Challenge Name: Amoozon Activation Codes

Description:

IR Case Files: Amoozon Activation Codes

Your team is working incident response for Amoozon™, the world's largest "supercloud" retailer and data hoarding platform.
A suspicious binary has been discovered during a routine endpoint inventory sweep, one that sits dormant unless it finds very specific codes. The binary's author remains unknown, and all attempts to activate its payload have failed.

For this challenge, you are provided a binary, amoozon.exe

Step 1 - Initial Triage

Drop the binary, amoozon.exe, and run it: nothing happens (or a bland error). No files created, no obvious output.

Inspect the folder, maybe: no important config, no magic flag in a .txt file. Classic "environment-dependent" or "persistence/implant" check.

Step 2: Static Strings & Artifact Discovery

Run strings amoozon.exe.
Several juicy discoveries:

```
Alpha
Software\CTFChallenge\PartA
Bravo
Software\CTFChallenge\PartB
Amoozon Security Agent: Registry launch codes not detected.
Activation sequence requires properly staged registry triggers.
Amoozon Activation Failed: Registry codes incorrect.
Payload will remain locked until correct combination is provided.
c|w{
9JLX
~=d]
lpHP
Mingw runtime failure:
  VirtualQuery failed for %d bytes at address %p
  Unknown pseudo relocation protocol version %d.
  Unknown pseudo relocation bit size %d.
glob-1.0-mingw32
GCC: (MinGW.org Cross-GCC Build-20200531-1) 9.2.0
```

Human-readable error pop-ups: "Registry launch codes not detected", "Activation failed", etc.

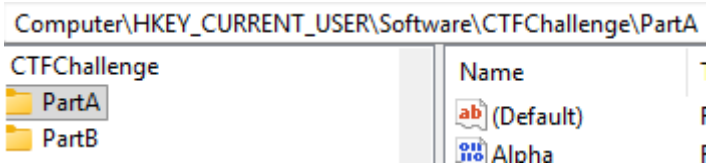
Most Importantly:
Registry paths: Software\CTFChallenge\PartA, Software\CTFChallenge\PartB
Value names: Alpha, Bravo

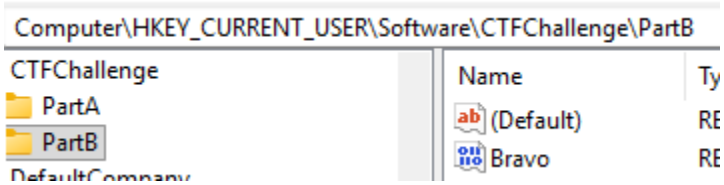
This hints strongly: the binary leverages the Windows Registry for its unlock mechanism.

Step 3 - Registry (Regedit) Engineering

Pop open regedit:

- Go to HKEY_CURRENT_USER\Software\
- Create \CTFChallenge
- Create keys: PartA and PartB
- Under PartA, make a DWORD value, exact name: Alpha
- Under PartB, make another DWORD value: Bravo





But, you don't know the required values yet!

Hence,

Step 4 - Disassembly

Loading in IDA:

- Drop the binary (stripped or not) into IDA or your disassembler of choice.
- Upon examining imports, note heavy use of RegOpenKeyExA and RegQueryValueExA, classic WinAPI calls for registry operations.

Locating the Main Check Loop:

- Identify code at (loc_40188B):

```
mov     [ebp+var_18], 0
mov     [ebp+var_1C], 0
lea     eax, [ebp+var_18]
mov     [esp+48h+var_40], eax                   ; Stores result
mov     [esp+48h+lpValueName], offset ValueName ; "Alpha"
mov     [esp+48h+dwMilliseconds], offset aSoftwareCtfcha ; "Software\\CTFChallenge\\PartA"
call    sub_401410                              ; Registry read
...
lea     eax, [ebp+var_1C]
mov     [esp+48h+var_40], eax
mov     [esp+48h+lpValueName], offset aBravo   ; "Bravo"
mov     [esp+48h+dwMilliseconds], offset aSoftwareCtfcha_0 ;
"Software\\CTFChallenge\\PartB"
call    sub_401410
```

-
- These lines fetch the DWORD registry values “Alpha” and “Bravo”,store them to [ebp+var_18] and [ebp+var_1C].

Deep Dive: sub_401410 (Registry Loader)

Disassembling sub_401410 confirms:

- Calls RegOpenKeyExA with HKEY_CURRENT_USER and appropriate subkey
- Calls RegQueryValueExA for the explicit value
- Requires it to be of type REG_DWORD (type field forced to 4) and stores its value
- Returns error up the stack on missing key, wrong value type, typo, or permissions issue

Takeaway:

- Only exact keys (Alpha, Bravo, REG_DWORD) will proceed to the main validation logic.

Critical Block: Main Registry Value Comparison (loc_40191C et seq)

Moving forward in the listing:

```
loc_40191C:
mov     eax, [ebp+var_18]    ; Alpha
cmp     eax, 539h            ; 0x539 == 1337
jnz     short loc_401930     ; Fail if not 1337
mov     eax, [ebp+var_1C]    ; Bravo
cmp     eax, 1092h           ; 0x1092 == 4242
jz      short loc_401943     ;
```

- Verification: The program checks that Alpha == 1337 and Bravo == 4242, Jackpot!
- Fail route: If not, call failure routine at loc_401930, which prints activation error.

Success Route: The Flag Unlock Chain (loc_401943 and beyond)

At success:

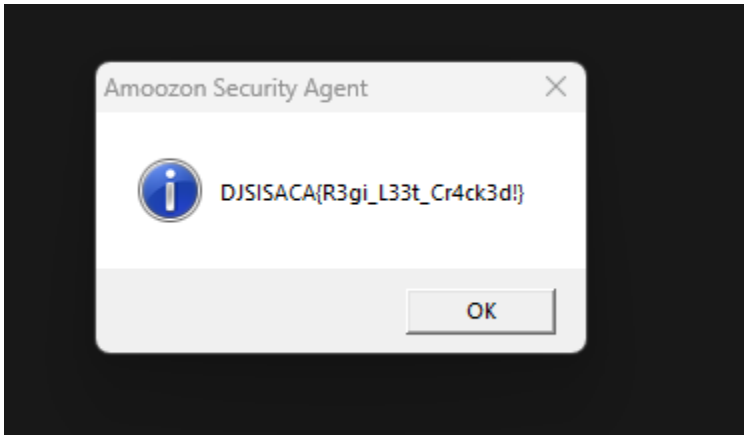
```
loc_401943:
mov     edx, [ebp+var_1C]
mov     eax, [ebp+var_18]
[...]
call    sub_40167D
call    sub_4016AC
call    sub_40171B
```

So now,

In regedit:

- Go to HKCU\Software\CTFChallenge\PartA\, create a DWORD Alpha with value 1337.
- Go to HKCU\Software\CTFChallenge\PartB\, create a DWORD Bravo with value 4242.

If all done well, this should pop up,



Flag: DJSISACA{R3gi_L33t_Cr4ck3d!}