Modbus Inspector

Modbus Inspector Overview

Type	Inspector (service)
Usage	Inspect
Instance Type	Multiton
Other Inspectors Required	`stream_tcp`
Enabled	`false`

The Modbus protocol defines a communication standard to exchange messages between a Supervisory Control and Data Acquisition (SCADA) system and a Programmable Automation Controller (PLC). The Modbus protocol uses TCP port 502.

The `modbus` inspector detects and analyzes Modbus messages in network traffic. When enabled, the Modbus intrusion rule options provide access to certain Modbus protocol fields.

Best Practices for Configuring the Modbus Inspector

If your network does not contain an enabled Modbus device, you should not enable the `modbus` inspector in a network analysis policy that you apply to traffic.

Modbus Inspector Parameters

Modbus TCP port configuration

The `binder` inspector defines the Modbus TCP port configuration. For more information, see the Binder Inspector Overview.

Example:

```
[
    {
        "when": {
            "role": "server",
            "proto": "tcp",
            "ports": "502"
        },
        "use": {
            "type": "modbus"
        }
    },
    {
        "when": {
            "role": "any",
            "service:" "modbus"
        },
        "use": {
            "type":"modbus"
        }
    }
]
```

Note The `modbus` inspector does not provide any parameters.

Modbus Inspector Rules

Enable the `modbus` inspector rules to generate events and, in an inline deployment, drop offending packets.

Table 1: Modbus Inspector Rules

GID:SID	Rule Message
144:1	length in Modbus MBAP header does not match the length needed for the given function
144:2	Modbus protocol ID is non-zero
144:3	reserved Modbus function code in use

Modbus Inspector Intrusion Rule Options

You can use a `modbus` option alone or in combination with the `content` and `byte_jump` intrusion rule options.

modbus_data

Sets the data cursor to the beginning of the Modbus `Data` field.

Syntax: `modbus_data;`

Examples: `modbus_data;`

modbus_func

Verifies that the Modbus `Function` field matches the specified Modbus function code. You can set a positive integer or string literal to represent a Modbus function code.

Type: string

Syntax: `modbus_func: <function>;`

Valid values:

Table 2: Modbus Function Code Values

Code	String
1	read_coils
2	read_discrete_inputs
3	read_holding_registers
4	read_input_registers
5	write_single_coil
6	write_single_register
7	read_exception_status
8	diagnostics
11	get_comm_event_counter
12	get_comm_event_log
15	write_multiple_coils
16	write_multiple_registers
17	report_slave_id
20	read_file_record
21	write_file_record

Code	String
22	mask_write_register
23	read_write_multiple_registers
24	read_fifo_queue
43	encapsulated_interface_transport

Examples:

```
modbus_func: read_coils;
modbus_func: 8;
```

modbus_unit

Verifies that the Modbus Unit ID in the message matches the specified unit ID. You can set a number to represent the Modbus Unit ID.

Type: integer

Syntax: `modbus_unit: <unit_id>;`

Valid range: `0` to `255`

Examples:

```
modbus_unit: 1;
```