Technical work: CoAP protocol in Suricata

1. When suricata starts first it verifies and registers their module in memory or configuration. From the suricata.c file PostConfLoadedSetup() is the main function and this function is meant to contain code that needs to be run once the configuration has been loaded.
2. From this function suricata register their application layer protocols parser, AppLayerSetup() set the detection and parser of application layer protocol AppLayerSetup() use AppLayerParserRegisterProtocolParsers() to register the different application layer parsers
3. We have to create two files in the src directory with name “app-layer-coap.c” and “app-layer-coap.h” where we define the CoAP parser.
4. Create a function RegisterCOAPParsers() in “app-layer-coap.c”
5. Now we call RegisterCOAPParsers() function in the AppLayerParserRegisterProtocolParsers() to register the CoAP parse
6. Two changing in “app-layer-coap.c” are

```
#include "util-debug.h"
#include "decode-events.h"
#include "util-unittest-helper.h"
#include "util-validate.h"

#include "runmodes.h"
#include "app-layer-coap.h"
```

```
RegisterIKEV2Parsers();
RegisterKRB5Parsers();
RegisterDHCPParsers();
RegisterTemplateRustParsers();
RegisterTemplateParsers();
RegisterCOAPParsers();
```

7. With the registration of CoAP parser in “app-layer-coap.c”, we also need to define the CoAP protocol name as a global so other function can use it
8. This thing define in “app-layer-protos.c”, after adding CoAP variable in it file look like

```
        case ALPROTO_TEMPLATE_RUST:
            proto_name = "template-rust";
            break;
        case ALPROTO_FAILED:
            proto_name = "failed";
            break;
        case ALPROTO_COAP:
            proto_name = "coap";
            break;
```

```
if (strcmp(proto_name,"template")==0) return ALPROTO_TEMPLATE;
if (strcmp(proto_name,"template-rust")==0) return ALPROTO_TEMPLATE_RUST;
if (strcmp(proto_name,"failed")==0) return ALPROTO_FAILED;
if (strcmp(proto_name,"coap")==0) return ALPROTO_COAP;
```

9. We also need to define CoAP variable in “app-layer-protos.h” header file struct

```
  ALPROTO_KRB5,
  ALPROTO_DHCP,
  ALPROTO_TEMPLATE,
  ALPROTO_TEMPLATE_RUST,
  ALPROTO_COAP,
```

10. Check that suricata.yml or configuration file have settings of CoAP protocol and if it is available then it is enable or not if it is enable in the suricata.yaml then suricata process it further

```
if (AppLayerProtoDetectConfProtoDetectionEnabled("udp", proto_name)) {
    //associating a string (e.g coap) that would be used to write rules.
    AppLayerProtoDetectRegisterProtocol(ALPROTO_COAP, proto_name);


    //(STREAM_TOCLIENT) and to server (STREAM_TOSERVER).
    /*
    if (COAPRegisterPatternsForProtocolDetection() < 0 )
        return;
    */
    //app-layer protocol port detection registration.
        if (!AppLayerProtoDetectPPParseConfPorts("udp", IPPROTO_UDP,
                                                 proto_name, ALPROTO_COAP,
                                                 0, sizeof(CoapHeader),
                                                 COAPProbingParser, COAPProbingParser)) {
        }
}

//If conditional is true then register various parser functions. These function are a
//two-dimentional (Flow * App_layer) array variables.
if (AppLayerParserConfParserEnabled("udp", proto_name)) {
```

11. In AppLayerParserConfParserEnabled(), If conditional is true then register various parser functions. These function are assigned to a two-dimensional (Flow * App_layer) array variables
12. COAPStateAlloc() allocates the COAP state memory
13. COAPStateFree() used to frees the CoAP state memory
14. COAPGetTxDetectState() return the DetectEngineState (coap_state->de_state), an instance of CoapState
15. COAPSetTxDetectState() function set the given detect engine state
16. Enum and struct use in “app-layer-protos.h” are
    Enum where we need to define CoAP application data events

```
enum {
    COAP_DECODER_EVENT_INVALID_PROTOCOL_ID,
    COAP_DECODER_EVENT_UNSOLICITED_RESPONSE,
    COAP_DECODER_EVENT_INVALID_LENGTH,
    COAP_DECODER_EVENT_INVALID_VALUE,
    COAP_DECODER_EVENT_VALUE_MISMATCH,
};
```

17. CoAP Con message struct which have all flags related to the Con message

```
typedef struct CoapConMsg_ {

    uint16_t    protoNameLen;   /*COAP Protocol name length*/
    uint8_t    *protoName;      /*COAP Protocol Name*/
    uint8_t     version;        /*COAP Protocol Version*/
    uint8_t     conFlag;
    uint8_t     uname;
    uint8_t     pass;
    uint8_t     retain;
    uint8_t     willFlag;
    uint8_t     cleanSess;
    uint16_t    kalive;
    uint16_t    clientIDLen;
    uint8_t     *clientID;

} CoapConMsg;
```

18. ConAck which define connection acknowledgement type of CoAP message

```
typedef struct CoapConAckMsg_ {
    uint8_t     returnCode;         /*COAP Return Code*/
} CoapConAckMsg;
```

19. This define the CoAP subscribe message type

```
typedef struct CoapSubMsg_ {

    uint8_t     dupflag;
    uint8_t     QoSFlag;
    uint16_t    mid;
    uint16_t    topicLen;
    uint8_t     *topic;
    uint8_t     QoS;

} CoapSubMsg;
```

20. CoAP subscribe acknowledgement struct flags

```
typedef struct CoapSubAckMsg_ {

    uint16_t    mid;
    uint8_t     QoS;

} CoapSubAckMsg;
```

21. This struct define the CoAP public message flags

```
typedef struct CoapPubMsg_ {

    uint8_t     dupflag;
    uint8_t     QoS;
    uint8_t     retain;
    uint16_t    topicLen;
    uint8_t     *topic;
    uint8_t     *message;

} CoapPubMsg;
```

22. COAP Transaction Structure, request/response are define here

```
typedef struct CoapTransaction_ {
    struct CoapState *coap;

    uint64_t    tx_num;         /**< intern
    uint32_t    logged;         /**< flags
    uint16_t    transactionId;
    uint8_t     ver;
    uint8_t     pktType;
    uint8_t     tknLength;        /*Type of
    uint8_t     methodCode;         /*Lengt
    uint8_t     responseCode;
    uint8_t     pktCodeClass;
    uint8_t     pktCodeDetail;
    uint16_t    msgID;

    uint8_t    *token;

    uint8_t     opt1Delta;
    uint8_t     opt1Length;
    uint8_t     *opt1Value;

    uint8_t    *payload;


    AppLayerDecoderEvents *decoder_events;
    DetectEngineState *de_state;

    TAILQ_ENTRY(CoapTransaction_) next;
} CoapTransaction;
```

23. In this struct we have to define CoAP State Structure.

```
typedef struct CoapState_ {
    TAILQ_HEAD(, CoapTransaction_)    tx_list;    /**< transaction list */
    CoapTransaction                   *curr;      /**< ptr to current tx */
    uint64_t                          transaction_max;
    uint16_t                          events;
    uint8_t                           givenup;    /**< bool indicating flood. */
} CoapState;
```

24. Also we have to define a CoAP message command, we replace a string message with an integer number, so we can use it easily.

- COAP Different Message Types.

```
#define COAP_MSG_TYP_CONFIRM                    0
#define COAP_MSG_TYP_NON_CONFIRM                1
#define COAP_MSG_TYP_ACK                        2
#define COAP_MSG_TYP_RST                        3
```

- COAP Different Methods of code.

```
#define COAP_MSG_METHOD_CODE_GET                1
#define COAP_MSG_METHOD_CODE_POST               2
#define COAP_MSG_METHOD_CODE_PUT                3
#define COAP_MSG_METHOD_CODE_DELETE             4
```

- COAP Different Response.

```
#define COAP_MSG_RESPONSE_CODE_CREATED                  65
#define COAP_MSG_RESPONSE_CODE_DELETED                  66
#define COAP_MSG_RESPONSE_CODE_VALID                    67
#define COAP_MSG_RESPONSE_CODE_CHANGED                  68
#define COAP_MSG_RESPONSE_CODE_CONTENT                  69
#define COAP_MSG_RESPONSE_CODE_BAD_REQUEST              128
#define COAP_MSG_RESPONSE_CODE_UNAUTHORIZED             129
#define COAP_MSG_RESPONSE_CODE_BAD_OPTION               130
#define COAP_MSG_RESPONSE_CODE_FORBIDDEN                131
#define COAP_MSG_RESPONSE_CODE_NOT_FOUND                132
#define COAP_MSG_RESPONSE_CODE_METHOD_NOT_ALLOWED       133
#define COAP_MSG_RESPONSE_CODE_NOT_ACCEPTABLE           134
#define COAP_MSG_RESPONSE_CODE_PRECONDITION_FAILED      140
#define COAP_MSG_RESPONSE_CODE_REQUEST_ENTITY_TOO_LARGE 141
#define COAP_MSG_RESPONSE_CODE_UNSUPPORTED_CONTENT_FORMAT 142
#define COAP_MSG_RESPONSE_CODE_INTERNAL_SERVER_ERROR    160
#define COAP_MSG_RESPONSE_CODE_NOT_IMPLEMENTED          161
#define COAP_MSG_RESPONSE_CODE_BAD_GATEWAY              162
#define COAP_MSG_RESPONSE_CODE_SERVICE_UNAVAILABLE      163
#define COAP_MSG_RESPONSE_CODE_GATEWAY_TIMEOUT          164
#define COAP_MSG_RESPONSE_CODE_PROXYING_NOT_SUPPORTED   165
```

25. After registering the CoAP protocol in suricata , now we have to enable the detection of the CoAP protocol and its different types of keywords in suricata.

- We have to add a list of keyword use in the CoAP protocol in “detect-engine-register.h”. These keywords are used in detection process

```
DETECT_AL_COAP_TOKEN_LEN,
DETECT_AL_COAP_VERSION,
DETECT_AL_COAP_PACKET_TYPE,
DETECT_AL_COAP_TOPIC,
DETECT_AL_COAP_MESSAGE_ID,
DETECT_AL_COAP_OPT_DELTA,
DETECT_AL_COAP_OPT_LENGTH,
DETECT_AL_COAP_PAYLOAD,
DETECT_AL_COAP_METHOD_CODE,
DETECT_AL_COAP_RESPONSE_CODE,
```

- After this we have to create a detect “.c” and “.h” file for each keyword like

```
C detect-coap-payload.c
C detect-coap-payload.h
```

- In the “.c” file, we have to create a register function for the detection of the keyword and this function is use in the main detection file

```
/**
 * \brief Registration function for coap keyword
 */
void DetectCoapPayloadRegister(void)
{
    DetectCoapRegister();
}
```

- SigTableSetup(void) in “Detect-engine-register.c” is the main function where we add the register function for the CoAP protocol keywords

```
DetectTargetRegister();
DetectTemplateRustBufferRegister();
DetectTemplateBufferRegister();
DetectBypassRegister();
DetectCoapTokenLenRegister();
DetectCoapVersionRegister();
DetectCoapPktTypeRegister();
DetectCoapTopicRegister();
DetectCoapMessageIdRegister();
DetectCoapOptDeltaRegister();
DetectCoapOptLengthRegister();
DetectCoapPayloadRegister();
DetectCoapMethodCodeRegister();
DetectCoapResponseCodeRegister();
```

Every CoAP keyword detection register perform this type of functionality

- In DetectCoapMethodCodeRegister(void) first fill the table where the keyword name, description, setup function where message is parsed and free memory function that free memory associated with DetectCoapMethodCode are needed to be defined

```
sigmatch_table[DETECT_AL_COAP_METHOD_CODE].name          = "coap_method";
sigmatch_table[DETECT_AL_COAP_METHOD_CODE].desc          = "Match COAP method code";
sigmatch_table[DETECT_AL_COAP_METHOD_CODE].url           = NULL;
sigmatch_table[DETECT_AL_COAP_METHOD_CODE].Match         = NULL;
sigmatch_table[DETECT_AL_COAP_METHOD_CODE].Setup         = DetectCoapMethodCodeSetup;
sigmatch_table[DETECT_AL_COAP_METHOD_CODE].Free          = DetectCoapMethodCodeFree;
sigmatch_table[DETECT_AL_COAP_METHOD_CODE].RegisterTests = NULL;
```

- DetectCoapPktTypeParse() parse the CoAP message and match the defined keywords of CoAP with the parse data and check that it valid keyword value

```
ret = pcre_exec(msg_type_parse_regex, msg_type_parse_regex_study, str, strlen(str), 0, 0, ov, MAX_SUBSTRINGS);
```

```
/* We have a correct Coap message type option */
coap = (DetectCoapMethodCode *) SCCalloc(1, sizeof(DetectCoapMethodCode));
if (unlikely(coap == NULL))
    goto error;

    if (strcmp("get", ptr) == 0)
        coap->methodCode = COAP_MSG_METHOD_CODE_GET;
    else if (strcmp("post", ptr) == 0)
        coap->methodCode = COAP_MSG_METHOD_CODE_POST;
    else if (strcmp("put", ptr) == 0)
        coap->methodCode = COAP_MSG_METHOD_CODE_PUT;
    else if (strcmp("delete", ptr) == 0)
        coap->methodCode = COAP_MSG_METHOD_CODE_DELETE;
```

- This function is used in DetectCoapMethodCodeSetup() that used is to add the parsed "id" option into the current signature

```
/* Okay so far so good, lets get this into a SigMatch and put it in the Signature. */
sm = SigMatchAlloc();
if (sm == NULL)
    goto error;

sm->type    = DETECT_AL_COAP_METHOD_CODE;
sm->ctx     = (void *) coap;

SigMatchAppendSMToList(s, sm, g_coap_buffer_id);

SCReturnInt(0);
```

- DetectAppLayerInspectEngineRegister() in DetectCoapMethodCodeRegister(void) register inspect engine at start up time

26. After the change in code now we have to add the newly added files support into the suricata make file, so suricata also compiles the files created by us.

- add this code in make file to add the support of our created files in suricata

```
  detect-modbus.$(OBJEXT) detect-coap-token-len.$(OBJEXT) \
  detect-coap-version.$(OBJEXT) detect-coap-packet-type.$(OBJEXT) \
detect-coap-topic.$(OBJEXT) detect-coap-message-id.$(OBJEXT) \
detect-coap-opt-delta.$(OBJEXT) detect-coap-opt-length.$(OBJEXT) \
detect-coap-payload.$(OBJEXT) detect-coap-method.$(OBJEXT) \
detect-coap-response.$(OBJEXT) detect-xbits.$(OBJEXT) \
```

```
detect-coap-relation.h \
detect-coap-token-len.c detect-coap-token-len.h\
detect-coap-version.c detect-coap-version.h\
detect-coap-packet-type.c detect-coap-packet-type.h\
detect-coap-topic.c detect-coap-topic.h\
detect-coap-message-id.c detect-coap-message-id.h\
detect-coap-opt-delta.c detect-coap-opt-delta.h\
detect-coap-opt-length.c detect-coap-opt-length.h\
detect-coap-payload.c detect-coap-payload.h\
detect-coap-method.c detect-coap-method.h\
detect-coap-response.c detect-coap-response.h\
```

```
include ./$(DEPDIR)/detect-coap-token-len.Po
include ./$(DEPDIR)/detect-coap-version.Po
include ./$(DEPDIR)/detect-coap-packet-type.Po
include ./$(DEPDIR)/detect-coap-topic.Po
include ./$(DEPDIR)/detect-coap-message-id.Po
include ./$(DEPDIR)/detect-coap-opt-delta.Po
include ./$(DEPDIR)/detect-coap-opt-length.Po
include ./$(DEPDIR)/detect-coap-payload.Po
include ./$(DEPDIR)/detect-coap-method.Po
include ./$(DEPDIR)/detect-coap-response.Po
```

```
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/detect-coap-token-len.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/detect-coap-version.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/detect-coap-packet-type.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/detect-coap-topic.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/detect-coap-message-id.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/detect-coap-opt-delta.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/detect-coap-opt-length.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/detect-coap-payload.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/detect-coap-method.Po@am__quote@
@AMDEP_TRUE@@am__include@ @am__quote@./$(DEPDIR)/detect-coap-response.Po@am__quote@
```

- These are the newly added files in “src” directory to enable CoAP protocol support.

app-layer-coap.c
app-layer-coap.h
detect-coap-compare.h
detect-coap-message-id.c
detect-coap-message-id.h
detect-coap-method.c
detect-coap-method.h
detect-coap-opt-delta.c
detect-coap-opt-delta.h
detect-coap-opt-length.c
detect-coap-opt-length.h
detect-coap-packet-type.c
detect-coap-packet-type.h
detect-coap-payload.c
detect-coap-payload.h
detect-coap-relation.h
detect-coap-response.c
detect-coap-response.h
detect-coap-token-len.c
detect-coap-token-len.h
detect-coap-topic.c
detect-coap-topic.h
detect-coap-version.c
detect-coap-version.h

27. Changing in suricata.yaml

```yaml
port-groups:
  HTTP_PORTS: "80"
  SHELLCODE_PORTS: "!80"
  ORACLE_PORTS: 1521
  SSH_PORTS: 22
  DNP3_PORTS: 20000
  MODBUS_PORTS: 502
  FILE_DATA_PORTS: "[$HTTP_PORTS,110,143]"
  FTP_PORTS: 21
  COAP_PORTS: 5683
```

```
# DNP3
dnp3:
  enabled: no
  detection-ports:
    dp: 20000

coap:
  enabled: yes
  detection-ports:
    dp: 5683
```

28. Add rule for CoAP

Now add an example rule in suricata.rules file to test CoAP support in suricata

```
alert coap any any -> any any (msg:"hello"; coap_method: post; sid:121212;)
```

Installing Suricata

- Extract “suricata_with_coap_enabled.tar.gz” is a directory
- Use this command to go inside the directory ‘’cd suricata-4.1.2”
- Install dependencies

sudo apt update sudo apt-get install libpcre3-dbg libpcre3-dev autoconf automake libtool libpcap-dev libnet1-dev libyaml-dev libjansson4 libcap-ng-dev libmagic-dev libjansson-dev zlib1g-dev pkg-config rustc cargo

- Now configure the suricata using

./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var

- After this install suricata

make
sudo make install
sudo make install-conf

29. Run Suricata

Use this command to run suricata for pcap

```
sudo suricata -c /etc/suricata/suricata.yaml -v -r /home/iot/Downloads/coap.pcap
```