Hackers Keyboard

Extract all HID DATA then map to keycode.

```
# USB HID Keycode to Character Mapping with Shift Handling
keycode_map = {
  0x04: 'a', 0x05: 'b', 0x06: 'c', 0x07: 'd', 0x08: 'e', 0x09: 'f', 0x0A: 'g', 0x0B: 'h', 0x0C: 'i', 0x0D: 'j',
  0x0E: 'k', 0x0F: 'l', 0x10: 'm', 0x11: 'n', 0x12: 'o', 0x13: 'p', 0x14: 'q', 0x15: 'r', 0x16: 's', 0x17: 't',
  0x18: 'u', 0x19: 'v', 0x1A: 'w', 0x1B: 'x', 0x1C: 'y', 0x1D: 'z', 0x1E: '1', 0x1F: '2', 0x20: '3', 0x21: '4',
  0x22: '5', 0x23: '6', 0x24: '7', 0x25: '8', 0x26: '9', 0x27: '0', 0x28: 'Enter', 0x2C: ' ', 0x2D: '-', 0x2E: '=',
  0x2F: '[', 0x30: ']', 0x31: '\\', 0x33: ';', 0x34: "'", 0x35: '`', 0x36: ',', 0x37: ',' 0x38: '/'
}
# Shifted Characters Mapping
shifted_keycode_map = {
  0x04: 'A', 0x05: 'B', 0x06: 'C', 0x07: 'D', 0x08: 'E', 0x09: 'F', 0x0A: 'G', 0x0B: 'H', 0x0C: 'I', 0x0D: 'J',
  0x0E: 'K', 0x0F: 'L', 0x10: 'M', 0x11: 'N', 0x12: 'O', 0x13: 'P', 0x14: 'Q', 0x15: 'R', 0x16: 'S', 0x17: 'T',
  0x18: 'U', 0x19: 'V', 0x1A: 'W', 0x1B: 'X', 0x1C: 'Y', 0x1D: 'Z', 0x1E: '!', 0x1F: '@', 0x20: '#', 0x21: '$',
  0x22: '%', 0x23: '^', 0x24: '&', 0x25: '*', 0x26: '(', 0x27: ')', 0x2D: '_', 0x2E: '+', 0x2F: '{', 0x30: '}',
  0x31: '|', 0x33: ':', 0x34: '"', 0x35: '~', 0x36: '<', 0x37: '>', 0x38: '?'
}
# Input HID data
hid_data = """
00000b0000000000
0000000000000000
000008000000000
0000000000000000
00000f000000000
0000000000000000
00000f0000000000
000000000000000
00001200000000000
000000000000000
00002c0000000000
000000000000000
00000b000000000
000000000000000
0000120000000000
0000000000000000
0000130000000000
0000000000000000
0000080000000000
000000000000000
00002c0000000000
0000000000000000
00001c0000000000
0000000000000000
0000120000000000
000000000000000
0000180000000000
000000000000000
00002c0000000000
000000000000000
0000060000000000
0000000000000000
```

00000400000000
0000000000000
000011000000000
0000000000000
00002c000000000
00000000000000
00000a000000000
00000000000000
000008000000000
00000000000000
000017000000000
00000000000000
00002c000000000
00000000000000
000017000000000
00000000000000
00000b000000000
00000000000000
00000c000000000
00000000000000
000016000000000
00000000000000
00002c000000000
00000000000000
00000900000000
00000000000000
00000f00000000
00000000000000
00000400000000
00000000000000
00000a000000000
00000000000000
02000000000000
020033000000000
02000000000000
00000000000000
02000000000000
020027000000000
02000000000000
0000000000000
00002c000000000
0000000000000
000016000000000
0000000000000
000017000000000
0000000000000
000012000000000
0000000000000
000018000000000
0000000000000
000017000000000
0000000000000
000006000000000
0000000000000
000017000000000
0000000000000
00000900000000

COCCIO	
C2002/200000000000000000000000000000000	0000000000000
2220000000000000000000000000000000000	0200000000000
00000500000000000000000000000000000000	02002f000000000
000005000000000 00000000000000 00000000	02000000000000
0000100000000000000000000000000000000	00000000000000
0000100000000000000000000000000000000	00005000000000
00001-0000000000 00000000000000 00000000	
000000000000000	
00005b000000000 00001c000000000 0000c000000000	
0001c000000000000000000000000000000000	00005b000000000
00000000000000000000000000000000000000	00000000000000
00000000000000000000000000000000000000	00001c000000000
00000000000000000000000000000000000000	
00000000000000000000000000000000000000	
00000500000000000000000000000000000000	00000000000000
00000c00000000000000000000000000000000	
00000000000000000000000000000000000000	00000000000000
000017000000000 000013000000000 00001300000000	00000c000000000
00000000000000000000000000000000000000	00000000000000
00000000000000000000000000000000000000	
00000000000000000000000000000000000000	00000000000000
00000000000000000000000000000000000000	
00000000000000000000000000000000000000	0000000000000
00001900000000000000000000000000000000	000020000000000
00000000000000000000000000000000000000	0000000000000
0000160000000000 00001e000000000 0000000000	000019000000000
00000000000000000000000000000000000000	0000000000000
00001e0000000000 00000000000000 00000000	000016000000000
00000000000000000000000000000000000000	0000000000000
00008000000000000000000000000000000000	00001e000000000
0000000000000000000 000000000000000 0000	0000000000000
00000600000000000000000000000000000000	00000800000000
000000000000000000 00001b000000000 000023000000000 0000000000000	0000000000000
00001b000000000 0000000000000 00002300000000 00000000	00000600000000
0000000000000000000 000023000000000 000008000000000 00000800000000	0000000000000
0000230000000000 00000000000000 00000800000000	00001b000000000
0000000000000000000 000000000000000 0000	0000000000000
00000800000000000000000000000000000000	000023000000000
00000000000000000000000000000000000000	0000000000000
00001500000000000000000000000000000000	00000800000000
0000000000000000000 00001c0000000000 020000000000	0000000000000
00001c00000000000000000000000000000000	
000000000000000000 02000000000000 0200300000000	0000000000000
020000000000000000 020030000000000 0200000000	00001c000000000
020030000000000000 02000000000000 00000000	
020000000000000000 00000000000000 000000	02000000000000
000000000000000000 000000000000000 00001200000000	02003000000000
00000a000000000 000000000000000 0000120000000000	
0000000000000000 000012000000000 00001200000000	
000012000000000 00000000000000 00001200000000	
0000000000000000 0000120000000000 00000000	
000012000000000 00000000000000 00000700000000	
0000000000000 000007000000000	
000007000000000	
0000000000000	
	0000000000000

```
00002c0000000000
0000000000000000
00000f0000000000
000000000000000
0000180000000000
0000000000000000
0000060000000000
0000000000000000
00000e00000000000
000000000000000
.....
# Decode the HID data
decoded_output = []
for line in hid_data.splitlines():
 line = line.strip() # Remove extra whitespace
 if len(line) == 16: # Ensure the line has exactly 16 hex characters
     bytes_data = bytes.fromhex(line)
     modifier = bytes_data[0] # Modifier byte (Shift, Ctrl, etc.)
     keycode = bytes_data[2] # Keycode byte
     if keycode != 0:
       if modifier & 0x02: # Check if Shift is pressed
         char = shifted_keycode_map.get(keycode, ")
         char = keycode_map.get(keycode, ")
       decoded_output.append(char)
   except ValueError:
     continue
# Join the decoded characters into a string
decoded_text = ".join(decoded_output)
print("Decoded Text:")
print(decoded_text)
```

The code failed. So I just manually do it. KEKW

https://usb.org/sites/default/files/hut1_5.pdf at chapter 10

STOUTCTF{BY3YFBITP3VS1ECX6ERY}