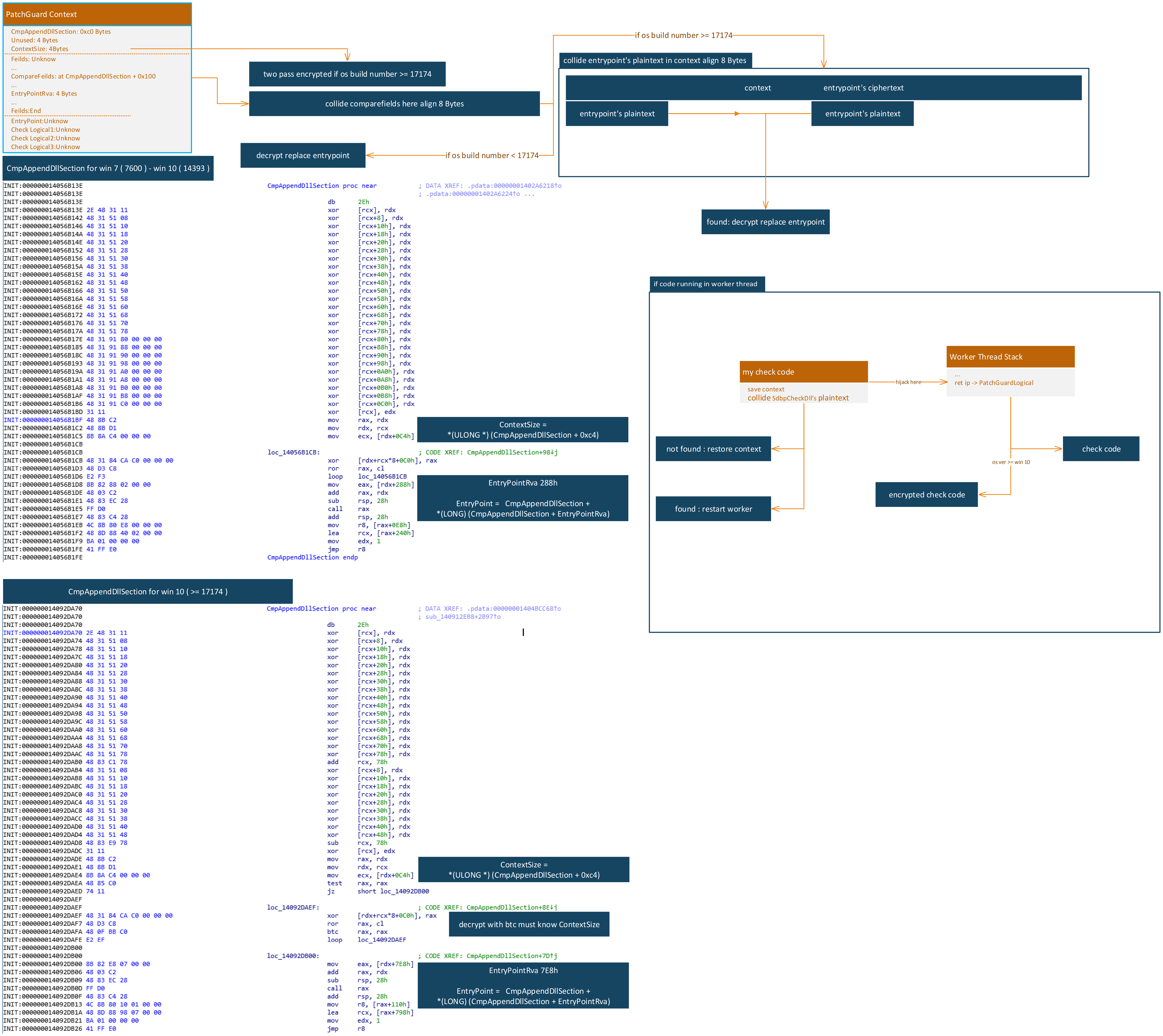
PatchGuard Context
CmpAppendDllSection: 0xc0 Bytes
Unused: 4 Bytes
ContextSize: 4Bytes
Feilds: Unknow
...
CompareFeilds: at CmpAppendDllSection + 0x100
...
EntryPointRva: 4 Bytes
...
Feilds:End
EntryPoint:Unknow
Check Logical1:Unknow
Check Logical2:Unknow
Check Logical3:Unknow
two pass encrypted if os build number >= 17174
collide comparefields here align 8 Bytes
if os build number >= 17174
collide entrypoint's plaintext in context align 8 Bytes
context
entrypoint's ciphertext
entrypoint's plaintext
entrypoint's plaintext
if os build number < 17174
decrypt replace entrypoint
found: decrypt replace entrypoint
if code running in worker thread
my check code
save context
collide SdbpCheckDll's plaintext
hijack here
Worker Thread Stack
...
ret ip -> PatchGuardLogical
not found : restore context
found : restart worker
check code
os ver >= win 10
encrypted check code

CmpAppendDllSection for win 7 (7600) - win 10 (14393)

```
INIT:000000014056B13E                                 CmpAppendDllSection proc near           ; DATA XREF: .pdata:00000001402A6218↑o
INIT:000000014056B13E                                                                         ; .pdata:00000001402A6224↑o ...
INIT:000000014056B13E                                                 db      2Eh
INIT:000000014056B13E 2E 48 31 11                                     xor     [rcx], rdx
INIT:000000014056B142 48 31 51 08                                     xor     [rcx+8], rdx
INIT:000000014056B146 48 31 51 10                                     xor     [rcx+10h], rdx
INIT:000000014056B14A 48 31 51 18                                     xor     [rcx+18h], rdx
INIT:000000014056B14E 48 31 51 20                                     xor     [rcx+20h], rdx
INIT:000000014056B152 48 31 51 28                                     xor     [rcx+28h], rdx
INIT:000000014056B156 48 31 51 30                                     xor     [rcx+30h], rdx
INIT:000000014056B15A 48 31 51 38                                     xor     [rcx+38h], rdx
INIT:000000014056B15E 48 31 51 40                                     xor     [rcx+40h], rdx
INIT:000000014056B162 48 31 51 48                                     xor     [rcx+48h], rdx
INIT:000000014056B166 48 31 51 50                                     xor     [rcx+50h], rdx
INIT:000000014056B16A 48 31 51 58                                     xor     [rcx+58h], rdx
INIT:000000014056B16E 48 31 51 60                                     xor     [rcx+60h], rdx
INIT:000000014056B172 48 31 51 68                                     xor     [rcx+68h], rdx
INIT:000000014056B176 48 31 51 70                                     xor     [rcx+70h], rdx
INIT:000000014056B17A 48 31 51 78                                     xor     [rcx+78h], rdx
INIT:000000014056B17E 48 31 91 80 00 00 00                            xor     [rcx+80h], rdx
INIT:000000014056B185 48 31 91 88 00 00 00                            xor     [rcx+88h], rdx
INIT:000000014056B18C 48 31 91 90 00 00 00                            xor     [rcx+90h], rdx
INIT:000000014056B193 48 31 91 98 00 00 00                            xor     [rcx+98h], rdx
INIT:000000014056B19A 48 31 91 A0 00 00 00                            xor     [rcx+0A0h], rdx
INIT:000000014056B1A1 48 31 91 A8 00 00 00                            xor     [rcx+0A8h], rdx
INIT:000000014056B1A8 48 31 91 B0 00 00 00                            xor     [rcx+0B0h], rdx
INIT:000000014056B1AF 48 31 91 B8 00 00 00                            xor     [rcx+0B8h], rdx
INIT:000000014056B1B6 48 31 91 C0 00 00 00                            xor     [rcx+0C0h], rdx
INIT:000000014056B1BD 31 11                                           xor     [rcx], edx
INIT:000000014056B1BF 48 8B C2                                        mov     rax, rdx
INIT:000000014056B1C2 48 8B D1                                        mov     rdx, rcx
INIT:000000014056B1C5 8B 8A C4 00 00 00                               mov     ecx, [rdx+0C4h]
INIT:000000014056B1CB
INIT:000000014056B1CB                                 loc_14056B1CB:                          ; CODE XREF: CmpAppendDllSection+98↓j
INIT:000000014056B1CB 48 31 84 CA C0 00 00 00                         xor     [rdx+rcx*8+0C0h], rax
INIT:000000014056B1D3 48 D3 C8                                        ror     rax, cl
INIT:000000014056B1D6 E2 F3                                           loop    loc_14056B1CB
INIT:000000014056B1D8 8B 82 88 02 00 00                               mov     eax, [rdx+288h]
INIT:000000014056B1DE 48 03 C2                                        add     rax, rdx
INIT:000000014056B1E1 48 83 EC 28                                     sub     rsp, 28h
INIT:000000014056B1E5 FF D0                                           call    rax
INIT:000000014056B1E7 48 83 C4 28                                     add     rsp, 28h
INIT:000000014056B1EB 4C 8B 80 E8 00 00 00                            mov     r8, [rax+0E8h]
INIT:000000014056B1F2 48 8D 88 40 02 00 00                            lea     rcx, [rax+240h]
INIT:000000014056B1F9 BA 01 00 00 00                                  mov     edx, 1
INIT:000000014056B1FE 41 FF E0                                        jmp     r8
INIT:000000014056B1FE                                 CmpAppendDllSection endp
```

ContextSize =
*(ULONG *) (CmpAppendDllSection + 0xc4)

EntryPointRva 288h

EntryPoint = CmpAppendDllSection +
*(LONG) (CmpAppendDllSection + EntryPointRva)

CmpAppendDllSection for win 10 (>= 17174)

```
INIT:000000014092DA70                                 CmpAppendDllSection proc near           ; DATA XREF: .pdata:00000001404BCC68↑o
INIT:000000014092DA70                                                                         ; sub_140912EB8+2B97↑o
INIT:000000014092DA70                                                 db      2Eh
INIT:000000014092DA70 2E 48 31 11                                     xor     [rcx], rdx
INIT:000000014092DA74 48 31 51 08                                     xor     [rcx+8], rdx
INIT:000000014092DA78 48 31 51 10                                     xor     [rcx+10h], rdx
INIT:000000014092DA7C 48 31 51 18                                     xor     [rcx+18h], rdx
INIT:000000014092DA80 48 31 51 20                                     xor     [rcx+20h], rdx
INIT:000000014092DA84 48 31 51 28                                     xor     [rcx+28h], rdx
INIT:000000014092DA88 48 31 51 30                                     xor     [rcx+30h], rdx
INIT:000000014092DA8C 48 31 51 38                                     xor     [rcx+38h], rdx
INIT:000000014092DA90 48 31 51 40                                     xor     [rcx+40h], rdx
INIT:000000014092DA94 48 31 51 48                                     xor     [rcx+48h], rdx
INIT:000000014092DA98 48 31 51 50                                     xor     [rcx+50h], rdx
INIT:000000014092DA9C 48 31 51 58                                     xor     [rcx+58h], rdx
INIT:000000014092DAA0 48 31 51 60                                     xor     [rcx+60h], rdx
INIT:000000014092DAA4 48 31 51 68                                     xor     [rcx+68h], rdx
INIT:000000014092DAA8 48 31 51 70                                     xor     [rcx+70h], rdx
INIT:000000014092DAAC 48 31 51 78                                     xor     [rcx+78h], rdx
INIT:000000014092DAB0 48 83 C1 78                                     add     rcx, 78h
INIT:000000014092DAB4 48 31 51 08                                     xor     [rcx+8], rdx
INIT:000000014092DAB8 48 31 51 10                                     xor     [rcx+10h], rdx
INIT:000000014092DABC 48 31 51 18                                     xor     [rcx+18h], rdx
INIT:000000014092DAC0 48 31 51 20                                     xor     [rcx+20h], rdx
INIT:000000014092DAC4 48 31 51 28                                     xor     [rcx+28h], rdx
INIT:000000014092DAC8 48 31 51 30                                     xor     [rcx+30h], rdx
INIT:000000014092DACC 48 31 51 38                                     xor     [rcx+38h], rdx
INIT:000000014092DAD0 48 31 51 40                                     xor     [rcx+40h], rdx
INIT:000000014092DAD4 48 31 51 48                                     xor     [rcx+48h], rdx
INIT:000000014092DAD8 48 83 E9 78                                     sub     rcx, 78h
INIT:000000014092DADC 31 11                                           xor     [rcx], edx
INIT:000000014092DADE 48 8B C2                                        mov     rax, rdx
INIT:000000014092DAE1 48 8B D1                                        mov     rdx, rcx
INIT:000000014092DAE4 8B 8A C4 00 00 00                               mov     ecx, [rdx+0C4h]
INIT:000000014092DAEA 48 85 C0                                        test    rax, rax
INIT:000000014092DAED 74 11                                           jz      short loc_14092DB00
INIT:000000014092DAEF
INIT:000000014092DAEF                                 loc_14092DAEF:                          ; CODE XREF: CmpAppendDllSection+8E↓j
INIT:000000014092DAEF 48 31 84 CA C0 00 00 00                         xor     [rdx+rcx*8+0C0h], rax
INIT:000000014092DAF7 48 D3 C8                                        ror     rax, cl
INIT:000000014092DAFA 48 0F BB C0                                     btc     rax, rax
INIT:000000014092DAFE E2 EF                                           loop    loc_14092DAEF
INIT:000000014092DB00
INIT:000000014092DB00                                 loc_14092DB00:                          ; CODE XREF: CmpAppendDllSection+7D↑j
INIT:000000014092DB00 8B 82 E8 07 00 00                               mov     eax, [rdx+7E8h]
INIT:000000014092DB06 48 03 C2                                        add     rax, rdx
INIT:000000014092DB09 48 83 EC 28                                     sub     rsp, 28h
INIT:000000014092DB0D FF D0                                           call    rax
INIT:000000014092DB0F 48 83 C4 28                                     add     rsp, 28h
INIT:000000014092DB13 4C 8B 80 10 01 00 00                            mov     r8, [rax+110h]
INIT:000000014092DB1A 48 8D 88 98 07 00 00                            lea     rcx, [rax+798h]
INIT:000000014092DB21 BA 01 00 00 00                                  mov     edx, 1
INIT:000000014092DB26 41 FF E0                                        jmp     r8
```

ContextSize =
*(ULONG *) (CmpAppendDllSection + 0xc4)

decrypt with btc must know ContextSize

EntryPointRva 7E8h

EntryPoint = CmpAppendDllSection +
*(LONG) (CmpAppendDllSection + EntryPointRva)