The 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
September 21-23, 2017 Bucharest, Romania

Packet-wise Compression and Forwarding of Industrial Network Captures

Gerhard Hansch, Peter Schneider, Sven Plaga
Fraunhofer Institute for Applied and Integrated Security (AISEC),
Parkring 4, 85748 Garching (near Munich), Germany
{gerhard.hansch, peter.schneider, sven.plaga}@aisec.fraunhofer.de

***Abstract* – Network traffic captures are necessary for a variety of security applications like identification of malicious patterns or training of intrusion detection systems. While monitoring of enterprise networks is common practice, it is rarely done for industrial production environments due to low bandwidth, confidential production data and sensitive legacy components.**

To address these challenges, we present methods for non-interfering recording, compression, and transmission of industrial network packet captures. Since a large portion of industrial network traffic consists of status reports that change only slightly, we replace recurring byte strings per connection to reduce the data sent, which also provides a form of concealment.

We evaluate our approach by a prototypical implementation on self-generated and publicly available industrial network captures and compare our substitution algorithm to the standard zlib algorithm as well as a combination of both methods.

***Keywords* – industrial communication, data acquisition, data compression, communication system traffic control, network security**

I. Introduction

Interconnection across multiple networks makes industrial production systems vulnerable to network threats like intrusion, exploitation, data extrusion, or malware. Common practice to detect such attacks in enterprise environments are intrusion detection systems (IDS) that monitor and validate transmitted packets but hardly meet the specific requirements of industrial production environments.

During numerous industrial research projects, we learned that plenty of excellent approaches in detecting network intrusions exist to date. Research in this area often suffers from the lack of published records from real industrial networks. Such records are necessary input for development, operation and optimization of intrusion detection as well as malicious pattern recognition systems. As probing industrial production systems is a non-trivial task, even operators of industrial facilities often have no idea how to capture, forward and store such records. Besides the fact that the recording must not interfere production at all, they have to deal with proprietary protocols, latency constraints, very long product lifetimes (>20 years), unsecured connections, and low bandwidth connections. A possible solution for field devices with bandwidth constraints can be compression of the relevant data, which has been probed by acquisition units. Unfortunately, this provides no solution for the other challenges outlined above, where high costs, technical restrictions, or regulatory approvals deny equipment modification. Furthermore, the transferred data sometimes includes confidential production information, which is often considered an asset well worth protecting.

This situation can be considered a dilemma for the development of anomaly detection algorithms and machine-learning approaches. Therefore, reliable mechanisms for data probing and pre-processing are required. We address this situation by providing the following contributions:

- A method for online deduplication and compression of captured packets, and
- an evaluation of the proposed method, focusing on the reduction ratio for different types of industrial network traffic.

The paper is organized as follows: In Section I, we motivate the need for industrial data acquisition. An overview of related work is given in Section II. Section III contains requirements and methods for network traffic recording, forwarding, and storage. At Section IV, we present our lookup-table based data deduplication method, which we evaluate in Section V. Finally, we give a conclusion in Section VI.

II. Related Work

Efficient capture and provision of network traffic from industrial production environments for the purpose of intrusion detection are demanded by numerous domains. Network intrusion detection systems are commonly used to detect anomalies or malicious patterns in the traffic of enterprise networks. A good starting point are surveys on IDS for industrial control systems (ICS) and

The presented work is a research result of the German national security reference project IUNO (http://www.iuno-projekt.de). The project is funded by the Federal Ministry of Education and Research (BMBF) and aims to provide building-blocks for security in the emerging field of Industry 4.0.

supervisory control and data acquisition (SCADA) [1], [2], as they provide a basic taxonomy, classifications as well as reviews on different concepts. While the network packets are subject of analysis by IDS, logs of normal operation are required to identify malicious patterns [3]–[5], or to train algorithms to detect anomalies [4], [6], [7]. Consequently, determining factors for the quality of the classification are the algorithm as well as the amount, quality, and composition of training data.

For bandwidth-friendly acquisition of this data, reduction and compression techniques such as the *Deflate* algorithm of the zlib library [8] can be used. However, common compression algorithms like LZ77 [9] in zlib only use a fixed window size in which data is replaced. Redundancies that occur on a lower frequency than this fixed window can capture cannot be eliminated. Additionally, as this algorithm uses a relative referencing approach instead of a direct referencing, it is required that the data is decompressed in the same order as it was compressed.

A viable source of recorded and documented network intrusion test data gathered from enterprise networks are the DARPA challenge datasets from 1998 to 2000 [10]. They contain conventional office IT traffic as well as some well-described attacks but no ICS specific communication. A more recent, ICS specific Modbus dataset, including attacks and labels, is provided by [11]. These have been generated by a simulated control system, but are still interesting for first analysis. Another dataset covers the traffic of an industrial test lab for hands-on testing captured during the 4SICS conference in Sweden [12]. Although it is quite big, this dataset does not contain any labels or information about possibly included attacks.

A domain that also requires lightweight transmission is wireless sensor networking (WSN). In contrast to this field, where bandwidth is also a challenge [13], ICS usually do not have memory, power, or computation limitations.

III. Network Traffic Capturing, Forwarding, and Storing

Network capturing is the first step in acquiring useful datasets for further processing. To maintain sufficient quality, it must be ensured that all networks of the monitored nodes are recorded. For safety reasons, the monitoring system should be passive, i.e. it should have a write restriction to the line, so that the operation could not be impaired at any time. To ensure privacy, it is necessary to treat sensitive information, like production information or credentials, separately.

Depending on the media and the connected components, additional hardware interfaces might be required for the capturing. Ethernet oriented communication systems can be tapped using common network capturing software like wireshark, tcpdump or scapy. In case of serial network protocols, a conversion to a packet-based format is required for IP-based forwarding.

On the hardware side, network wiretap devices (TAPs), port mirroring, or software modules can be used at the communication endpoints to probe the network. In the case of Ethernet, these TAPs are probing devices that enable recording of the data transmitted across a network cable. Placed in front of a monitored device, TAPs are able to capture all sent or received packets but are not able to actively interfere the monitored communication. Generally, TAPs are a viable choice to integrate a monitor feature into legacy devices and infrastructure, as no modification of existing components is required. Port mirroring is another technique, which can be applied either at the network switch (in a star-topology) that connects the monitored device, or at the port of the device itself. While wiretapping is often easier to implement, mirroring the receiving port guarantees that all packets are recognized exactly as the machine receives them. Due to the required modifications, port-mirroring is viable for modern infrastructures or new devices, but rarely an option for legacy environments. A technique, similar to port mirroring is to tap the data by a software implementation. In this case, a special library aware of every network API call could be used to collect the communication packets and send them to an aggregator. This option is viable for the outgoing traffic of a control system where only specific packets have to be captured.

To allow forwarding of all incoming data, the monitoring network should be faster than the monitored one, as the captured data is enriched by additional meta-information like timestamps or system-conditions. As the captures are by now in a packet-based state, they can be forwarded via standard TCP/IP communication, using state-of-the-art encryption libraries where needed. The deduplication described in the next chapter needs to be done before any encryption.

While it can be necessary to restore the captures to their original form for further analysis, it is also a memory friendly option to store them in the compressed form – together with the decompression scheme – in a (time series) database.

IV. Data Deduplication using Lookup Tables

One approach to reduce the size of the captured data and minimize the transmission of potentially confidential information is to replace frequent and critical content with IDs from a lookup-table. This causes the network traffic to decrease over time, because all industrial network packets should become known in the lookup-table.

When a new packet is detected at the sensor, variable sections are determined and transmitted unchanged, while the rest of the packet is merged into one byte sequence. The IDs and their assigned byte sequences are maintained in a lookup table, which can also be increased when new byte sequences are recognized. Newly assigned IDs are announced by transmitting the tuple `<ID><byte sequence>`.

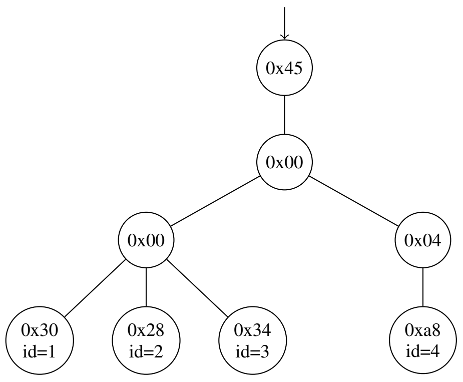


Figure 1. Prefix tree (trie) used for deduplication.

This sequence is then looked up and if it has been found, the corresponding identification number will be transmitted instead; otherwise, a new number is generated and transmitted along the byte sequence. Internally, a prefix tree (also referred to as trie) is used to store sequences of the packets. Each node in the trie may carry an ID, which is used to replace the data sequence in a message during transmission. By using a prefix tree instead of the usual hash map for lookup-tables, the memory footprint of the table can be reduced significantly while not harming the performance. The performance advantage originates from the fact that many byte sequences only differ in few positions yielding large overlaps to other sequences. An example of four shortened paths in the trie is shown in Figure 1. The paths resemble four different network packets:

- id=1: 0x45000030
- id=2: 0x45000028
- id=3: 0x45000034
- id=4: 0x450004a8

Using a plain list for the storage, the memory footprint of these four short sequences is 16 bytes. By removing the large overlap of these sequences with the trie only 8 bytes are needed to represent the same data.

Fields that are never substituted are located in the packet headers. As the previously described datasets mainly consist of IP/TCP communication, other protocols were excluded from deduplication. For the IPv4 header these are the total length, the identification number and the header checksum, while in the TCP header the sequence and acknowledgement number, the data offset, flags, and checksum fields are excluded.

In the case of encrypted data a small amount of data deduplication is still possible, as TCP header and payload are substituted separately from the IP header. Since encryption takes place on higher network layers, deduplication of IP headers redeems the overhead of transmitting IDs for the TCP header and the encrypted payload over time.

```
syntax = "proto3";

message Packet {
  bytes  static = 5;
  uint32 ipv4remainder_id = 9;
  bytes  ipv4remainder_value = 1;
  uint32 tcpremainder_id = 3;
  bytes  tcpremainder_value = 2;
}
```

Figure 2. Protobuf specification of the message format.

V. Evaluation

We evaluated our method with different datasets to test its efficiency for different packet types. Thereby, we spent specific attention on whether the assumption of industrial network data having lots of redundant and small packets holds true.

A. Test Data and Implementation

To evaluate the proposed method we implemented a demonstration script, based on Python 3, scapy [14], and Google's protobuf [15], which we applied to the following datasets:

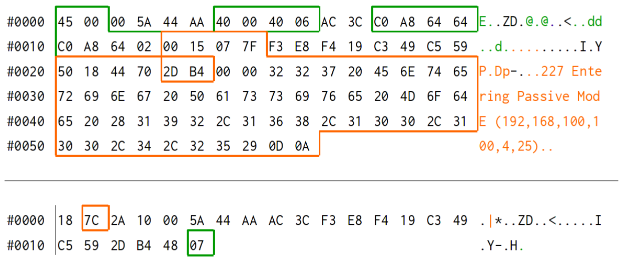
- MODBUS [11]
- ICS/SCADA [12]
- DARPA [10]
- Recorded data from a production robot.

The script reads and iteratively processes packets from a source. To determine the efficiency of the approach, several different packet captures in the pcap format were used. Each packet is analyzed by scapy to detect the protocols and get easy access to the individual header fields.

According to the message format provided in Figure 2, the fields that are always transmitted are concatenated into a single byte sequence, which is called `static`. The remaining fields are split into two separate byte sequences. The first one contains all remaining fields of the IPv4 header and is called `ipv4_remainder`, while the second sequence, the remainder of the TCP header and its payload, is called `tcp_remainder`. These are passed to separate prefix trees, which generate a new ID for the byte sequences or return the already known one if it is present. In the case that the byte sequences are new in the tree, they are transmitted in the corresponding `_value` field shown in Figure 2 along with their new ID in the `_id` field. Once the sequences start repeating, only the ID is transmitted, thus omitting the redundant parts as shown in Figure 3.

Dataset	number of packets	original size in bytes	zlib ratio	deduplication ratio	combined ratio
Modbus [11]					
run8.pcap	72186	5008499	0.96	0.88	**0.86**
channel_2d_3s.pcap	383312	17449820	1.17	**0.71**	0.90
run11.pcap	72498	4955264	0.96	**0.87**	0.87
run1_3rtu_2s.pcap	305932	15870003	1.05	**0.69**	0.81
4SICS - S7 [12]					
4SICS-GeekLounge-151020.pcap	246137	18000708	0.91	**0.36**	0.46
4SICS-GeekLounge-151021.pcap	1253100	101191303	0.85	0.55	**0.53**
4SICS-GeekLounge-151022.pcap	2274747	139975880	1.02	**0.71**	0.78
Self Recorded Robot					
lab.pcap	10120	3227398	0.39	0.95	**0.36**
Office Network					
KDD'99 Day 1 [10]	1362869	280299459	0.81	0.93	**0.76**

Table I. DATA REDUCTION EFFICIENCY. THE MOST EFFICIENT APPROACH FOR EACH DATASET IS HIGHLIGHTED BY BOLD PRINTING.



```
#0000  45 00 00 5A 44 AA 40 00 40 06 AC 3C C0 A8 64 64  E..ZD.@.@..<..dd
#0010  C0 A8 64 02 00 15 07 7F F3 E8 F4 19 C3 49 C5 59  ..d.........I.Y
#0020  50 18 44 70 2D B4 00 00 32 32 37 20 45 6E 74 65  P.Dp-...227 Ente
#0030  72 69 6E 67 20 50 61 73 73 69 76 65 20 4D 6F 64  ring Passive Mod
#0040  65 20 28 31 39 32 2C 31 36 38 2C 31 30 30 2C 31  E (192,168,100,1
#0050  30 30 2C 34 2C 32 35 29 0D 0A                    00,4,25)..

#0000  18 7C 2A 10 00 5A 44 AA AC 3C F3 E8 F4 19 C3 49  .|*..ZD..<.....I
#0010  C5 59 2D B4 48 07                                .Y-.H.
```

Figure 3. Example of the deduplication effect on a FTP packet captured in an industrial context. The upper part resembles the original packet while the bottom part displays the deduplicated packet. The colors highlight the substituted parts of the packets.

B. Compression Efficiency

For the test bed, we used our implementation to process and forward the described network captures.

Table I shows a performance comparison of data deduplication versus the well-known zlib compression algorithm [8]. The table indicates the remaining size and remaining size ratio for three different approaches, i.e. the smaller the numbers the more effective the approach.

The first approach called *zlib* uses packet-wise zlib compression. As it can be seen from the indicated ratios for the different datasets, the efficiency is varying severely and can even lead to an increase of data in some cases. The self-recorded dataset from a robot running in a laboratory yielded a very low compressed size. As zlib compresses bytes inside a fixed search window, local redundancies can be removed more efficiently. However, for the remaining datasets the packet-wise zlib compression reduced the data size only in a few cases while in some the resulting packets even increased. This is caused by the fact that especially the industrial network packets like Modbus and S7 communication tend to be quite short. For these short packets, an effective compression using stateless zlib is simply not possible.

The results of the proposed deduplication method using a lookup-table is shown in the next two columns of Table I. Except for two cases, our method achieves smaller packets and therefore a higher compression ratio than the previously described zlib approach. It is worth noting that although our method still involves some overhead and theoretically can lead to an increase in the data volume, in practice, none of our test datasets shows such an increase.

Finally, chaining both approaches was tested by first deduplicating the data as described foremost and then using the zlib compression on top of the remaining packets. As can be seen by the results, this combined approach is always more efficient than just using plain zlib alone. However, in many cases using zlib on top of the deduplicated packets even increases the size again. It is assumed that this is caused by the fact that for very small packets the zlib compression does not work as good as expected.

C. Private Encoding

Another benefit of the deduplication approach is that the transmitted data gets indirectly encoded. As a result, transmitted values and commands are concealed. However, more important, a possible eavesdropping attacker could not identify what the sender and recipient of the packets have been originally as the IP header deduplication covers this information and therefore omits them from the transmission in over 99% of the packets. An example what a possible attacker might see is given in the lower part of Figure 3. The marked parts resemble the original data and their substitution by an ID. Note that most of the valuable information, i.e. sender and receiver IP and ports and the actual payload, cannot directly be inferred from the deduplicated packet. Using this online deduplication approach, an attacker is not directly able to infer the meaning of the transmitted IDs. As the IDs are only once transmitted with their corresponding byte sequence, an attacker is only able to reassemble the original packets if he already captured this packet during the learning phase. In practice, this leads to a system that is vulnerable during start up and becomes highly specialized to the needs of the sender and receiver over time while building up a customized protocol of IDs, which are private only to the sending and receiving node.

D. Discussion

Since a lot of the typical industrial network traffic consists of small status report messages with a low entropy, a high amount of redundancy deduplication in the captures that should be forwarded is possible. As highlighted in Table I, we achieve better overall compression by deduplication or combination than by stateless zlib compression alone. The reason for this is that in contrast to office network protocols, industrial networking protocols use small but very frequent packets. Thereby, the limited set of header field combinations, like source, destination and parameters, but also frequent payload are efficiently substitutable parts. For high entropy data like encrypted network traffic, or the fast changing position reports of the robot in our laboratory, deduplication of packet payload is hardly effective. Even in such cases, the reliable savings through deduplication of packet headers prevents an overhead in the long term.

While the usage of offline compression algorithms can achieve a much higher reduction of data size, these require knowledge of all data to be processed, i.e. they rely on processing the data in blocks. Doing an online data deduplication only based on prior packets allows for removing redundancy between packets while future data is still unknown.

In cases where adequate protection is required due to confidentiality of the captured data, the concealment substitution of process specific parameter to unspecific identifiers is often not sufficient, as an attacker that monitors the connection from the beginning is able to also create the corresponding lookup-table. Instead, conventional measures like TLS encryption are necessary and applicable, as they do not interfere each other.

VI. Conclusions

In this paper, we presented efficient methods to capture traces from industrial production networks, and transfer them via bandwidth-constrained connections. We discussed different mirroring and wiretapping strategies to capture such packets from industrial networks, considering the sensitivity of components and data. Further, we presented and evaluated a method to reduce the size of network captures for a loss-free transfer via bandwidth-limited networks and storage savings.

While our method of byte-sequence substitution using a prefix tree reliably reduces the transmitted packet size and conceals the substituted content, it can further benefit from a combination with the zlib compression for remaining redundancies. In this way, we were able to reduce the size of the packet-wise compressed and forwarded industrial network captures from our test data by up to 64%. As the efficiency increases over time, we expect an even better compression rate for long-term operation.

References

[1] B. Zhu and S. Sastry, "SCADA-specific Intrusion Detection/Prevention Systems: A Survey and Taxonomy," in *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*, 2010.

[2] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–29, 2014.

[3] R. Udd, M. Asplund, S. Nadjm-Tehrani, M. Kazemtabrizi, and M. Ekstedt, "Exploiting Bro for Intrusion Detection in a SCADA System," 2016. [Online]. Available: http://dl.acm.org/ft_gateway.cfm?id=2899028&type=pdf

[4] M. Mantere, M. Sailio, and S. Noponen, "Network Traffic Features for Anomaly Detection in Specific Industrial Control System Network," *Future Internet*, vol. 5, no. 4, pp. 460–473, 2013.

[5] Y. Yang, K. McLaughlin, S. Sezer, Y. B. Yuan, and W. Huang, "Stateful intrusion detection for IEC 60870-5-104 SCADA security," in *2014 IEEE Power & Energy Society General Meeting*, 2014, pp. 1–5.

[6] L. A. Maglaras and J. Jiang, "Intrusion detection in SCADA systems using machine learning techniques," in *2014 Science and Information Conference (SAI)*, 2014, pp. 626–631.

[7] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li, "AI^2: Training a Big Data Machine to Defend," in *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*. IEEE, 2016, pp. 49–54.

[8] J.-l. Gailly and M. Adler, "Zlib compression library," 2004.

[9] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337–343, 1977.

[10] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Computer networks*, vol. 34, no. 4, pp. 579–595, 2000.

[11] A. Lemay and J. M. Fernandez, "Providing scada network data sets for intrusion detection research," in *9th Workshop on Cyber Security Experimentation and Test (CSET 16)*. Austin, TX: USENIX Association, 2016. [Online]. Available: https://www.usenix.org/conference/cset16/workshop-program/presentation/lemay

[12] NETRESEC, "Capture files from 4sics geek lounge." [Online]. Available: https://www.netresec.com/?page=PCAP4SICS

[13] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, vol. 2. IEEE, 2005, pp. 8–13.

[14] P. Biondi, "Network packet manipulation with scapy," 2007.

[15] G. Inc., "Protocol buffers - google's data interchange format," 2008. [Online]. Available: https://github.com/google/protobuf