1. oracle_p1

原理: 利用資料儲存方式的漏洞，去覆蓋原有的資料並取得正確資料

先用 oracle1 generate secret key 1，不猜

再用 oracle2 generate secret key 2

在 gen_secret() 當中，使用 local variable : char buf[64] 存取產生的 secret key

gen_secret() 結束之後，回傳 buf 的位置，

但 buf 的位置會被釋放，當作沒有被使用

所以在第二次 call gen_secret() 時 宣告 char buf[64]時，又會 access 到同一個位置，並更改此位置當中的資料

因為 struct oracle_t 中的 secret 是 **char pointer**

存的是 gen_secret() 回傳的 buf 位置

所以 oracle1 在驗證時，access 到的 secret key1 其實是 secret key2 (被覆寫了)

所以在 oracle1 或 oracle2 答錯得到的 secret key 都會是 secrete key2

作法: 先 call oracle1 再 call orcale2，答錯拿到 secret key 後就可以解

```
typedef struct {
    int stage;
    int key;
    char *secret;
}   oracle_t;
```

2. oracle_p2

原理: 若相同的 seed 作為 srand() 的參數，rand() 所產生的序列會是相同的

oseed 只有在釋出答案或答題正確的時候會被重設為 -1

所以我們在第一次 call oracle1 的時候，如果不先回答到最後，先 call oracle2

Oracle2 產生的 secret key 的 rand() 也是使用和 oracle1 的 oseed 而 srand()

如果使用相同的 seed，就會產生出一模一樣的序列

所以我們可以透過答錯，得到 oracle1 的 oseed，再用程式使用該 oseed 去產生相同的 rand() 序列

在 oracle1 gen_secret() 時用 oseed 產生了 4 次 rand() 作為 secret key1

在 oracle2 init 時會用 oseed 產生 1 次 rand() 作為 gen_secret() 的 k1 之後才會在 gen_secret() 用 oseed 產生 4 次 rand() 作為 secret key2

程式模擬: 用 oracle1 的 oseed 作為 srand() 的 seed 產生 5 次 rand()，再產生 4 次 rand() 最後的 4 次 rand() 就會是產生 secret key2 的序列

3. webcrawler_p3

原理: 利用 gethostbyname() 並非 thread-safe 的特性
Server 是用兩個 thread 去建立連線
而兩個 thread 都是使用 gethostbyname2() 去取得 ip 位置，
但 gethostbyname2() 是使用自身的 global buffer 去儲存 ip，
並回傳 global buffer 的位置，所以不同的 thread 都 call gethostbyname() 時，global buffer 會被後來 的人覆蓋
當 thread1 先 call 了 gethostbyname()，取得了該 global buffer 的位置，
切換至 Thread2 後，thread2 call gehostbyname2() 時，會覆寫 global buffer
那再切換回 thread1 的時候，thread1 access 到的 ip 會是 thread2 查詢的結果

做法: 先連 google.com/10000 再連 localhost/10000 (port 要一樣) 因為 google.com/10000 應該是連不上的，所以會進入第二次迴圈
而此時 global buffer 的 ip 已經被改成 Localhost，第二次迴圈就會去連上 Localhost/1000