
            { #remove the specified rule
                Remove-NetFirewallRule -DisplayName
$display
                Write-Host "Firewall rule '$display' removed successfully" -ForegroundColor
Green
            }
            catch
            {
                Write-Host "Failed to remove firewall rule '$display': $_" -ForegroundColor
Red
            }
        }
        else
        {
            Write-Host "Firewall rule '$display' not found" -ForegroundColor
Yellow
        }
    }
    Default
    {
        Write-Host "Invalid choice. Please enter a valid number (1, 2, or 3)." -
ForegroundColor Red
    }
}
}
#Function to perform regular system audits (Sdwheeler, n.d.)
function Systemaudit
{
    try
    { # Check changes in file sizes and permissions
        Get-ChildItem -Path C:\ -Recurse | Get-Acl
        # Check creation of new user accounts
        Get-LocalUser
        Write-Host "System audit completed." -BackgroundColor Cyan
    }
}

```

```

    catch
    {
        Write-Host "Failed to perform system audit: $_" -BackgroundColor Red
    }
}

# Function to monitor system logs(Sdwheeler, n.d.-b)
function MonitorSystemlogs
{
    try
    {
        #Monitor system logs to check security threats
        Get-EventLog -LogName Security
        Write-Host "System logs monitored." -BackgroundColor Cyan
    }
    catch
    {
        Write-Host "Failed to monitor system logs: $_" -BackgroundColor Red
    }
}

# Function to update system(JasonGerend, n.d.-b)
function Updatesystem
{
    # Import the PSWindowsUpdate module
    try
    {
        #Import module check for updates and install updates
        Import-Module PSWindowsUpdate -ErrorAction Stop
        $updates = Get-WindowsUpdate -AcceptAll
        if ($updates)
        {
            Install-WindowsUpdate -AcceptAll -AutoReboot
            Write-Host "System updated successfully." -BackgroundColor Cyan
        }
        else
        {
            Write-Host "No updates available." -BackgroundColor Blue
        }
    }
    catch
    {
        Write-Host "Failed to update system: $_" -BackgroundColor Red
    }
}

#Function to backup data(HitSubscribe, 2024)
function Backupdata
{
    try
    {
        #Define the source and destination

```

```

$Source = Read-Host "Enter the source path for backup"
$Destination = Read-Host "Enter the destination path for backup"
#Validate source and destination
if (-not (Test-Path -Path $Source -PathType Container))
{
    #Error
    throw "Source path does not exist "
}
if (-not (Test-Path -Path $Destination -PathType Container))
{
    #Error
    throw "Destination path does not exist "
}
# Perform the backup
Copy-Item -Path $Source -Destination $Destination -Recurse -Force
Write-Host "Data backed up successfully." -BackgroundColor Cyan
}
catch
{
    Write-Host "Failed to backup data: $_" -BackgroundColor Red
}
}
# Function to optimize resource allocation(Sdwheeler, n.d.-c)
function Optimizeresource
{
    try
    {
        #get amount of free memory
        $freememory = (Get-WmiObject Win32_OperatingSystem).FreePhysicalMemory / 1MB
        #if memory is less then 1024 mb display text
        if ($freememory -lt 1024)
        {
            Write-Host "Low memory. please try and close unnecessary apps" -BackgroundColor
Cyan
        }#otherwise show that memory usage is acceptable
        else
        {
            Write-Host "Memory usage is acceptable " -BackgroundColor Cyan
        }
    }
    catch
    {
        Write-Host "Failed to optimize system: $_" -BackgroundColor Red
    }
}
# Function to prevent system downtime(JasonGerend, n.d.-c)
function Preventsystemdowntime
{
    try

```

```

{ #reboot every week task
  $task = "WeeklyReboot"
  # Check if the task already exists
  $existing = Get-ScheduledTask -TaskName $task -ErrorAction SilentlyContinue
  if ($null -eq $existing) # check to see if and if task doesn't exist, add
it
  {
    #schedule shutdown with restart argument
    $action = New-ScheduledTaskAction -Execute "shutdown.exe" -Argument "/r /t 0"
    #Trigger to make it happen on every sunday at 3 am
    $trigger = New-ScheduledTaskTrigger -Weekly -DaysOfWeek Sunday -At 3am
    #register scheduled task
    Register-ScheduledTask -Action $action -Trigger $trigger -TaskName $task -
Description "Reboots the system every Sunday at 3AM"
    #Send email to HR
    Sendemailreboot -Subject $subject -Body $body
    Write-Host "System downtime prevention scheduled." -BackgroundColor
Blue
  }
  else
  {
    Write-Host "The scheduled task for System downtime prevention '$task' already
exists. No action taken." -BackgroundColor Cyan
  }
}
catch
{
  Write-Host "Failed to prevent system downtime: $_" -BackgroundColor
Red
}
}

#function to send email(Yung, 2024)
function Sendemailreboot
{
  # Define credentials
  $EmailFrom = "harshulshukla99@gmail.com"
  $EmailTo = "Hshukla6048@conestogac.on.ca"
  $Subject = "System Reboot Scheduled"
  $Body = "Hello, A scheduled restart will take place on every sunday at 3 Am . Thank you."
  #Gmail app-specific password
  $Appsificpassword = "pier ptoq xdue ojgz" | ConvertTo-SecureString -AsPlainText -
Force
  #Create the credential object
  $Credential = New-Object System.Management.Automation.PSCredential($EmailFrom,
$Appsificpassword)

```

```

# Create the SMTP client
$SMTPServer = "smtp.gmail.com"
$SMTPClient = New-Object Net.Mail.SmtpClient($SMTPServer, 587)
$SMTPClient.EnableSsl = $true
$SMTPClient.Credentials = $Credential.GetNetworkCredential()

# Send the email
try
{
    $SMTPClient.Send($EmailFrom, $EmailTo, $Subject, $Body)
    Write-Host "Email sent successfully" -BackgroundColor
    Cyan
}
catch
{
    Write-Host "Failed to send email: $_" -BackgroundColor
    Red
}
}

# Function to manage software updates and Logging(Sdwheeler, n.d.-c)
#Specify the log file path
$log = "C:\Logs\SoftwareUpdates.log"
function ManageSoftwareUpdates #define function
{# Set default log file path
    param (
        [string]$Log = "C:\Logs\SoftwareUpdates.log"
    )
    try #Create Logs log file if it doesn't
    exist
    {
        $Directory = "C:\Logs"
        if (-not (Test-Path -Path $Directory -PathType Container))
        {# Create Logs directory if it doesn't exist
            New-Item -Path $Directory -ItemType Directory -Force
        }
        #Check for updates and patches
        $updates = Get-WindowsUpdate
        #List of installed patches
        $patches = Get-HotFix
        #Array to collect log messages
        $Messages = @()
        # loop for each available object
        foreach ($update in $updates)
        { #install updates
            Install-WindowsUpdate -KBArticleID $update.KBArticleID -AcceptAll -AutoReboot
            $Messages += "Update $($update.KBArticleID) installed successfully."
        }
    }
}

```



```

    }
    foreach ($patch in $patches)
    {
        #install the patch
        Install-WindowsUpdate -KBArticleID $patch.HotFixID -AcceptAll -AutoReboot
        $Messages += "Patch $($patch.HotFixID) installed successfully."
    }
    #Display log messages in a table format
    $Messages | Format-Table
}

Table
Write-Host "Software updates and patches managed successfully" -ForegroundColor
Green
}
catch
{
    #error text
    $errortext = "Failed to manage software updates and patches: $_"
    #show error to user
    Write-Host $errortext -ForegroundColor Red
    #error message to log ffile
    $errortext | Out-File -FilePath $Log -Append
}
}

# Function to plan for infrastructure scalability(Sdwheeler, n.d.-b)
function Infrastructurescalability
{
    # Get average CPU usage over a interval
    try
    {
        $cpu = (Get-Counter -Counter "\Processor(_Total)\% Processor Time" -SampleInterval 2 -
MaxSamples 10).CounterSamples.CookedValue | Measure-Object -Average | Select-Object -
ExpandProperty Average
        # Check to see CPU usage exceeds 80
        if ($cpu -gt 80)
        {
            #high usage
            Write-Host "CPU usage is high.Add more resources" -BackgroundColor Red
        }
        else
        {
            Write-Host "CPU usage is normal." -BackgroundColor Cyan
        }
    }
    catch
    {
        Write-Host "Failed to plan for infrastructure scalability: $_" -BackgroundColor
Red
    }
}

#Function to ensure compliance with regulations(0365devx, 2023)

```

```

function Compliancewithregulations
{
    try
    {
        #check for encryption function
        function Dataencryption
        {
            ##if it's true return
            $isEncrypted = $true
            return $isEncrypted
        }
        $isDataEncrypted = Dataencryption
        #if not encrypted
        if (-not $isDataEncrypted)
        {
            Write-Host "Data is not encrypted. This is not compliant" -BackgroundColor Red
        }
        else
        {
            Write-Host "Data is encrypted. This is compliant with regulations" -
BackgroundColor Cyan
        }
    }
    catch
    {
        Write-Host "Failed to ensure compliance with regulations: $_"
    }
}
#Function to tune system performance(Dotnet-Bot, n.d.)
function Tunesystemperformance
{
    try
    {
        #Get the list of processes
        $processes = Get-Process
        #Loop through each process
        foreach ($process in $processes)
        {
            #If the process is using more than 1GB of memory, lower its priority
            if ($process.PagedMemorySize -gt 1GB)
            {
                $process.PriorityClass = "BelowNormal"
            }
        }
        Write-Host "System performance tuned" -BackgroundColor Cyan
    }
    catch
    {
        Write-Host "Failed to tune system performance: $_" -BackgroundColor Red
    }
}

```

```

    }
}
# Function to provide technical support and troubleshooting(Sdwheeler, n.d.-e)
function TechnicalsupportAndtroubleshooting
{
    #Ask the user to enter the service name
    try
    {
        $service = Read-Host -Prompt "Enter the service name for support and
troubleshooting"
        $service = Get-Service -Name $service -ErrorAction SilentlyContinue
        #Check if the service exists.
        if ($null -eq $service)
        {
            Write-Host "Service $service does not exist." -BackgroundColor Red
        }
        else #Display service related details
        {
            Write-Host "Service Details:"
            Write-Host "Name: $($service.Name)"
            Write-Host "Display Name: $($service.DisplayName)"
            Write-Host "Status: $($service.Status)"
            Write-Host "Start Type: $($service.StartType)"
            Write-Host "Description: $($service.Description)"
            #Check service dependencies
            $dependencies = Get-Service $service | Select-Object -ExpandProperty
DependentServices
            if ($dependencies)
            {
                Write-Host "Dependencies: $($dependencies.Name -join ', ')"
            }
            # event logs related to the given service
            $events = Get-WinEvent -LogName System -FilterXPath
"*[System[Provider[@Name='Service Control Manager'] and (EventID=7000 or EventID=7009 or
EventID=7011 or EventID=7022 or EventID=7023 or EventID=7024 or EventID=7031) and
EventData[@ServiceName='$serviceName']]]" -ErrorAction SilentlyContinue
            #display most recent events
            if
($events)
            {
                #related event to service that was entered
                Write-Host "Recent Events Related to $service" -BackgroundColor
Cyan

                #show event in this format
                $events | Select-Object -First 5 | Format-Table TimeCreated, LevelDisplayName,
Message -AutoSize
            }
        }
    }
}

```

```

        else
        {
            #No issues found
            Write-Host "No recent events related to $service found." -BackgroundColor
Red
        }
    }
}
catch
{
    Write-Host "Failed to provide technical support and troubleshooting: $_" -
BackgroundColor Red
}
}
# Main script Logic
try
{
    while ($true) # Display menu options to the user
    {
        Write-Host "Choose an option:" -BackgroundColor Cyan
        Write-Host "1. Security tasks"
        Write-Host "2. User management tasks"
        Write-Host "3. Data protection tasks"
        Write-Host "4. Resource management tasks"
        Write-Host "5. System downtime tasks"
        Write-Host "6. Software update tasks"
        Write-Host "7. Infrastructure scalability tasks"
        Write-Host "8. Compliance tasks"
        Write-Host "9. Performance tuning tasks"
        Write-Host "10. Technical support tasks"
        Write-Host "0. Exit" -BackgroundColor Red
        #Ask for choice
        $choice = Read-Host "Enter your choice"
        switch ($choice)
        {
            "1"
            {
                $securityAction = Read-Host "What security action would you like to take? (1)
Malware scan/(2) Firewall management/(3) System audit/(4) Log monitoring/(5) System update"
                switch ($securityAction) #switch statement for our 5 options
                {
                    "1" { Scanmalware } #call function to scan malware
                    "2" { Firewallrules } #call function for firewall
                    "3" { Systemaudit } #call function for system audit
                    "4" { Monitorsystemlogs } #call function to monitor logs
                    "5" { Updatesystem } #call function to update
                }
            }
        }
    }
}

```

```

        default { Write-Host "Invalid option." }    #for invalid option
    }
}
"2"
{
    $userAction = Read-Host "What user management action would you like to take?
(1) Add user/(2) Update user/(3) Enable user/(4) Disable user/(5) Delete user"
    switch ($userAction)        #switch statement for our user related actions
    {
        "1" { Adduser }        #call function to add user
        "2" { Updateuser }     #call function to update user
        "3" { Enableuser }     #call function to enable user
        "4" { Disableuser }    #call function to disable user
        "5" { Deleteuser }     #call function to delete user
        default { Write-Host "Invalid option." }    #For invalid option
    }
}
"3"
{ #for backup data option
    $dataAction = Read-Host "What data protection action would you like to take?
(1) Backup data"
    switch ($dataAction) #have used switch statement in case we want to add option
    {
        "1" { Backupdata }    #call function
        default { Write-Host "Invalid option." }    #For invalid option
    }
}
"4"
{ #resource allocation
    $resourceAction = Read-Host "What resource management action would you like to
take? (1) Optimize resource allocation"
    switch ($resourceAction)
    {
        "1" { Optimizeresource }    #call function
        default { Write-Host "Invalid option." }    #For invalid option
    }
}
"5"
{ #sunday shutdown funtion
    $downtimeAction = Read-Host "What system downtime action would you like to
take? (1) Prevent system downtime"
    switch ($downtimeAction)
    {
        "1" { Preventsystemdowntime }    #call function
        default { Write-Host "Invalid option." }    #For invalid option
    }
}

```

```

    }
}
"6"
{ #manage updates
    $updateAction = Read-Host "What software update action would you like to take?
(1) Manage software updates and patches"
    switch ($updateAction)
    {
        "1" { ManageSoftwareUpdates -LogFilePath $logFilePath } #call function
        default { Write-Host "Invalid option." } #For invalid option
    }
}
"7"
{ #infrastructure scale
    $infrastructureAction = Read-Host "What infrastructure scalability action
would you like to take? (1) Plan for infrastructure scalability"
    switch ($infrastructureAction)
    {
        "1" { Infrastructurescalability } #call function
        default { Write-Host "Invalid option." } #For invalid option
    }
}
"8"
{ #check compliance
    $complianceAction = Read-Host "What compliance action would you like to take?
(1) Ensure compliance with regulations"
    switch ($complianceAction)
    {
        "1" { Compliancewithregulations } #call function
        default { Write-Host "Invalid option." } #For invalid option
    }
}
"9"
{ #tune performance
    $performanceAction = Read-Host "What performance tuning action would you like
to take? (1) Tune system performance"
    switch ($performanceAction)
    {
        "1" { Tunesystemperformance } #call function
        default { Write-Host "Invalid option." } #For invalid option
    }
}
"10"
{ #for troubleshooting

```



```

        $supportAction = Read-Host "What technical support action would you like to
take? (1) Provide technical support and troubleshooting"
        switch ($supportAction)
        {
            "1" { Technicalsupportandtroubleshooting }      #call function
            default { Write-Host "Invalid option." }        #For invalid option
        }
    }
    "0"
    { return } # Exit the script
    default { Write-Host "Invalid option." } #For invalid option
}
}
}
catch
{
    Write-Host "An unexpected error occurred: $_" -BackgroundColor Red
}

<#
                Script References
Dotnet-Bot. (n.d.). Process.PagedMemorySize Property (System.Diagnostics).
Microsoft Learn. https://learn.microsoft.com/en-
us/dotnet/api/system.diagnostics.process.pagedmemorysize?view=net-8.0

HitSubscribe. (2024, February 29). PowerShell commands every developer should know: 50+
cmDLets for getting things done,
monitoring performance, debugging. Stackify. https://stackify.com/powershell-commands-every-
developer-should-know/

JasonGerend. (n.d.-a). ActiveDirectory Module. Microsoft Learn.
https://learn.microsoft.com/en-us/powershell/module/activedirectory/?view=windowsserver2022-ps

JasonGerend. (n.d.-b). Get-WindowsUpdateLog (WindowsUpdate). Microsoft Learn.
https://learn.microsoft.com/en-us/powershell/module/windowsupdate/get-
windowsupdatelog?view=windowsserver2022-ps

JasonGerend. (n.d.-c). New-ScheduledTask (ScheduledTasks). Microsoft Learn.
https://learn.microsoft.com/en-us/powershell/module/scheduledtasks/new-
scheduledtask?view=windowsserver2022-ps

JasonGerend. (n.d.-d). Start-MPScan (Defender). Microsoft Learn.
https://learn.microsoft.com/en-us/powershell/module/defender/start-
mpscan?view=windowsserver2022-ps

```

0365devx. (2023, March 29). IsEncrypted. Microsoft Learn.

<https://learn.microsoft.com/en-us/exchange/client-developer/web-service-reference/isencrypted>

Paolomatarazzo. (2023, November 21). Manage Windows Firewall with the command line - Windows Security. Microsoft Learn.

<https://learn.microsoft.com/en-us/windows/security/operating-system-security/network-security/windows-firewall/configure-with-command-line?tabs=powershell>

Sdwheeler. (n.d.-a). Get-ACL (Microsoft.PowerShell.Security) - PowerShell. Microsoft Learn.

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-acl?view=powershell-7.4>

Sdwheeler. (n.d.-b). Get-Counter (Microsoft.PowerShell.Diagnostics) - PowerShell. Microsoft Learn.

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.diagnostics/get-counter?view=powershell-7.4>

Sdwheeler. (n.d.-c). Get-EventLog (Microsoft.PowerShell.Management) - PowerShell. Microsoft Learn.

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-eventlog?view=powershell-5.1>

Sdwheeler. (n.d.-d). Get-HotFix (Microsoft.PowerShell.Management) - PowerShell. Microsoft Learn.

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-hotfix?view=powershell-7.4>

Sdwheeler. (n.d.-e). Get-WinEvent (Microsoft.PowerShell.Diagnostics) - PowerShell. Microsoft Learn.

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.diagnostics/get-winevent?view=powershell-7.4>

Sdwheeler. (n.d.-f). Get-WMIObject (Microsoft.PowerShell.Management) - PowerShell. Microsoft Learn.

<https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-wmiobject?view=powershell-5.1>

Yung, Z. (2024, April 1). Send Emails from Powershell:

Tutorial with Examples | Mailtrap Blog. Mailtrap. <https://mailtrap.io/blog/powershell-send-email/>

#>

REFERENCES

Assignment 4: Research Assignment. (n.d.).

https://conestoga.desire2learn.com/d2l/lms/dropbox/user/folder_submit_files.d2l?db=928901&ou=1000701

Company, B. (2023, May 25). *6 Administrative tasks you should automate.* BPM

Company. <https://www.bpmcompany.eu/en/top-6-of-time-consuming-and-tedious-office-tasks-that-should-be-more-automated/>

Creating New User Accounts in Active Directory with ADUC and PowerShell |

Windows OS Hub. (2024, March 15). Windows OS Hub.

<https://woshub.com/new-aduser-create-active-directory-users-powershell/>

how do I create enterprise AD users? - WUYING Workspace - Alibaba Cloud

Documentation Center. (n.d.). <https://www.alibabacloud.com/help/en/wuying-workspace/wuying-workspace-pro-edition/create-modify-and-delete-ad-users>

Premium Vector | System administrator. (2021, October 5). Freepik.

https://www.freepik.com/premium-vector/system-administrator_19150103.htm

Softchris. (n.d.). *Introduction to PowerShell - training.* Microsoft Learn.

<https://learn.microsoft.com/en-us/training/modules/introduction-to-powershell/>

Softchris. (n.d.). *Introduction to PowerShell - training.* Microsoft Learn.

<https://learn.microsoft.com/en-us/training/modules/introduction-to-powershell/>

Yung, Z. (2024, April 1). *Send Emails from Powershell: Tutorial with Examples | Mailtrap Blog.*

Mailtrap. <https://mailtrap.io/blog/powershell-send-email/>

APPENDIX

A. Definitions

Active Directory (AD): Windows domain networks can use Active Directory (AD), which was developed by Microsoft as a directory service. Authentication and authorization services are provided, together with a framework for managing and allocating resources in a networked context.

PowerShell: command-line shell and scripting language. With syntax similar to traditional programming languages, users may issue commands, run scripts, and automate operations with this purpose-driven platform for system administration and automation.

Malware: Malware is purposely designed software that aims to harm computer systems or data, interfere with their functioning, or gain unauthorized access. Among the examples are viruses, worms, ransomware, and spyware.

Firewall: Firewalls are network security systems that monitor and control all incoming and outgoing network traffic using pre-established security rules. Its responsibility is to protect reliable internal networks from shady external networks.

System Audit: A system audit is a thorough analysis of the IT infrastructure of a company to determine operational effectiveness, security, and compliance. To find vulnerabilities and guarantee policy compliance.

Data Backup: Making backup copies of your data and having them on hand in case the original is lost or corrupted is known as data backup, backups are useful in case of incident or data corruption.

System Downtime: When a computer system or network is not operating at its best, it is called system downtime. It may be because of maintenance tasks, security events, software bugs, or hardware malfunctions.

Compliance: To follow the rules, laws, norms, and standards that are pertinent to the industry in which a business operates. It guarantees that procedures and systems follow moral and legal standards.

Performance tuning: To increase the speed, effectiveness, and responsiveness of applications and services.

Technical Support: When a user is having issues with hardware, software, or IT systems, technical support provides help and troubleshooting services. Its goal is to fix problems and get everything back to normal.

B. Full Forms

ADUC: Active Directory Users and Computers

CPU: Central Processing Unit

GUI: Graphical User Interface

HTTP: Hypertext Transfer Protocol

HTTPS: Hypertext Transfer Protocol Secure

PS: PowerShell

RAM: Random Access Memory

SMTP: Simple Mail Transfer Protocol

SSL: Secure Sockets Layer

WMI: Windows Management Instrumentation

C. Software used.

Microsoft Visio, Microsoft Word, VS code, Microsoft Excel