Towards Understanding Man-on-the-Side Attacks (MotS) in SCADA Networks

Peter Maynard and Kieran McLaughlin
Centre for Secure Information Technology, Queen's University Belfast, U.K.

Keywords: HTTP, ICS, IEC 60870-5-104, Man-in-the-Middle, Man-on-the-Side, SCADA, Off-path.

Abstract: We describe a new class of packet injection attacks called Man-on-the-Side (MotS), previously only seen where state actors have "compromised" a number of telecommunication companies. MotS injection attacks have not been widely investigated in scientific literature, despite having been discussed by news outlets and security blogs. MotS came to attention after the Edward Snowden revelations, which described large scale pervasive monitoring of the Internet's infrastructure. For an advanced adversary attempting to interfere with IT connected systems, the next logical step is to adapt this class of attack to a smaller scale, such as enterprise or critical infrastructure networks. MotS is a weaker form of attack compared to a Man-in-the-Middle (MitM). A MotS attack allows an adversary to read and inject packets, but not modify packets sent by other hosts. This paper presents practical experiments where we have implemented and performed MotS attacks against two testbeds: 1) on HTTP connections, by redirecting a victim to a host controlled by an adversary; and 2) on an Industrial Control network, where we inject falsified command responses to the victim. In both cases, the victims accept the injected packets without generating a suspiciously large number of unusual packets on the network. We then perform an analysis of three leading Network Intrusion Detection Systems (IDSs) to determine whether the attacks are detected, and discuss mitigation methods.

1 INTRODUCTION[1]

The Snowden revelations have been a driving factor towards a more secure digital world. One clear example of this is that TLS has become widely deployed on systems which previously did not use encryption. This paper builds upon leaked information regarding MotS attacks. MotS has traditionally been considered only achievable by a global pervasive adversary (Trammell et al., 2015), such as a state actor, who is capable of using their global timing advantage to inject forged packets before a victim receives legitimate packets, which are then correctly discarded by the target. In contrast, the experiments presented here investigate a localised implementation of MotS attacks against two protocols, HTTP and IEC 60870-5-104 (IEC104). IEC104 is widely used within Europe and Asia for controlling and monitoring critical infrastructure e.g. gas, electricity, etc. With state actors known to be targeting infrastructure such as electricity grids, our motivation is to better understand how such adversaries may adopt MotS attack techniques, which they are known to have used in the past in different circumstances.

In 2013, a NIST report (Forrester, 2013) advocated the improvement of industrial networks through the use of zero-trust networks. DeCusatis et.el (DeCusatis et al., 2016) discussed how to implement a zero-trust network, which would mitigate this class of attacks. DeCusatis implemented a policy engine and gateway which prevents unauthorised packets from being accepted by a victim, as well as ensuring that the attacker does not receive identifiable information when their probe is dropped. However, within in the Industrial Control System (ICS) domain, it is not possible to quickly deploy radical changes as seen with TLS. Information from the National Grid (National Grid, 2013) shows that this major upgrade of critical networks is not possible in the time frame laid out. Due to the long life cycle of ICS equipment and the criticality of the physical systems they control, many security recommendations for IT networks are not suitable for Operational Technology (OT) networks. As a consequence OT networks frequently rely on unencrypted protocols. Threats are typically

[1]Reproducible experiments can be found at:
https://github.com/PMaynard/mots

Maynard, P. and McLaughlin, K.
Towards Understanding Man-on-the-Side Attacks (MotS) in SCADA Networks.
DOI: 10.5220/0009782302870294
In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications (ICETE 2020) - SECRYPT*, pages 287-294
ISBN: 978-989-758-446-6

mitigated with the use of Virtual Private Networks (VPNs), air gapped networks, network segmentation, firewalls, and Demilitarised Zones (DMZs).

However, industrial control networks are often targeted by state actors, as was the case with the TRISIS malware (Johnson et al., 2017). This was a Remote Access Trojan (RAT) that targeted Triconex Safety Instrumented System (SIS) devices, and triggered a shut down of an industrial site in the Middle East. Another example is CRASHOVERRIDE (Dragos, 2017), which performed network enumeration and exploitation of Supervisory Control And Data Acquisition (SCADA) protocols including IEC104. This malware has been linked to the Ukrainian power outages in 2016. Therefore, it is logical to assume that MotS has, or is likely to be, deployed by similar adversaries into an industrial network at some point in time.

Many papers that propose novel network IDSs, tend verify their effectiveness using the outdated KDD'99 dataset or common attacks (i.e. MitM, replay and injection (Maynard et al., 2014; Green et al., 2017)). We provide a modern technique which should be considered when validating intrusion detection methods. We investigate the detection responsiveness of the current state of the art network IDSs (Snort, Zeek, Suricata) when confronted with MotS, for both HTTP and IEC104. We provide reproducible experiments and a packet captures of our attacks, so they may be used to validate other IDSs.

The paper continues with an overview of related work (§2) and background (§3), which details what required for a MotS attack, and introduces ICS/SCADA networking architecture. Following on to the experimentation methodology (§4), and experimental results (§5). Finally, a review of current detection systems (§6) and conclusion (§7).

2 RELATED WORK

In 2012, Gilad et al. (Gilad and Herzberg, 2012) proved that it is possible to inject packets into a TCP stream, by using browser based malware to identify the IP-ID, then spoof a request. Despite reports from IETF RFC 6528, which state that TCP segment injection is mitigated by having the majority of data needed to perform this "not well known" to the attacker. While the IETF suggests VPN or TCP Authentication Option (2010) to prevent injection. At the same, time the IETF acknowledges the TCP reset vulnerability is a known issue. As of 2019, Alexander et al. (Alexander et al., 2019) can identify the IP-port four-tuple representing an active TCP connection, by taking advantage of side-channels within the Linux kernel 4.0 and above. Packet injection and TCP resets are a widely known issue, with current mitigations easily bypassed. As indicated in the introduction, the existing state of the art IDSs for ICS networks are not validated against these types of attacks.

The Chinese Great Firewall exploits the TCP reset vulnerability, and is discussed by Weaver et al. (Weaver et al., 2009) (2009) who propose mitigations. Also in 2012, Victor Julien added a detection rule for overlapping data to the open source network IDS Suricata. Essentially, detecting TCP injection and MotS attacks as seen in the detection section. In 2014, Gilad et al. (Gilad and Herzberg, 2014) improved upon their previous TCP injection attacks, so it does not require the use of web malware. They also detail the history of TCP injection attacks. While Gilad et al. focus on injecting attacks over the internet, we choose to focus on exploiting a target on the local network, assuming that it has been compromised by a skilled adversary.

Between 2013 and 2014, The Guardian and The Intercept (Schneier, 2013; Gallagher and Greenwald, 2014) discuss a nation state operation called "QUANTUM Inject", which uses their wide scope to monitor and inject packets into TCP streams. Following this media attention, the security company Fox-IT performed a deep dive into QUANTUM Inject (Haagsma, 2015). They developed a PoC and released patches for a number of open source network IDSs. In 2015, a report from the IETF (Trammell et al., 2015) discussed a threat model of a passive pervasive attacker, explicitly stating MotS as one example. The company NetReSec, which performs network security training and develops a packet analyser tool, have written a number of times about MotS between 2015-16 (Hjelmvik, 2015). They attempted to raise awareness of the attack, and developed additional detection tools as well as identifying TCP injection attacks from China.

Marczak et al. (Marczak et al., 2015) likened the Distributed Denial of Service (DOS) attack from "China's Great Cannon" to the QUANTUM system. In that the techniques used to inject packets are similar to the QUANTUM Inject method. Finally, the closest related work the authors identified was by Nakibly et al. (Nakibly et al., 2016), who detail MotS attacks being used to inject Javascript, HTTP redirections and malware dropping. They performed a study to analyse incoming and outgoing connections of four university institutions, and identified several different threat groups. They proposed a few MotS detection methods, which use the IP features Time To Live (TTL) and IP ID, and timing analysis to identify MotS

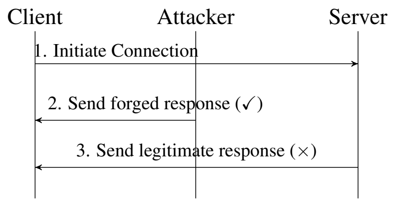

Figure 1: Sequence Diagram of Man-on-the-Side (MotS) attack.

in the wild. The authors are unaware of any papers detailing MotS class of attack on industrial networks, and are certain there is no academic literature that provides reproducible experiments and packet captures that may be used to validate new IDSs.

3 BACKGROUND

MotS, like SYN-flooding and DNS-Spoofing attacks, are known as *off-path*. This differs from traditional MitM attacks (Conti et al., 2016) which are considered *on-path*. An *on-path* attacker needs to control the links between the victim and host, while an *off-path* attacker does not. Although MotS is a weaker class of attack than MitM, it is capable of packet injection, without having to maintain connections as a proxy. Previous work by the authors (Maynard et al., 2014) which used Address Resolution Protocol (ARP) spoofing to maintain an *on-the-path* attack, required a large number of ARP packets, as well as the need to continuously forward packets to the victim. Depending on the network and host, this may be easily detectable. It can also be resource intensive. On the other hand, a MotS attack can be performed with a minimum of one forged packet. Also, MotS can not be mitigated by locking down switch ports using Media Access Control (MAC) Access Control Lists (ACLs), which can mitigate MitM.

A MotS attack can be successfully performed if the attacker can: a) observe a victim's request; b) adequately replicate a response that the victim would accept; and c) transmit a response to the victim quicker than legitimate response. If these three requirements are met, then an adversay can perform this attack. Fundamentally, this attack exploits the design of TCP (RFC 793), whereby it is normal for TCP segments to arrive more than once, and that the first segment that arrives is accepted. The attacker needs to be in a position to know the four-tuple (Local IP address, Local Port, Remote IP address, Remote Port) that identifies the target connection, along with the next sequence number expected by the target.

Figure 1 shows the sequence diagram of a MotS

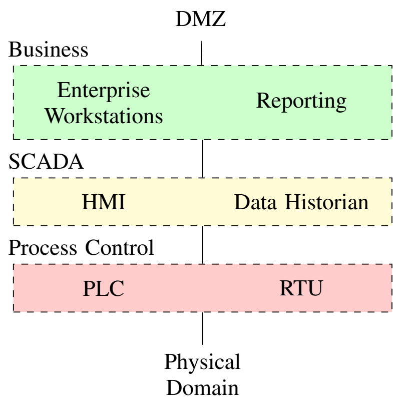

Figure 2: A high level network diagram of an Industrial Control System (ICS).

attack, consisting of a Client, Server and Attacker. First, the Client initiates a connection with the server and a TCP handshake is performed, followed by a request to the server. The attacker is positioned within the network so that they can view this request (the attacker does not need to view the handshake). The attacker creates and sends a forged response, which spoofs the four-tuple information, including the TCP sequence number. The forged packet is accepted by the client. Subsequently the legitimate response arrives and is dropped by the client, as it has already acknowledged the forged packet.

3.1 Industrial Control System Networks

The terms Industrial Control System (ICS) and Supervisory Control And Data Acquisition (SCADA) are often used interchangeably. However, SCADA may be considered part of an ICS, which is used to monitor and control physical devices. Figure 2 shows a high level network diagram of an ICS network, that highlights the different network segments, which have limited access between each other.

In the diagram, the SCADA enclave is located in the middle and provides information to the business network, as well as monitoring the process control enclave. A Human Machine Interface (HMI) may be used to monitor the physical process, while a Data Historian maintains a record of plant operations. The process control enclave contains a number of devices used to govern physical instruments and actuators that interact with the physical domain. Typical devices include Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTUs) which are rugged embedded devices designed for a distinct task. The network may consist of serial links, Ethernet, and IP, as well as field bus protocols; one such protocol is IEC104 (International Electrotechnical Commission, 2006). If MotS were to be performed on ICS network, the risks to the system may be:

- **Unauthorised Access to Information**: Redirect information to the adversary, further gaining an insight into the operation of the system.
- **Unauthorised Modification or theft of Information**: Bypassing controls an injecting modified information.
- **Denial of Service or Prevention of Authorised Access**: By injecting packets that cause the device to trigger a DOS.
- **Denial/Claim of Action that Took/Not Took Place**: By preventing legitimate responses from arriving the adversary can force records into an incomplete, or fictitious state.

To perform these attacks on an industrial site without causing any adverse effects on the system would require a lot of planning and resources. The exact physical device would need to be tested, and running the same program code as the live system, while also ensuring the devices are networked similarly. However, the attack may also be performed blindly without all the information, yet the exact results may be unknown.

4 METHODOLOGY

When performing these experiments we assume that a highly skilled adversary has compromised the network and has positioned themselves so they can monitor and inject packets at any point of the network. This is consistent with previously reported intrusions of state actors within industrial environments, where it is viable for an adversary to compromise the network infrastructure, such as switches and routers, and have this level of provenance. Each experiment is configured so that there is a: client, server, and attacker. The attacker is positioned so that they are in between the client and server, this is ensured by introducing an artificial delay of 500ms to the responding device. This allows us to validate the methods used to inject the forged packets. In an industrial setting, the responding device may be in a remote location, using several communication mediums to respond to the operator, such as radio or public internet, providing an attacker with multiple opportunities. Once the injection has been completed, a recorded packet capture is parsed by the three IDSs, and their results manually analysed. The experiments will focus more on the IEC104 protocol over HTTP since our aim is to prove the possibility of applying this attack on protocols other than HTTP(Haagsma, 2015), and within a localised network.

5 EXPERIMENTS

We performed four different MotS attacks on the two protocols, HTTP and IEC104. The HTTP experiments inject a false response then redirect the client. The IEC104 experiments consider the injection of a recorded response, and a completely forged response.

5.1 HTTP

HTTP is a stateless protocol and is generally closed after each request/response is completed, resulting in the TCP handshake has to be performed for each request. With MotS, it is possible to target HTTP connections based on IP addresses, as well as a specific user by triggering on cookie headers (Gallagher and Greenwald, 2014) and other identifiable data sent within a HTTP connection. The network layout consists of three hosts (Client, Server, and Attacker) and a switch. A 500ms artificial delay has been added to the server to ensure messages sent by the attacker will reliably arrive ahead of the legitimate response. When the client initiates a HTTP GET request, the attacker generates a forged packet based IP/TCP fields of the request. Then, the attacker reverses the direction of the IP/TCP source and destination addresses/ports, while the remainder of the IP header is unchanged. A random IP ID is generated, and the forged packet's TCP sequence number is set to the request's TCP acknowledgement, and the forged packet's TCP acknowledgement to the request's TCP sequence number. Finally, the forged payload is injected and all checksums calculated before being transmitted to the client. The legitimate response from the server arrives at the client afterwards, and is consequently dropped by the client.

5.1.1 Experiment 1: Inject Response Page

The forged response contains a simple HTML page with a heading. Instead of the legitimate page, the attacker could inject malicious JavaScript, as seen by (Nakibly et al., 2016). The result of this exchange is that the client renders the forged HTTP response in the web browser, while the legitimate response is dropped without the browser being notified. The connection needs to be terminated by sending the TCP flags `FIN ACK`, which shuts down the connection. It is worth noting that if the attack uses `PUSH ACK` TCP flags, which are the flags set under normal circumstances, the TCP connection will remain open and the two payloads are rendered as one by the browser. This happens even if the HTTP header `Connection: Close` is sent.

5.1.2 Experiment 2: Inject Redirect

This time the forged responses contain a ”HTTP 301 Moved Permanently”, instead of ”HTTP 200 OK”. This status line requires the header field `Location`, along with a URI for redirection. Once received, the browser will automatically open and render the specified URI. This can be used to redirect the user to an attacker-controlled site, which has obvious security implications. The outcome is that the client follows the 302 redirection URI, and drops the legitimate response. Additionally, due to the nature of HTTP, the server records the Get request into the server logs, and is not aware that the client did not receive the response. Consequently, the only way to detect this happening is to monitor the network layer.

5.2 IEC 60870-5-104

IEC104 (International Electrotechnical Commission, 2006) is a protocol for transporting IEC 60870-5-101 (IEC101) frames over TCP. IEC101 is a plaintext telecontrol protocol designed for serial links. In these experiments we will create forged IEC104 packets with IEC101 payloads. The structure of an IEC104 packet, comprises two segments, the Application Service Data Unit (ASDU) and Application Protocol Control Information (APCI). The ASDU contains the data to be exchanged, represented as Information Elements, and data about the number of information elements are contained within the Data Unit Identifier (DUI). The APCI is sent with all packets and is used to specify the start/end bits along with the ASDU length. The protocol manages protection against loss and duplication of messages using a counter mechanism. Section 5.1 of EN 60870-5-104:2006 (International Electrotechnical Commission, 2006) specifies that a send sequence number N(S) and a receive sequence number N(R) are used based on the method defined in ITU-T X.25. Both sequence numbers are incremented by one for each Application Protocol Data Unit (APDU) and each direction. These are acknowledged by the receiver, by returning the receive sequence number N(R) to the sender using the supervisory format. Once acknowledged, the sender can remove the correctly transmitted APDUs from its buffer.

Figure 3 describes the SCADA testbed used to perform the experiments. The attacker is located on the same Process Control enclave as the PLC, while the victim, the HMI is located within the SCADA enclave. The attacker uses the port mirroring feature of the Process Control switch to monitor the network. The attacker builds the forged packet from the

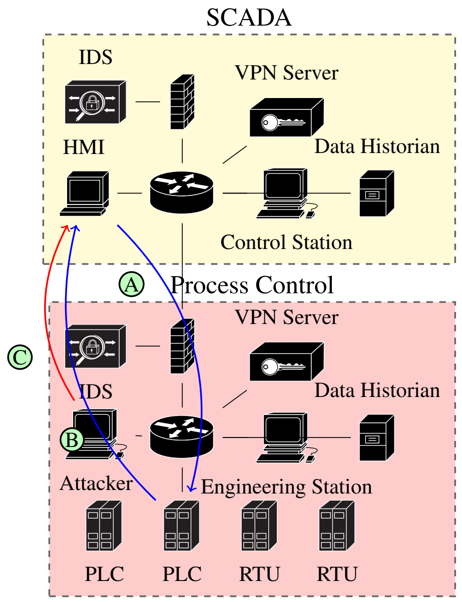

Figure 3: A generalised network diagram showing two enclaves: SCADA and Process Control.

initial General Interrogation (GI) request sent by the HMI to the PLC (Figure 3: Step A), by switching the MAC and IP source/destination addresses, and switching the TCP sequence and acknowledgement numbers. Additionally, the send and receive counters for IEC104 are calculated to match the expected response. The attacker then waits for the PLC to send an `ActCon` (Figure 3; Step B), before sending the forged response to the HMI (Figure 3; Step C). In the case of IEC104, unlike the previously described HTTP attacks, the TCP flags remain as `PUSH ACK`, which will be explained later.

These experiments are based on a GI (GI and C_IC_NA_1 will be used interchangeably) of a PLC initiated from the HMI. In normal operation, the PLC would return its current status in response to a GI. The testbed is configured to generate and return data that simulates a real world response. This contains single-point, double-points, and step-position information. The objective of these experiments is to investigate how an attacker can forge a fake response to change the data viewed by a HMI operator.

Figure 4 is a sequence diagram detailing the steps of a GI. After the TCP handshake is successful, the sender will initialise the connection for data transmission using the `STARTDT = act` command, if successful the receiving station will respond with `STARTDT = con`. Next, the sender performs a GI request, using an ASDU frame (`C_IC_NA_1 = act`), which is acknowledged by receiving end as `C_IC_NA_1 = act con`, followed by the interrogation data. The dotted line is where the

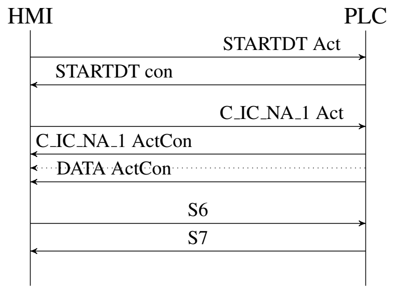

Figure 4: Sequence diagram of a Human Machine Interface (HMI) performing an General Interrogation (GI) command on a Programmable Logic Controller (PLC). The dotted line shows the injected response.

forged packets are injected within the sequence. Finally, the data is confirmed using the Supervisory format N(S), containing the number of frames received.

5.2.1 Experiment 3: Inject a Replayed Response

This experiment will replay a pre-captured response from the PLC, matching the same number of APCI/ASDU responses expected from the PLC, so that an attacker can hide the physical state from the operator. Effectively, this attack will force the data visible to the operator to become out of sync with the physical process, in a similar way that Stuxnet hid real-time information from operators. Since this is a replay attack, the injected response will contain the same number of information elements, except the values will be reused from an earlier transmission. Listing 1 shows a packet capture of this experiment. The first packets are the TCP handshake (1-3), followed by the `STARTDT act` and `STARTDT con` messages (4-6) to start data transmission on this connection from the HMI to the PLC. The HMI sends a GI (7), and at this point, the attacker generates the forged response. The PLC then acknowledges the GI (8-9), triggering the attacker to send their forged response (10). The forged response is accepted by the HMI (11), after which the legitimate GI response arrives (12), and is marked in the packet capture as `TCP Spurious Retransmission`. The HMI responds to the PLC with a TCP ACK (13), which is marked as `TCP Dup ACK`, since it was already used by the forged response.

After the replayed packet has been accepted, the PLC software hangs, then drops because the TCP sequence number becomes out of sync due to the injected segment. Although, as was the case in the HTTP experiments, it is possible to close the connection, the operation of IEC104 normally requires long lived connections. Deliberately bringing down this connection with a TCP reset could alert a system administrator to an issue or intrusion within the network. While the result is that the HMI operator is now viewing stale data, the connection will be dropped and a redundant connection used in its place. The IEC104 standard states that several connections may be maintained with the remote device, and periodically refreshed using the `STARTDT` command. Allowing the connection to degrade using the time out mechanism, overusing the TCP RST method, reduces the likelihood of triggering an alert.

Listing 1: Packet Capture of IEC104 Experiment 3.

```
1   10.50.50.103 > 10.50.50.99  TCP 74 39394 > 2404 [SYN]
        Seq=0 Win=29200 Len=0
2    10.50.50.99 > 10.50.50.103 TCP 74 2404 > 39394 [SYN,
         ACK] Seq=0 Ack=1 Win=8192 Len=0
3   10.50.50.103 > 10.50.50.99  TCP 66 39394 > 2404 [ACK]
        Seq=1 Ack=1 Win=29312 Len=0
4   10.50.50.103 > 10.50.50.99  104apci 72 <- U (STARTDT
        act)
5    10.50.50.99 > 10.50.50.103 104apci 72 -> U (STARTDT
         con)
6   10.50.50.103 > 10.50.50.99  TCP 66 39394 > 2404 [ACK]
        Seq=7 Ack=7 Win=29312 Len=0
7   10.50.50.103 > 10.50.50.99  104asdu 82 <- I (0,0) ASDU
        =1 C_IC_NA_1 Act      IOA=0
8    10.50.50.99 > 10.50.50.103 104asdu 82 -> I (0,1) ASDU
         =1 C_IC_NA_1 ActCon  IOA=0
9   10.50.50.103 > 10.50.50.99  TCP 66 39394 > 2404 [ACK]
        Seq=23 Ack=23 Win=29312 Len=0
10   10.50.50.99 > 10.50.50.103 104asdu 692 -> I (1,1)
         ASDU=1 C_IC_NA_1 ActCon  IOA=0 | -> I (2,1)
         ASDU=1 M_SP_NA_1 Inrogen IOA[60]=65583,... | ->
          I (3,1) ASDU=1 M_SP_NA_1 Inrogen IOA
         [44]=1120366,... | -> I (4,1) ASDU=1 M_DP_NA_1
         Inrogen IOA[30]=561192,... | -> I (5,1) ASDU=1
         M_ST_NA_1 Inrogen IOA[2]=1874019,... | -> I
         (6,1) ASDU=1 C_IC_NA_1 ActTerm IOA=0
11  10.50.50.103 > 10.50.50.99  TCP 66 39394 > 2404 [ACK]
        Seq=23 Ack=649 Win=30464 Len=0
12   10.50.50.99 > 10.50.50.103 104asdu 692 [TCP Spurious
         Retransmission] | -> I (1,1) ASDU=1 C_IC_NA_1
         ActCon  IOA=0 | -> I (2,1) ASDU=1 M_SP_NA_1
         Inrogen IOA[60]=65583,... | -> I (3,1) ASDU=1
         M_SP_NA_1 Inrogen IOA[44]=1120366,... | -> I
         (4,1) ASDU=1 M_DP_NA_1 Inrogen IOA
         [30]=561192,... | -> I (5,1) ASDU=1 M_ST_NA_1
         Inrogen IOA[2]=1874019,... | -> I (6,1) ASDU=1
         C_IC_NA_1 ActTerm IOA=0
13  10.50.50.103 > 10.50.50.99  TCP 78 [TCP Dup ACK 11#1]
        39394 > 2404 [ACK] Seq=23 Ack=649 Win=30464 Len
        =0
```

5.2.2 Experiment 4: Inject Two Step Position Responses

This experiment will investigate injecting two step position responses into the connection, rendering the HMI with an incomplete view of the system. In Experiment 3, an old response was returned containing the full amount of expected data, however in Experiment 4 a forged response containing only two step position values, i.e. fewer than expected, will be returned. The motivation for this experiment is to determine what would happen when a smaller than expected response arrives on an active TCP connection. Note that in the HTTP experiments two payloads would be rendered as one if the connection was not terminated. Listing 2 shows a packet capture of this experiment. The first packets are the TCP handshake and IEC104 start data transfer acknowledgement (1-6). Next, the GI request and acknowledgement are seen (7-9), followed by the forged packet (10-11). Note the payload is 120bits compared to the legitimate payload of 692bits (12). As with the previous experiment, the HMI confirms the received frames using the Supervisory format (14), this time with the receive sequence number of 4, since there were 4 frames sent: Opening interrogation command “ActCon”; two step positions; and Closing interrogation command “ActTerm”. Subsequently, the connection keeps using the wrong TCP sequence number which eventually causes it to time out.

Listing 2: Packet Capture of IEC104 Experiment 4.

```
 1  10.50.50.103 > 10.50.50.99  TCP 74 39440 > 2404 [SYN]
        Seq=0 Win=29200 Len=0
 2   10.50.50.99 > 10.50.50.103 TCP 74 2404 > 39440 [SYN,
         ACK] Seq=0 Ack=1 Win=8192 Len=0
 3  10.50.50.103 > 10.50.50.99  TCP 66 39440 > 2404 [ACK]
        Seq=1 Ack=1 Win=29312 Len=0
 4  10.50.50.103 > 10.50.50.99  104apci 72 <- U (STARTDT
        act)
 5   10.50.50.99 > 10.50.50.103 104apci 72 -> U (STARTDT
         con)
 6  10.50.50.103 > 10.50.50.99  TCP 66 39440 > 2404 [ACK]
        Seq=7 Ack=7 Win=29312 Len=0
 7  10.50.50.103 > 10.50.50.99  104asdu 82 <- I (0,0) ASDU
        =1 C_IC_NA_1 Act      IOA=0
 8   10.50.50.99 > 10.50.50.103 104asdu 82 -> I (0,1) ASDU
         =1 C_IC_NA_1 ActCon  IOA=0
 9  10.50.50.103 > 10.50.50.99  TCP 66 39440 > 2404 [ACK]
        Seq=23 Ack=23 Win=29312 Len=0
10   10.50.50.99 > 10.50.50.103 104asdu 120 -> I (1,1)
         ASDU=1 C_IC_NA_1 ActCon  IOA=0 | -> I (2,1)
         ASDU=1 M_ST_NA_1 Inrogen IOA[2]=1874019,... |
         -> I (3,1) ASDU=1 C_IC_NA_1 ActTerm IOA=0
11  10.50.50.103 > 10.50.50.99  TCP 66 39440 > 2404 [ACK]
        Seq=23 Ack=77 Win=29312 Len=0
12   10.50.50.99 > 10.50.50.103 TCP 692 [TCP Out-Of-Order]
          2404 > 39440 [PSH, ACK] Seq=23 Ack=23 Win
         =66560 Len=6
13  10.50.50.103 > 10.50.50.99  TCP 78 [TCP Dup ACK 11#1]
        39440 > 2404 [ACK] Seq=23 Ack=77 Win=29312 Len=0
14  10.50.50.103 > 10.50.50.99  104apci 72 <- S (4)
```

The PLC and the web server both had an artificial delay of 500ms to allow for the injected packets to reach the victim before the legitimate response. This is due to the limited scale of the laboratory testbed, and consequent low latency, this enabled the experiments to be developed without concern for the attacker to beat the legitimate packets. Nonetheless, in this testbed, when the artificial delays were completely removed, it was found that the forged response would arrive ahead of the legitimate response approximately 1 in 7 times.

6 DETECTION

Each of the four experiments were analysed offline using three of the most recent state of the art network Intrusion Detection Systems. These are Zeek 2.6.2 (formerly Bro), Snort 2.9.13-1, and Suricata 4.1.4.

Table 1: Network IDS detection results. (●) Yes (○) No (◐) Partial.

	HTTP		IEC104	
Experiment	1	2	3	4
Zeek	◐	◐	◐	◐
Snort	●	◐	○	○
Suricata	●	●	○	●

All of the engines were updated to the latest public rule sets as of May 2019, and were configured to parse offline packet capture files. Table 1 shows details of the detection results for all three engines. Successful detection is recorded as either ’Yes’, ’No’, or ’Partial’. ’Yes’ indicates an exact match on the correct packet, with the triggered rule returning an accurate description of the respective attack. ’Partial’ means, for example, a rule triggered at the correct time but the reason for the alert, or the description provided, was not accurate. ’No’ indicates no IDS alert was triggered. Zeek could be considered the least accurate of the three engines in terms of its alerts. For the HTTP experiments, Zeek alerted with ‘*FIN_advanced_last_seq*’. This alert is triggered because of the server accepting the forged packet, which brings down the connection. For the IEC104 experiments Zeek alerts with ‘*window_recision*’ on the exact injected packet. However the IDS was not triggering due to the specific MotS activity, rather it alerted because the TCP recv-window shrank by more than the amount of data being ACKed, as detailed in RFC793 Section 3.7, which is strongly discouraged. Therefore, although Zeek alerted in all four cases, the reason for each alert would not be obvious to an analyst monitoring the alerts. Zeek has built in support for detecting MotS attacks since 2015, however, it is disabled by default and does not detect the experiments. Snort alerted on the correct injected packet in the first HTTP experiment, with the message ‘*INVALID CONTENT-LENGTH OR CHUNK SIZE*” using the ’http_inspect’ module. The second HTTP experiment was also correctly detected, however, the alerts are due to the legitimate segments arriving after the connection is closed with ‘*Data sent on stream not accepting data*’ and ‘*Reset outside window*’. These are seen in both experiments 1 and 2. The only event which Snort alerts for regarding the IEC104 injections are ‘*Consecutive TCP small segments exceeding threshold*’, which is due to the server attempting to renegotiate the connection, and causing a time out. Therefore this was considered as not specifically detecting the MotS attack. Suricata had the most accurate detection of the three IDSs. With the HTTP experiments, the ACK packets with the wrong sequence numbers are detected and, as with Snort, the first HTTP injection is detected with '*SURICATA HTTP unable to match response to request*'. The most interesting alert is '*SURICATA STREAM reassembly overlap with different data*', which exactly describes the MotS attack. This is displayed for all experiments, except experiment 3, which related to injecting a replayed response. Experiment 3 did not trigger any alerts from Suricata, because the forged payload was the same as the legitimate one.

7 CONCLUSION

Further work should be performed for each of the NIDS to include detection alerts for the MotS class of attacks. While Suricata can detect this, and notify the operator with a somewhat descriptive name, it fails to articulate the issue as a malicious action. Network IDS for deployment within OT networks must have support for the industrial protocols, otherwise, detection of these kinds of attacks and others will go unnoticed. As discussed in the introduction, zero-trust networks, VPNs (to an extent), and TLS are the best mitigations of this class of attack. IEC104 is has a companion standard, IEC 62351, detailing securing end to end communication, however in internal networks, these defences are often not deployed. In part due to a lax security mindset, or the additional complexity and risk of deployment. While this class of attack is mitigated for much of the public internet via TLS, critical network operators need to be aware of the potential damage this attack may have on a system. Although deploying a mitigation approach such as zero-trust networking is the gold standard, many network operators are not in a position to do so. Threat actors, such as nation states have used this method successfully over several years, and it seems likely this approach will continue to be exploited in the future. Especially in the ICS domain, where unauthenticated local traffic such as IEC104 is still commonplace. As the experiments with Zeek, Snort and Suricata have shown, further work is required to provide these network IDS platforms with the detection rules and mechanisms capable of accurately detecting MotS being exploited by an intruder.

REFERENCES

Alexander, G., Espinoza, A. M., and Crandall, J. R. (2019). Detecting TCP/IP Connections via IPID Hash Collisions. *Proceedings on Privacy Enhancing Technologies*, 2019(4).

Conti, M., Dragoni, N., and Lesyk, V. (2016). A Survey of Man In The Middle Attacks. *IEEE Communications Surveys & Tutorials*.

DeCusatis, C., Liengtiraphan, P., Sager, A., and Pinelli, M. (2016). Implementing Zero Trust Cloud Networks with Transport Access Control and First Packet Authentication. In *2016 IEEE International Conference on Smart Cloud (SmartCloud)*.

Dragos (2017). CRASHOVERRIDE - Analysis of the Threat to Electric Grid Operations. Technical report, Dragos, Inc.

Forrester (2013). Developing a Framework to Improve Critical Infrastructure Cybersecurity. Technical report, NIST.

Gallagher, R. and Greenwald, G. (2014). How the NSA Plans to Infect 'Millions' of Computers with Malware.

Gilad, Y. and Herzberg, A. (2014). Off-Path TCP Injection Attacks. *ACM Transactions on Information and System Security*.

Gilad, Y. and Herzberg, A. (Augeuest 2012). Off-Path Attacking the Web. In *6th USENIX Workshop on Offensive Technologies*.

Green, B., Krotofil, M., and Abbasi, A. (2017). On the Significance of Process Comprehension for Conducting Targeted ICS Attacks. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, CPS '17.

Haagsma, L. (2015). Deep dive into QUANTUM INSERT.

Hjelmvik, E. (2015). Covert Man-on-the-Side Attacks.

International Electrotechnical Commission (2006). EN 60870-5-104:2006. Technical report, British Standards Institution.

Johnson, B., Caban, D., Krotofil, M., Scali, D., Brubaker, N., and Glyer, C. (2017). Attackers Deploy New ICS Attack Framework TRITON and Cause Operational Disruption to Critical Infrastructure.

Marczak, B., Weaver, N., Dalek, J., Ensafi, R., Fifield, D., McKune, S., Rey, A., Scott-Railton, J., Deibert, R., and Paxson, V. (2015). An Analysis of China's "Great Cannon". In *5th USENIX Workshop on Free and Open Communications on the Internet*. USENIX Association.

Maynard, P., McLaughlin, K., and Haberler, B. (2014). Towards Understanding Man-In-The-Middle Attacks on IEC 60870-5-104 SCADA Networks. In *ICS-CSR*.

Nakibly, G., Schcolnik, J., and Rubin, Y. (2016). Website-Targeted False Content Injection by Network Operators. In *USENIX Security Symposium*.

National Grid (2013). Response to NIST: "Developing a Framework to Improve Critical Infrastructure Cybersecurity. Technical report, National Grid.

Schneier, B. (2013). Attacking Tor: How the NSA targets users' online anonymity. *The Guardian*.

Trammell, B., Huitema, C., Schneier, B., Jennings, C., Borkmann, D., Barnes, R., and Hardie, T. (2015). Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement.

Weaver, N., Sommer, R., and Paxson, V. (2009). Detecting Forged TCP Reset Packets. In *NDSS*.