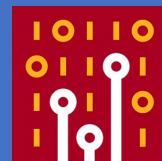




SharkFest '18 US

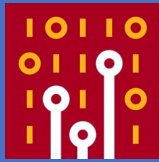


Using more of the features of Wireshark to Write Better Dissectors

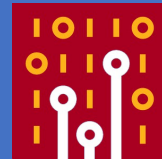
Richard Sharpe
Hammerspace



About me?



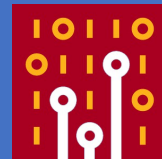
- Contributed to Wireshark since ~1999
 - Was called Ethereal then
- Do a lot of work with Wireless protocols now
- Wrote a bunch of early dissectors like SMB, FTP, etc.



- Dissector outline
- Converting values to meaning strings
- Adding units to displayed values
- Handling bit fields
- Custom display functions
- Dissector tables
- Expert information
- More...



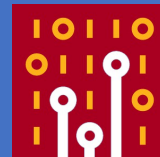
General hints



- Lots of information in doc/README.dissector
 - Can be hard to find your way around
- Use other dissectors to get hints
- 'git grep'



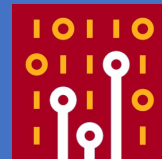
An example



- ▶ VHT Capabilities Info: 0x738fe9f2
- ▶ VHT Supported MCS Set
- ▼ Rx MCS Map: 0xaaaa
 -10 = Rx 1 SS: MCS 0-9 (0x2)
 -10.. = Rx 2 SS: MCS 0-9 (0x2)
 -10.... = Rx 3 SS: MCS 0-9 (0x2)
 -10... = Rx 4 SS: MCS 0-9 (0x2)
 -10 = Rx 5 SS: MCS 0-9 (0x2)
 -10.. = Rx 6 SS: MCS 0-9 (0x2)
 -10.... = Rx 7 SS: MCS 0-9 (0x2)
 -10... = Rx 8 SS: MCS 0-9 (0x2)
- ▶ Tx MCS Map: 0xaaaa
- ▼ Tag: VHT Operation
 - Tag Number: VHT Operation (192)
 - Tag length: 5
 - ▼ VHT Operation Info
 - Channel Width: 80 MHz (0x01)
 - Channel Center Segment 0: 42
 - Channel Center Segment 1: 0
 - ▶ Basic MCS Map: 0xffffc
- ▼ Tag: VHT Tx Power Envelope
 - Tag Number: VHT Tx Power Envelope (195)
 - Tag length: 5
 - ▶ Tx Pwr Info: 0x03
 - Local Max Tx Pwr Constraint 20MHz: -30.0 dBm
 - Local Max Tx Pwr Constraint 40MHz: -30.0 dBm



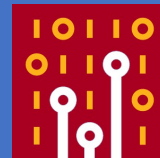
Dissector outline



- Definitions (hf and ett declarations)
- Any value_strings, true_false_strings
- The dissector function
- The header field registrations
- The ett value registrations



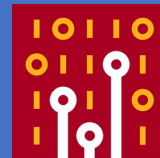
Dissector outline, cont



- You have a tvb (linear bag of bytes)
- You march along the tvb displaying fields
 - Bit fields, byte fields, 16-bit fields, 32-bit fields...
- Increment offset to indicate position in the tvb
- Insert sub-trees where you need them for grouping



Dissector outline, code



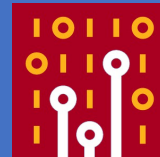
```
static int hf_ieee80211_tag_wnm_sleep_mode_action_type = -1;
. . .

static const value_string wnm_sleep_mode_action_types[] = {
    { 0, "Enter WNM-Sleep Mode" },
    { 1, "Exit WNM-Sleep Mode" },
    { 0, NULL }
};

static int
dissect_wnm_sleep_mode(tvbuff_t *tvb, packet_info *pinfo _U_, proto_tree *tree, void* data _U_)
{
    int offset = 0;
    proto_tree_add_item(tree, hf_ieee80211_tag_wnm_sleep_mode_action_type,
        tvb, offset, 1, ENC_LITTLE_ENDIAN);
    offset++;
    proto_tree_add_item(tree, hf_ieee80211_tag_wnm_sleep_mode_response_status,
        tvb, offset, 1, ENC_LITTLE_ENDIAN);
    offset++;
    proto_tree_add_item(tree, hf_ieee80211_tag_wnm_sleep_mode_interval,
        tvb, offset, 2, ENC_LITTLE_ENDIAN);
    offset += 2;
    return offset;
}
. . .
/* WNM-Sleep Mode */
{&hf_ieee80211_tag_wnm_sleep_mode_action_type,
 {"Action Type", "wlan.wnm_sleep_mode.action_type",
 FT_UINT8, BASE_DEC, VALS(wnm_sleep_mode_action_types), 0,
 "WNM-Sleep Mode Action Type", HFILL }},
```

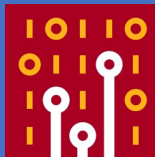



Dissector Outline, code



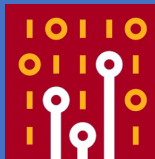
```
static int hf_ieee80211_tag_wnm_sleep_mode_action_type = -1;
. . .

static const value_string wnm_sleep_mode_action_types[] = {
    { 0, "Enter WNM-Sleep Mode" },
    { 1, "Exit WNM-Sleep Mode" },
    { 0, NULL }
};
```



Dissector Outline, code

```
static int
dissect_wnm_sleep_mode(tvbuff_t *tvb, packet_info *pinfo _U_,
proto_tree *tree, void* data _U_)
{
    int offset = 0;
    proto_tree_add_item(tree, hf_tag_wnm_sleep_mode_action_type,
                        tvb, offset, 1, ENC_LITTLE_ENDIAN);
    offset++;
    proto_tree_add_item(tree, hf_tag_wnm_sleep_mode_response_status,
                        tvb, offset, 1, ENC_LITTLE_ENDIAN);
    offset++;
    proto_tree_add_item(tree, hf_tag_wnm_sleep_mode_interval,
                        tvb, offset, 2, ENC_LITTLE_ENDIAN);
    offset += 2;
    return offset;
}
```

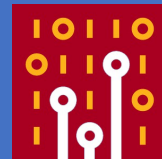


Dissector Outline, code

```
/* WNM-Sleep Mode */
{&hf_tag_wnm_sleep_mode_action_type,
 {"Action Type", "wlan.wnm_sleep_mode.action_type",
 FT_UINT8, BASE_DEC, VALS(wnm_sleep_mode_action_types), 0,
 "WNM-Sleep Mode Action Type", HFILL }},
```



Converting values to strings



- Define your value strings
- Refer to the value_string in a header field

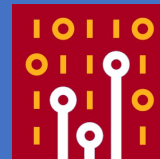
```
static const value_string ff_vht_mimo_cntrl_channel_width_vals[] = {  
    {0x00, "20 MHz"},  
    {0x01, "40 MHz"},  
    {0x02, "80 MHz"},  
    {0x03, "160 MHz / 80+80 MHz"},  
    {0, NULL}  
};
```

. . . some code . . .

```
{&hf_ieee80211_ff_vht_mimo_cntrl_channel_width,  
 {"channel width", "wlan.vht.mimo.control.chanwidth",  
  FT_UINT24, BASE_HEX, VALS(ff_vht_mimo_cntrl_channel_width_vals), 0x0000c0,  
  NULL, HFILL }},
```



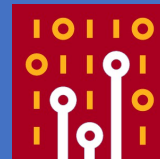
Values to strings, cont



- What if you have some unused values
 - Wireshark will display them as “unknown”
- What if the protocol requires “reserved”?
 - You have a couple of choices
- Define the reserved values
 - Not practical if a large number ($> \sim 10$)
- Use a range string
- Use a custom formatting field (later slide)



Values to strings, cont

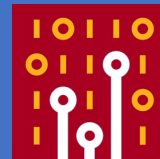


- Range strings
 - A value range and a string

```
static const range_string some_ranges_vals[] = {
    {0, 0, "No Limit"},
    {1, 1, "4 Basic subframes"},
    {2, 2, "8 Basic subframes"},
    {3, 3, "16 Basic subframes"},
    {4, 4, "32 Basic subframes"},
    {5, 5, "64 Basic subframes"},
    {6, 6, "128 Basic subframes"},
    {7, 7, "256 Basic subframes"},
    {8, 255, "reserved"},
    {0, 0, NULL}
};
```



Adding units to values

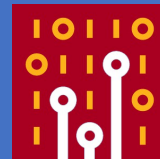


- When you need to add units to a value
 - Use `BASE_UNIT_STRING` and specify the unit
 - There are many predefined in `epan/unit_strings.h`

```
{&hf_hs20_reauth_delay,  
 {"Re-Auth Delay", "wlan.hs20.deauth.reauth_delay",  
  FT_UINT16, BASE_DEC|BASE_UNIT_STRING, &units_seconds,  
  0, NULL, HFILL }},
```



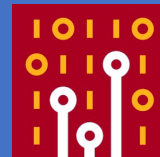
Handling bitfields



- Two ways
 - Write a series of `proto_tree_add_item` statements
 - One for each bitfield
 - Specifying the same offset each time
 - The `hf` fields control the bitmask used
 - Use `proto_tree_add_bitmask`
 - Or `proto_tree_add_bitmask_with_flags`



Handling bitfields, code 1



```
mfb_subtree = proto_item_add_subtree(ti, ett_mfb_subtree);
proto_tree_add_item(mfb_subtree, hf_ieee80211_htc_num_sts, tvb, offset, 4, ENC_LITTLE_ENDIAN);
proto_tree_add_item(mfb_subtree, hf_ieee80211_htc_vht_mcs, tvb, offset, 4, ENC_LITTLE_ENDIAN);
proto_tree_add_item(mfb_subtree, hf_ieee80211_htc_bw, tvb, offset, 4, ENC_LITTLE_ENDIAN);
```

. . .

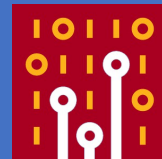
```
{&hf_ieee80211_htc_num_sts,
 {"NUM_STS", "wlan.htc.num_sts",
  FT_UINT32, BASE_DEC, NULL, 0x00000E00,
  "Recommended NUM_STS", HFILL }},

{&hf_ieee80211_htc_vht_mcs,
 {"VHT-MCS", "wlan.htc.vht_mcs",
  FT_UINT32, BASE_DEC, NULL, 0x0000F000,
  "Recommended VHT-MCS", HFILL }},

{&hf_ieee80211_htc_bw,
 {"BW", "wlan.htc.bw",
  FT_UINT32, BASE_DEC, VALS(ieee80211_htc_bw_recommended_vht_mcs_vals), 0x00030000,
  "Bandwidth for recommended VHT-MCS", HFILL }},
```



Handling bitfields, code 2



```
static const int *ieee80211_hta3[] = {
    &hf_ieee80211_hta_basic_stbc_mcs,
    &hf_ieee80211_hta_dual_stbc_protection,
    &hf_ieee80211_hta_secondary_beacon,
    &hf_ieee80211_hta_lsig_txop_protection,
    &hf_ieee80211_hta_pco_active,
    &hf_ieee80211_hta_pco_phase,
    NULL
};

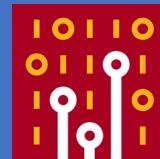
. . .

/* 2 byte HT additional capabilities */
proto_tree_add_bitmask_with_flags(tree, tvb, offset, hf_ieee80211_hta_cap2,
                                  ett_hta_cap2_tree, ieee80211_hta3,
                                  ENC_LITTLE_ENDIAN, BMT_NO_APPEND);

offset += 2;
```



Handling bitfields, code 3



```
{&hf_ieee80211_hta_cap2,  
 {"HT Additional Capabilities", "wlan.hta.capabilities",  
  FT_UINT16, BASE_HEX, NULL, 0,  
  "HT Additional Capability information", HFILL }},
```

. . .

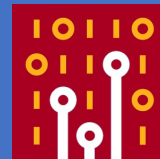
```
{&hf_ieee80211_hta_basic_stbc_mcs,  
 {"Basic STB Modulation and Coding Scheme (MCS)",  
  "wlan.hta.capabilities.basic_stbc_mcs",  
  FT_UINT16, BASE_HEX, NULL, 0x007f,  
  NULL, HFILL }},
```

. . .

```
{&hf_ieee80211_fc_to_ds,  
 {"To DS", "wlan.fc.tods",  
  FT_BOOLEAN, 8, TFS(&tods_flag), FLAG_TO_DS,  
  "To DS flag", HFILL }}, /* 4 */
```



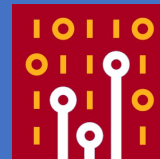
Custom display functions



- When you need precise control over how a field's value is displayed
 - Use a custom display function
 - Can perform many calculations
- Produces a better result than other techniques



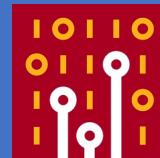
Custom display, code 1



```
/*  
 * Print the UL target RSSI field as per the spec.  
 * 0->30 map to -90 to -30 dBm.  
 * 31 maps to Max transmit power */  
static void  
ul_target_rssi_base_custom(gchar *result, guint32 target_rssi)  
{  
    if (target_rssi <= 30) {  
        g_snprintf(result, ITEM_LABEL_LENGTH, "%ddbM", -90 + (2 *  
target_rssi));  
    } else if (target_rssi == 31) {  
        g_snprintf(result, ITEM_LABEL_LENGTH, "Max transmit power");  
    }  
}  
  
. . .  
  
{&hf_ieee80211_he_ul_target_rssi,  
 {"UL Target RSSI", "wlan.htc.he.a_control.umrs.ul_target_rssi",  
 FT_UINT8, BASE_CUSTOM, CF_FUNC(ul_target_rssi_base_custom), 0x0,  
 NULL, HFILL }},
```



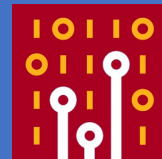
Custom Display, code 2



```
/*
 * Print the target RSSI field as per the spec.
 * 0->90 map to -110 to -20 dBm.
 * 127 maps to Max transmit power for assigned MCS
 * rest are reserved.
 */
static void
target_rssi_base_custom(gchar *result, guint32 target_rssi)
{
    if (target_rssi <= 90) {
        g_snprintf(result, ITEM_LABEL_LENGTH, "%ddbM", -110 + target_rssi);
    } else if (target_rssi == 127) {
        g_snprintf(result, ITEM_LABEL_LENGTH, "Max transmit power");
    } else {
        g_snprintf(result, ITEM_LABEL_LENGTH, "Reserved");
    }
}
```



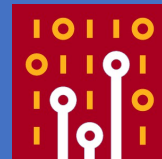
Custom display alternative



- `proto_tree_add_uint_format`
 - Will need to get the value from the `tvb`
 - `tvb_get_guint8` etc
 - Will need a series of if tests or a case statement
- Makes the code more cluttered



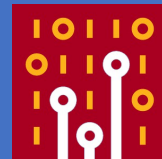
Dissector Tables



- When you need to make dissection functions available to other dissectors
 - You could simply make them non static and call them from the other function
 - Kind of ugly and frowned upon
- Use dissector tables
- A bit of work to do, however



Dissector Tables, code 1

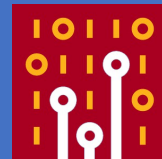


- Register the dissector table

```
wifi_alliance_anqp_info_table =  
    register_dissector_table("wlan.anqp.wifi_alliance.subtype",  
                            "Wi-Fi Alliance ANQP Subtype",  
                            proto_wlan, FT_UINT8, BASE_HEX);
```



Dissector Tables, code 2

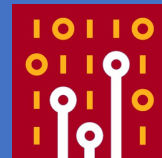


- Add dissectors to it
 - In the proto_reg_handoff function

```
dissector_add_uint("wlan.anqp.wifi_alliance.subtype",
                  WFA_SUBTYPE_HS20_ANQP,
                  create_dissector_handle(dissect_hs20_anqp, -1));
dissector_add_uint("wlan.ie.wifi_alliance.subtype",
                  WFA_SUBTYPE_SUBSCRIPTION_REMEDIATION,
                  create_dissector_handle(dissect_hs20_subscription_remediation,
                  -1));
dissector_add_uint("wlan.ie.wifi_alliance.subtype",
                  WFA_SUBTYPE_DEAUTHENTICATION_IMMINENT,
                  create_dissector_handle(dissect_hs20_deauthentication_imminent,
                  -1));
```



Dissector Tables, code 3

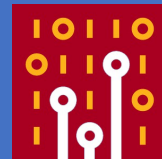


- Call through the dissector table
- You cannot pass offset
 - Need to create a new tvb
 - You can pass a pointer to a blob of data
- If there is no dissector for subtype, returns 0

```
subtvb = tvb_new_subset_remaining(tvb, offset);  
if (!dissector_try_uint_new(wifi_alliance_anqp_info_table, subtype,  
                           subtvb, pinfo, tree, FALSE, data))  
    call_data_dissector(subtvb, pinfo, tree);
```



Dissector Tables, code 4

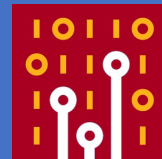


- What the dissector looks like

```
static int
dissect_hs20_anqp(tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree,
                 void *data)
{
    . . .
    return tvb_captured_length(tvb); // How many bytes we consumed
}
```



Dissector Tables, code 5

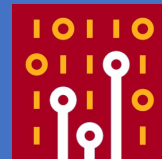


- Finding a dissector table from another dissector
 - In the `proto_reg_handoff` function

```
media_type_dissector_table = find_dissector_table("media_type");
```



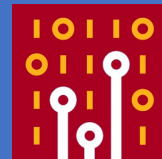
Expert information



- Adding expert information is relatively simple
 - Define ei variables similar to hf variable
 - Register your ei variables
 - Call `expert_add_info` or `expert_add_info_format`



Expert information, code 1

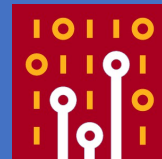


- Defining your ei variables

```
static expert_field ei_ieee80211_bad_length = EI_INIT;  
static expert_field ei_ieee80211_inv_val = EI_INIT;  
static expert_field ei_ieee80211_vht_tpe_pwr_info_count = EI_INIT;  
static expert_field ei_ieee80211_ff_query_response_length = EI_INIT;
```



Expert information, code 2

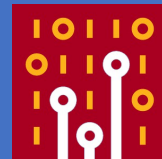


- Registering your ei definitions
- A number of groups (PI_MALFORMED) available
 - PI_DEBUG, PI_MALFORMED, PI_PROTOCOL, etc
- A number of severities (PI_ERROR) available
 - PI_ERROR, PI_WARN, PI_NOTE, etc

```
{ &ei_ieee80211_bad_length,  
  { "ieee80211.bad_length", PI_MALFORMED, PI_ERROR,  
    "wrong length indicated", EXPFILL }},
```




Expert information, code 3

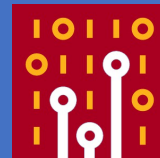


- Two functions you can call
 - `expert_add_info`
 - `expert_add_info_format`

```
expert_add_info(pinfo, tix, &ei_ieee80211_tag_measure_report_unknown);  
.  
.  
.  
expert_add_info_format(pinfo, cw_item, &ei_ieee80211_dmg_subtype,  
                        "DMG STA shouldn't transmit control wrapper frame");
```



A neat approach



```
{
    gint err = -1;
    int offset = some_value; // Usually from above us
    int size = some_size_value;

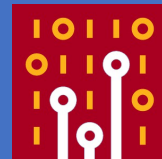
    offset += 3;
    do {
        if (size < 1) break;
        proto_tree_add_item(tlv_root, hf_nan_attr_dialog_token,
                            tvb, offset, 1, ENC_NA);
        size -= 1; offset += 1;

        if (size < 1) break;
        . . .

        if (size == 0)
            Err = 0;
    } while (0);
}
```



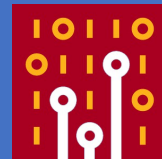
A neat approach, 2



```
if (err) {  
    expert_add_info_format(pinfo, tlv_item,  
                           &ei_wifi_nan_attr_len, "Error parsing attribute");  
}
```



The End



- Questions
- Feedback