Monitoring IP-ID Behavior for Spoofed IPv4 Traffic Detection

Sabrina Kaniewski*, Lukas Bechtel*, Michael Menth†, Tobias Heer*
* *Esslingen University of Applied Sciences, Germany,*
† *University of Tübingen, Germany,*
Email: {sabrina.kaniewski, lukas.bechtel, tobias.heer}@hs-esslingen.de, menth@uni-tuebingen.de

***Abstract*—Monitoring network traffic and devices is crucial for ensuring secure network operation, particularly in industrial networks, which operate for a long time and contain a mix of devices, i.e., devices with state-of-the-art security and legacy devices whose security features are often insufficient. Such legacy devices require additional compensating security measures for secure operation, especially due to increasing connectivity, exposing legacy devices to new security threats, such as spoofing. For this purpose, we propose an anomaly detection mechanism based on the IPv4 Identifier (IP-ID) field that provides hints for spoofed IPv4 devices. The IP-ID field is a 16-bit value in the IPv4 header. Receiving hosts use it to identify and reassemble parts of a fragmented IP packet. Multiple variants for assigning IP-IDs exist. For example, many legacy devices use a global counter with a fixed increment between subsequent packets. The proposed mechanism is based on the observation that IP-IDs in spoofed traffic may not comply with the previously observed IP-ID assignment behavior of the device. Such deviations can be detected as an anomaly and taken as a hint for spoofing. We provide an overview of existing assignment behaviors and present a classification algorithm using captured traffic. Likewise, we present a simple monitoring algorithm for detecting deviations in a host's classified IP-ID assignment behavior. We address various challenges, such as incomplete captures and unsuited deployment positions, and how to deal with them. We evaluate the feasibility of the algorithms on real-world traces. Further, we present a proof-of-concept implementation that detects various spoofing attacks in a testbed. The proposed mechanism improves security in industrial and related brownfield networks by passively detecting spoofed devices.**

***Index Terms*—Spoofing, IP-ID, Legacy devices, Compensating security measures, Brownfield networks**

I. Introduction

Monitoring network traffic and devices is essential in ensuring secure network operations. Implementing monitoring mechanisms is particularly important for brownfield networks, which contain both state-of-the-art and legacy devices, the latter often having insufficient security features. Brownfield networks are common for industries such as industrial automation or cyber-physical systems, which contain specialized devices with long lifespans. Legacy devices in these domains, e.g., networked devices in the Field and Control Layers, such as Programmable Logic Controllers (PLCs), lack the capabilities to support enhanced security measures, e.g., authentication. With increasing connectivity brought by concepts such as Industry 4.0 and the Internet of Things, exposing legacy devices to new security threats, alternative compensating security measures are required to secure operations.

In this paper, we propose an anomaly detection mechanism for spoofed traffic based on the 16-bit IP-ID value in the IPv4 header. Receiving hosts use the IP-ID to identify and reassemble fragmented IP packets. Hosts may assign the IP-ID using a variety of behaviors. For example, many legacy devices use a counter with a fixed increment between subsequent packets. The proposed mechanism is based on the observation that IP-IDs in spoofed traffic might differ from the previously observed IP-ID assignment behavior of the legacy device. These deviations can be identified as anomalies, hinting at spoofing. The mechanism applies to IPv4 traffic and is compatible with existing infrastructure and devices, as it passively monitors traffic. IPv6-capable devices are typically not legacy devices and already support security mechanisms themselves. The mechanism provides a compensating security measure to apply in brownfield networks, strengthening the secure operation of legacy devices.

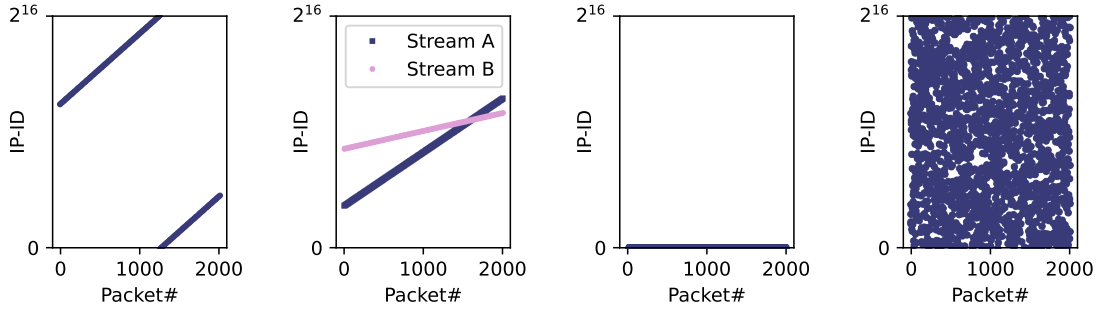
The proposed mechanism consists of two algorithms: classification and monitoring. The classification algorithm classifies the IP-ID assignment behavior of each device in the network based on traffic captures. The monitoring algorithm continuously checks in an online manner if the following IP-IDs match the previously observed assignment behavior and detects any deviations. This mechanism makes it very costly or impossible for an attacker to integrate an attack device in the network that complies with the expected IP-ID behavior.

We evaluate the proposed mechanism in two parts. First, we verify its applicability by applying the classification algorithm to network traces from various industrial testbeds. This part visualizes the different IP-ID assignment behaviors we can classify and monitor. Second, we present a proof-of-concept (PoC) that detects different spoofing attacks. We discuss arising challenges, such as incomplete captures or unsuited deployment positions, and how to deal with them.

For transparency, we open-source the code for the classification algorithm, the monitoring algorithm, and the PoC. We also detail the results of the application test cases in the evaluation: https://github.com/hs-esslingen-it-security/hses-ipid-monitoring.

This work has been funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) under support code 16KN084434 – Collaborative Project ESP and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 528745080 - FIP 68. The authors alone are responsible for the content of the paper.

Accepted for publication in Proceedings of *29th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Padova, Italy, September 10-13, 2024



(a) Global (b) Per-stream (c) Constant (d) Random

Figure 1: Different IP-ID assignment behaviors.

The remainder of the paper is structured as follows. Section II provides information on different IP-ID assignment behaviors and their suitability for monitoring. In Section III, we present related work in the context of network security and monitoring. We introduce the classification and monitoring algorithm and present an exemplary deployment in Section IV. In Section V, we evaluate the feasibility of the approach and present the PoC. We conclude this work in Section VI.

II. Background: IP-ID

In this section, we provide background information on the IP-ID specifications. We introduce common IP-ID assignment behaviors and discuss their suitability for monitoring. Further, we discuss the distributions of deployed behaviors to address the applicability of the proposed monitoring approach.

A. IP-ID Specifications and Assignment Classes

Every IP packet contains a 16-bit IP-ID in the IPv4 header. If an IP packet is fragmented, each fragment carries the same IP-ID, allowing for reassembly by the receiving host. Specifications of the IP-ID assignment [1, 2] state that each packet requires an IP-ID that is unique for the tuple of source address, destination address, and protocol for the maximum lifetime of a packet in the network. Specifically, this specification has to hold if fragmentation is possible.

The IP-ID assignment depends on the implementation of the network layer in the operating system of a host. RFC 4413 [3] and RFC 5225 [4] describe four common assignment classes, visualized in Figures 1a–1d and detailed in the following.

(a) **Global**. A global counter implementation assigns the IP-ID for all IP packets sent by the host. Typically, the counter is incremented by 1 (*IP-ID increment*) after each packet and restarts at 0 after exceeding the value $2^{16} - 1$ (*wrap-around*).

(b) **Per-stream**. A per-stream counter implementation is similar to a global counter implementation. It uses multiple counters to assign IP-IDs for different IP streams identified by the tuple of protocol, source, and destination. The counter is incremented after each packet in the stream.

(c) **Constant**. In the case of a constant implementation, the IP-ID value is set to a constant value, typically zero.

(d) **Random**. The random implementation uses a pseudo-random number generator to assign IP-IDs.

Current implementations show variations to these classes, e.g., a global counter where the two bytes of the IP-ID are byte-swapped [4], i.e., a little-endian byte order is used instead of a big-endian. Other behavior may result from non-standard increments and wrap-arounds or wrong configuration [5].

The proposed IP-ID-based mechanism requires predictable IP-ID values to distinguish expected from anomalous behavior. In the case of a random assignment, we cannot classify behavior as expected or anomalous. We can observe the frequency of IP-IDs in specific value ranges and conclude whether they behave randomly. For the monitoring of legacy devices, however, we do not consider the random assignment. In the case of a constant IP-ID, we can only detect anomalous packets if the IP-ID differs from the expected constant. For the global and per-stream assignment, increments in the IP-IDs are predictable. Hence, we can determine whether consecutive IP-IDs follow the assignment behavior of the device. We can further detect anomalous IP-IDs resulting from injected or replayed packets that lead to unexpected or duplicate IP-IDs, respectively. Therefore, variations of counter assignments are best-suited for monitoring.

B. IP-ID Assignment Distributions

Regarding the applicability of IP-ID monitoring, we need to consider the distributions of IP-ID counter implementations. In the following, we present studies that analyzed the distributions of deployed IP-ID implementations.

Salutari et al. [5] classified the IP-ID behavior of one host per /24 subnet in the public IPv4 space using ICMP probes. They reported a proportion of 18% global and 34% per-stream behavior in 2017. In 2022, Feng et al. [6] conducted an Internet-wide study of the distributions using ZMap. They reported that 68% of hosts implement an IP-ID counter with a linear increment. They further analyzed the IP-ID assignment algorithms in various TCP/IP stack implementations. Windows XP and Windows 7, e.g., implement a global, and Linux Kernel 3 versions implement a per-stream counter.

While these studies provide insights into the relative popularity of IP-ID counter implementations, these studies are limited to implementations in devices in the public IPv4 space. In particular, legacy devices placed, e.g., in isolated industrial

networks, are not considered. Legacy devices are more likely to use IP-ID counters, such as the global counter, which was the most common implementation in 2006 [3]. Thus, while the proposed IP-ID-based mechanism is limited to deterministic IP-ID counter implementations, these implementations are common in industrial brownfield systems.

III. Related Work

Researchers utilize the predictability of IP-ID increments of counter implementations for various purposes. We review related work that uses the IP-ID in the context of network security and monitoring.

A. Fingerprinting

The IP-ID assignment behavior is used in TCP/IP fingerprinting. TCP/IP fingerprinting uses TCP/IP header information unique for systems, such as the initial TTL value or TCP window size, to form signatures used to fingerprint them. This fingerprinting can either be active or passive. Active techniques send TCP/IP probes and analyze header fields in the response. For example, Nmap uses TCP and ICMP probes to determine the IP-ID assignment behavior of a device [7]. In contrast, passive fingerprinting approaches observe network traffic and use heuristics to identify systems or devices. Al Ghazo and Kumar [8] use the IP-ID increment of consecutive packets together with the initial TTL and MAC address to passively fingerprint industrial control devices.

TCP/IP fingerprinting tools, such as Nmap and the work by Al Ghazo and Kumar, are useful for determining the IP-ID assignment behavior of a device. However, while fingerprinting tools analyze IP-IDs to classify the assignment behavior, the goal in this work is to detect when a device behaves differently, e.g., due to being spoofed, by closely observing IP-ID sequence behavior. We follow a passive approach to not disturb processes through additional traffic, which is important for the application in critical infrastructures.

B. End-Points and End-Point Behavior

Other related work uses the IP-ID to observe end-points and end-point behavior. Bellovin [9] proposes an IP-ID-based algorithm for counting the number of hosts behind a NAT. It observes IP-IDs and groups them into sequences based on whether the IP-IDs match within a predefined threshold and time interval. Using the grouped sequences, it estimates the number of hosts behind the NAT. This algorithm works for the global counter implementation. However, for per-stream counter implementations, it overestimates the number of hosts. Further, random and constant assignments are not considered. Mongkolluksamee et al. [10] enhance this algorithm by associating IP-ID sequences with TCP sequence numbers and TCP source ports, which are initialized differently by operating systems at the beginning of a connection, e.g., at random or linearly ascending. By associating the different sequences, it is possible to count hosts implementing per-stream counters. Chen et al. [11] present an approach to measure the amount of internal and external network traffic sent by a server using a global IP-ID assignment. They observe gaps in IP-ID sequences over time at a gateway. Based on the size of the gaps, they infer how much traffic the server sent.

In contrast to these works, we observe changes in IP-ID behavior. Specifically, we check for the correctness and continuity of IP-ID sequences to detect security-relevant changes in device behavior. We can apply the mechanism to various IP-ID counter implementations since we classify their behavior with the specific increment and wrap-around.

C. Spoofing Defense

Templeton and Levitt [12] propose an IP-ID-based probing technique to detect spoofed packets. After receiving a packet, a probe is sent to the source. If the IP-ID in the response is greater than and close to the initially received IP-ID, the packet was likely sent by the claimed source. Rather than verifying IP-IDs of individual packets, we monitor IP-ID sequences. This way, we are not limited to the implementation of a global counter in a device to detect spoofing attempts and can detect unexpected as well as duplicate IP-IDs. Due to the passive approach, we can also apply the mechanism to network traces at a later time to identify attack devices in retrospect.

IV. IP-ID-based Anomaly Detection: Algorithms and Deployment

We monitor the traffic of legacy devices for their expected IP-ID assignment behavior to detect anomalous behavior caused by spoofing. The proposed mechanism operates on traffic data captured by one or multiple monitoring instances, e.g., a monitoring gateway. In the first classification phase, we use captured traffic to classify the IP-ID assignment behavior of each device in the traces. During monitoring, we continuously check whether the IP-IDs in sent packets follow the classified behavior and report deviations. Figure 2 provides insights into the deployment of the monitoring mechanism. In the following, we present the classification and monitoring algorithm in detail and address challenges regarding the deployment.

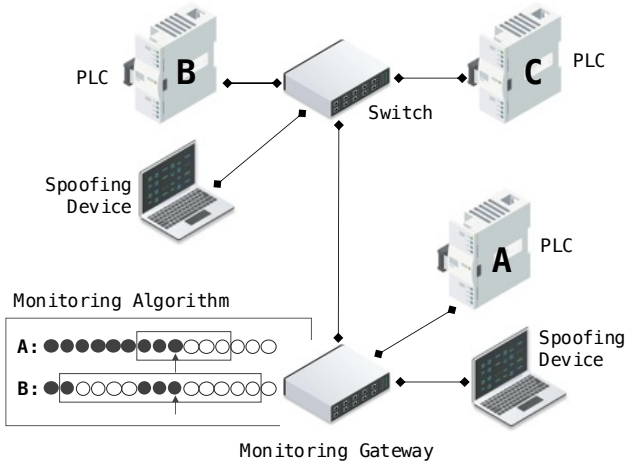
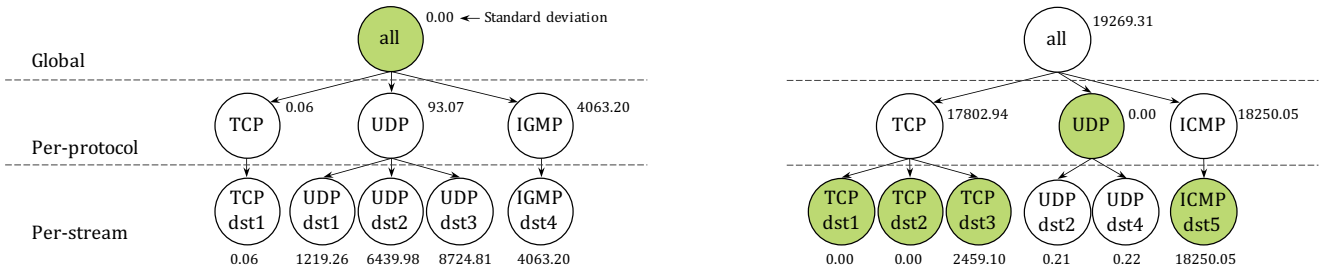


Figure 2: Example monitoring deployment.



(a) MODBUS TCP SCADA #1 HMI – Global behavior

(b) SWaT P2 PCN – Mixed behavior (per-protocol and per-stream)

Figure 3: Classification of IP-ID behavior. Standard deviations were calculated over IP-ID sequences of differences.

A. Classification Algorithm

We classify the IP-ID assignment behavior of each device in the traces in four steps, using traces captured over time.

(1) We split the traces per source IP. We then extract the source and destination IP address, protocol, and IP-ID value of each packet in the traces and store them in 4-tuples.

(2) For each source IP, we prepare the identification of the underlying IP-ID assignment behavior by aggregating the respective tuples into different groups based on common assignment behaviors: a group holding all tuples (*Global*), a group holding the tuples for each protocol (*Per-protocol*), and a group holding the tuples for each stream (*Per-stream*).

(3) For each resulting IP-ID sequence in the different groups, we compute the differences of consecutive IP-IDs, i.e., $\text{IP-ID}_{i+1} - \text{IP-ID}_i$. For the examplary sequence $[1\ 2\ 3\ 4\ 7\ 8\ 9]$, the corresponding sequence of differences is $[1\ 1\ 1\ 3\ 1\ 1]$.

(4) In each resulting group, we calculate statistics over the corresponding sequence of differences to check how well different assignment behaviors fit. If a device implements a global counter, the standard deviation of the sequence of differences in the *Global* group will be zero as all IP-ID increments are identical. If a device implements multiple per-protocol or per-stream counters, the standard deviation in individual *Per-protocol* or *Per-stream* groups will be zero. In the case of a constant assignment, i.e., increments of zero, the mean in the group will also be zero. In the case of random behavior, the statistics will show no such deterministic behavior, cf. features in [5]. To classify the underlying assignment behavior and identify the underlying counter(s), we compare the standard deviations of the individual groups. We identify the assignment behavior that fits best based on the group(s) with the lowest standard deviation. For this comparison, we proceed in a hierarchical order, from the *Global* group to the more granular *Per-protocol* and *Per-stream* groups. Figure 3 visualizes this process. Whenever we find a smaller (or equal) standard deviation in a more granular group, the device assigns its IP-IDs more granularly. We then select the more granular behavior. For example, in Figure 3a, we classify a global behavior as the standard deviation is the smallest for the *Global* group. Similarly, in Figure 3b, we classify a per-stream behavior for TCP and ICMP traffic and a per-protocol behavior for UDP traffic. For some groups that are part of the best-fitting behavior, the corresponding standard deviations may not be zero but considerably higher. For example, for the behavior in Figure 3b, TCP-dst3 shows no clear per-stream assignment compared to the other TCP streams. However, as we identify a per-stream behavior for the other TCP streams, all TCP streams must follow this behavior. Such a high standard deviation likely results from irregularities in the IP-ID sequence, caused, e.g., by traffic reordering or gaps. In addition, if the sequence includes a wrap-around, the difference of, e.g., -65535, impacts the standard deviation negatively even though it is expected behavior. To address these irregularities, we sort all observed differences of IP-IDs and trim this list on both sides by 0.5%. By doing so, we reduce the influence of such effects, allowing us to match the underlying assignment behavior more accurately while still using 99% of the input for classification. For a standard deviation greater than $(2^{16}-1)/\sqrt{12}$, we classify the behavior as undefined, cf. features for random behavior in [5].

For each source IP, we report the classified behavior, i.e., the assignment class and specific configuration, including the increment or constant and wrap-around value used. For the configuration, we use the mode and smallest observed difference smaller than zero (not considering the wrap-around value), respectively. We further include other meta-information, such as the largest observed gap.

B. Monitoring Algorithm

Using the monitoring algorithm, we aim to detect deviations in a host's classified IP-ID assignment behavior. While some irregularities in IP-ID sequences are normal, such as those caused by traffic reordering, gaps caused by traffic not passing through the capturing instance or packet loss, they must be identified during the classification phase and considered during monitoring. We use a monitoring window to assess the validity of observed IP-IDs in the expected sequence. This approach enables us to detect unexpected IP-IDs and account for packet reordering or loss. The algorithm operates on network traces in an online manner.

We implement the monitoring window using a bit vector with two control parameters: window size and history factor. We set the window size w using the largest gap observed during classification (gap_{max}), i.e., the largest difference in the IP-ID sequence (not considering wrap-around). In particular, we set the window size to a multiple of gap_{max}, with a

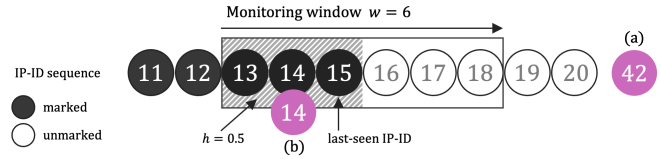


Figure 4: Anomalies detected by the monitoring algorithm: (a) unexpected IP-ID, (b) duplicate IP-ID.

default of $w = gap_{max} \cdot 2$. The resulting window ensures that traffic anomalies that may occur during the runtime do not result in false positives or interfere with the progress of the monitoring window. The history factor h specifies the percentage of already seen IP-IDs in the window, set by the user, e.g., $h = 0.5$. Figure 4 visualizes the concept of window size and history factor.

For each legacy device, we first initialize the respective monitoring window(s). The window bounds are set using the first IP-ID received during monitoring and the window size obtained during classification. When a following IP-ID falls within these bounds, it is accepted and the corresponding position marked. We then shift the window forward until the position aligns with the history factor, see Figure 4.

If a received IP-ID does not fall within the window or results in a duplicate not caused by fragmentation, we report an anomaly. Figure 4 illustrates these anomalies, i.e., (a) an unexpected IP-ID and (b) a duplicate IP-ID. An unexpected IP-ID that falls outside the window may be caused by spoofed packets. A duplicate IP-ID may result from replayed packets with the respective position in the window already marked.

C. Monitoring Deployment

The monitoring algorithm can be deployed at different positions in the network. When deploying it for multiple legacy devices, the distance of the devices to the monitoring instance influences the amount of traffic captured and, thus, the monitoring accuracy. For example, in the deployment visualized in Figure 2, the monitoring gateway does not capture packets sent between PLC B and C. We refer to such traffic not visible to the gateway as *internal traffic*. Depending on the position of the monitored device, different challenges arise.

When monitoring PLC A, the monitored device is directly connected to the monitoring instance. As all traffic of PLC A passes through the gateway, we can closely monitor IP-ID sequences for the expected behavior. Thus, we can detect anomalous IP-IDs with high accuracy using a small window.

When monitoring PLC B, we may have to deal with gaps caused by internal traffic if PLC B implements a global counter and communicates with PLC C. As a result, we experience a higher standard deviation during classification, resulting in a wider monitoring window. A too-wide window, e.g., with a size of several hundred, is not suited for closely monitoring IP-ID sequences. However, it still provides information on whether an attack device is active as IP-IDs in spoofed packets likely deviate from the expected behavior.

V. Evaluation

In this section, we evaluate the proposed mechanism. First, we evaluate its applicability by analyzing the IP-ID assignment behavior of different industrial devices, using network traces from several external physical testbeds. We apply the classification and monitoring algorithm to the traces and discuss the results. Second, we present the PoC implementation, evaluating the accuracy of the monitoring algorithm to detect spoofing attacks. Finally, we discuss the challenges of the proposed mechanism and how to address them.

A. Application Test Cases

To validate the applicability of the proposed mechanism, we analyze network traces from several physical industrial testbeds published for security research, see Table I. These testbeds cover different industrial application domains and devices in the scope of considered legacy devices, including PLCs, Remote Terminal Units (RTUs), Intelligent Electronic Devices (IEDs), and Human-Machine Interfaces (HMIs). We investigate how IP-IDs are set by these devices, whether the behaviors follow specifications or there are any deviations, and if we can successfully monitor them. We apply the classification and monitoring algorithm to traces from each testbed, using one part for classification and another for monitoring, and discuss the results.

1) Classification: We classify the IP-ID assignment behavior of 30 devices present in the traces, including PLCs, RTUs, IEDs, and HMIs. Note that some traces contain duplicate packets, possibly due to merging captures from multiple positions in the network. We filtered out duplicate packets before further analysis to improve accuracy.

We identified 19 (63.33%) global and 11 (36.66%) per-stream counter implementations. Therefore, the assignment behaviors of all devices are suited for monitoring. We manually reviewed the classification results to confirm that all classified behaviors match the underlying ones. The algorithm only misclassified the specific configuration of four devices, detecting an incorrect increment or wrap-around value due to noisy or insufficient data.

Table I: Overview of external testbed datasets used in the evaluation.

Testbed	Domain	Protocol	#Packets used for Classification	#Packets used for Monitoring	Open-Source Dataset
QUT_DNP3 [13]	Power transmission	DNP3	129.652	132.033	https://github.com/qut-infosec/2017QUT_DNP3
QUT_S7Comm [13]	Mining refinery	S7Comm	1.782.486	396.523	https://github.com/qut-infosec/2017QUT_S7comm
MODBUS TCP SCADA #1 [14]	Liquid pump	Modbus	394.905	66.299	https://github.com/tjcruz-dei/ICS_PCAPS
EPIC [15]	Smart grid	IEC 61850	8.971.319	8.414.915	–
SWaT [16]	Water treatment	EtherNet/IP	6.933.125	2.108.317	–

From the analyzed IP-ID behaviors, one device showed perfect behavior without any variations resulting from, e.g., packet reordering, gaps, or unusual configuration. However, slight variations in IP-ID sequences are usual, particularly if the monitoring instance is not placed directly in front of the legacy device. Figures 5a–5d illustrate variations and anomalies encountered in the analyzed behaviors. We observed (a) early wrap-arounds, (b) a mix of IP-ID class implementations, (c) an unexpected reset, and (d) gaps in IP-ID sequences. In the following, we discuss these anomalies and how we address them during classification and monitoring.

IP-ID counters typically wrap around after reaching $2^{16}-1$. Figure 5a shows an early wrap-around after reaching the value $2^{15}-1$ in two devices, possibly resulting from a specific configuration in the deployed Windows operating system [13, 14]. The classification algorithm determines any wrap-around value, given at least one wrap-around in the traces.

Implementations using multiple IP-ID counters show a mixed class behavior. For example, the PLCs in the EPIC testbed use one counter each for TCP and ICMP streams and a constant assignment for UDP traffic, see Figure 5b. The PLCs in the SWaT testbed use only one UDP counter but assign the IP-IDs for TCP connections considering the different streams. Further, the UDP counters use an increment of 1 while the TCP counters use an increment of 256, i.e., the counters are byte-swapped. As we check for the different behaviors during classification using grouped IP-ID sequences, we can identify such mixed implementations, see Figure 3b.

We observe a reset in one IP-ID sequence from 14902 to 1, see Figure 5c. As the authors [14] address no anomalies, we cannot be sure what caused this reset. A possible cause could be a restart. To ensure that restarts during classification do not negatively affect the window size, a maximum window size should be agreed on. To handle restarts during monitoring, we can use a backup window. For example, if we detect an anomaly and the counter restarts at 0, we can use the backup window to monitor subsequent IP-IDs. If the IP-IDs continue as expected, we can raise a dedicated warning reporting a restart and continue monitoring with the backup window.

Several IP-ID sequences show gaps, see Figure 5d. To verify that the gaps do not result from an unusual behavior, we investigated the corresponding TCP sequence numbers for continuity. The continuous sequence numbers indicate no missing TCP segments but rather that the gaps result from, e.g., internal traffic. Frequent gaps in the IP-ID sequences lead to a higher standard deviation during classification. Still, by following the presented process to identify the IP-ID behavior, we correctly classify the behavior even in the presence of gaps.

2) Monitoring: We applied the monitoring algorithm to the traces using the (corrected) configurations obtained after classification. The algorithm ran successfully for the QUT_DNP3, QUT_S7Comm, and MODBUS TCP SCADA traces. For the EPIC traces, the algorithm reported false positives in two cases where the devices experienced much larger gaps during monitoring than during classification. The monitoring of SWaT traces was unsuccessful since the algorithm reported numerous

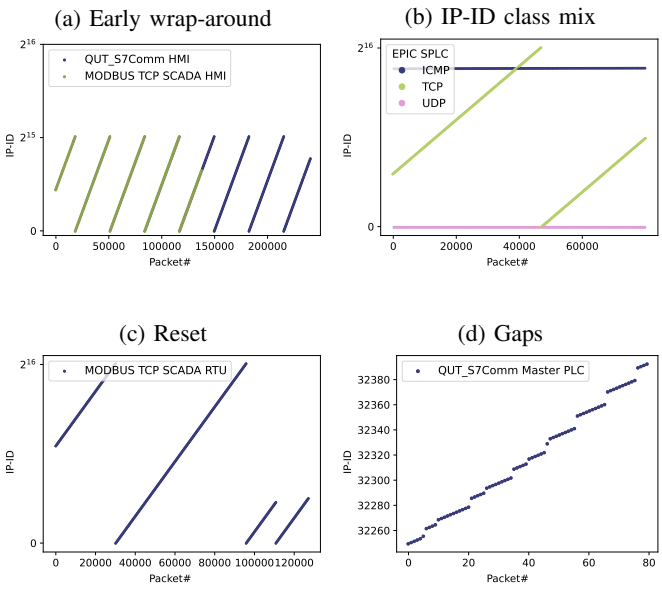


Figure 5: IP-ID sequence and assignment anomalies observed in the different testbed datasets.

unexpected IP-IDs. The SWaT traces contain duplicate packets and redundant information, most likely due to merging from multiple capturing points. While the classification was successful, such noisy traces are not suited for monitoring.

B. Proof-of-Concept (PoC)

Within the PoC, we discuss the accuracy of the monitoring algorithm to detect spoofing attacks. First, we address the underlying attack model. Second, we present the PoC setup and the different test cases considered.

1) Attack Model: We consider an attacker with physical access to the network able to place attack devices that spoof network addresses. The attack devices can be placed as a replacement for an authorized legacy device or as an additional device using, e.g., an unsecured switch port. The attack devices can send packets into the network.

2) Setup and Monitoring Test Cases: We set up a virtual testbed to implement the deployment shown in Figure 2. PLC A and PLC B communicate with each other in an intermittent behavior, exchanging TCP traffic to simulate industrial communication. Additionally, PLC A sporadically sends UDP traffic to simulate answers to, e.g., SNMP requests. PLC C also sporadically communicates with PLC B over TCP. The hosts representing the PLCs can mimic a global and a per-stream IP-ID behavior, allowing us to assess the detection of spoofing attacks for different assignment behaviors.

We discuss the two test cases of monitoring PLC A and PLC B. For each mimicked IP-ID assignment behavior, i.e., global and per-stream, we captured 200.000 packets each for both test cases and for classification and monitoring. We performed the spoofing attacks during both phases, whereby we first captured a classification phase without spoofing to analyze the impact on classification results. Spoofed TCP

packets were assigned a fixed IP-ID value of 1 to verify the detection accuracy during monitoring. In the following, we discuss the results and address emerging vulnerabilities.

Monitoring Test Case 1 - PLC A. The classification algorithm correctly classifies the global IP-ID assignment behavior of PLC A during classification without active spoofing, observing a gap_{max} of 1 and a standard deviation of 0.0 for the global group. With active spoofing, deviations caused by the attack device result in a gap_{max} of 65507 and a considerably higher standard deviation, distorting the result to such an extent that the algorithm classifies a per-stream behavior for the UDP stream. However, as these deviations are recognizable in the report, the attack device can be detected by manual review. We apply the monitoring algorithm to the monitoring traces using the configuration obtained during classification without active spoofing and detect all spoofed packets. As the bounds of the monitoring window are initialized using the first IP-ID received, they may be wrongly set if this first packet is spoofed. However, following IP-IDs of PLC A then result in subsequent unexpected IP-IDs, hinting at spoofing.

Mimicking a per-stream assignment, the classification algorithm correctly classifies the underlying per-stream assignment behavior of PLC A, with both streams showing a gap_{max} of 1 and a standard deviation of 0.0. With the attack device active, the algorithm also correctly classifies the per-stream behavior, despite the resulting gaps and higher standard deviation. However, when the attacker sends packets during classification for a new stream, e.g., introduces a new protocol or sends packets addressing another destination, the algorithm learns an unauthorized stream. During monitoring, packets in this stream may then not be detected as anomalous. After classification, the algorithm reports the classified behavior, including a list of the individual streams with source and destination IP, protocol, and specific IP-ID assignment behavior. In industrial environments, devices typically have a few known connections. Manually reviewing the classified behavior, therefore, ensures that no unauthorized streams are considered by the monitoring algorithm. When applying the monitoring algorithm using the configuration obtained for classification without active spoofing, we detect all spoofed packets. The other configuration, while the classified behavior is correct, is unsuited for monitoring due to the large gaps caused by the attack device. The resulting window would cover the whole IP-ID range. We would have to manually adjust the window size to a suitable value.

Monitoring Test Case 2 - PLC B. Due to internal traffic between PLC B and PLC C, the monitoring gateway does not capture the whole IP-ID sequence of PLC B in the case of a global IP-ID assignment. Despite resulting gaps, the algorithm correctly classifies the global behavior with an observed gap_{max} of 6 and a standard deviation of 0.16. With the attack device active, the algorithm wrongly classifies the behavior as per-stream, as in the other test case. Applying the monitoring algorithm, only 1 of 645 spoofed packets remained undetected as it fell into a gap during the wrap-around.

For a per-stream assignment, internal traffic between PLC B and PLC C does not cause gaps in the IP-ID sequences captured at the gateway, resulting in two monitored streams with a gap_{max} of 1 and a standard deviation of 0.0. With active spoofing, the classification algorithm also correctly classifies the behavior, despite deviations caused by the attack device. During monitoring, we again detect all spoofed packets.

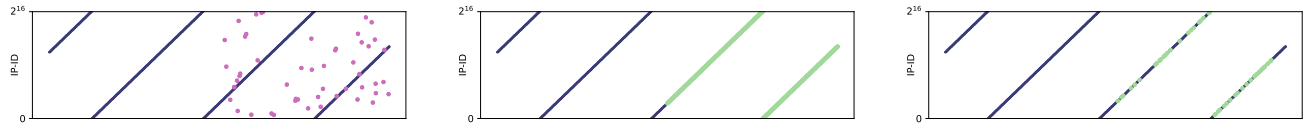
C. Discussion

IP-ID monitoring builds on the IP-ID specifications [1, 2] that require IP-IDs to be unique for the tuple of source address, destination address, and protocol for the maximum packet lifetime in the network if fragmentation is possible. This uniqueness limits the speed of connections between two hosts to 6.4 Mbps for typical packet sizes [2]. Devices follow fixed assignment behaviors, which may violate this specification at higher sending rates. However, this violation does not affect the proposed mechanism as we monitor specific assignment behaviors. The IP-ID sequences continue to follow these behaviors, although wrapping around faster.

If traffic captures contain many gaps or other noise, the classification results show higher standard deviations, leading to wider windows during monitoring. Additionally, the behavior of some streams or devices may even be undefined. Thus, to correctly classify the IP-ID assignment behavior, particularly the specific configuration of the increment and wrap-around values, we require sufficient traffic data, e.g., at least one IP-ID cycle to capture the wrap-around. However, despite wider windows, IP-IDs in packets sent by a spoofing device are still likely to fall outside the monitoring window. Further, even if the IP-ID assignment strategy of some streams is classified as undefined, the respective traffic streams are still known by the monitoring instance, allowing us to detect unknown streams initiated by an attack device during monitoring.

To closely monitor the IP-IDs of one or multiple legacy devices, we must identify a suitable position in the network, see Section IV-C. Further, we need to consider other devices that operate on packet headers, such as VPN gateways and NATs. Since these devices modify IP addresses or encapsulate IP headers, the IP-ID-based mechanism must be implemented within the cell or subnet where the monitored devices are placed. Otherwise, the algorithms cannot classify and monitor the IP-ID assignment behavior of individual devices.

Despite the discussed challenges, the proposed mechanism effectively detects attack devices spoofing network addresses. As we closely monitor the IP-ID sequence of a legacy device for the classified behavior, we can detect when the device behaves differently, hinting at a spoofing attack. In the PoC, we detected the spoofing attacks in each test case by identifying unexpected or duplicate IP-IDs. Even in the presence of gaps, an undetected duplicate can lead to multiple subsequent unexpected IP-IDs if the monitoring window advances too early. We can also detect spoofing by attack devices that use assignment behaviors other than a constant one, such as a random assignment, since spoofed packets eventually result in an anomaly, see Figure 6a.



(a) Attacker sends spoofed packets, resulting in detected unexpected IP-IDs.

(b) Attacker replaces the legacy device and mimics its IP-ID behavior.

(c) Attacker sends packets, mimicking and correcting the IP-IDs of the legacy device.

Figure 6: IP-ID sequences under different monitoring and bypassing scenarios.

Bypassing the monitoring algorithm requires the attacker to learn and then mimic the expected IP-ID assignment behavior of a legacy device. If the attacker solely spoofs the source address in generated packets or replays intercepted packets, the monitoring mechanism will detect unexpected or duplicate IP-IDs, respectively, see Figure 6a. When replacing a legacy device, the attacker will only successfully transmit packets if the IP-IDs set in the packets follow the learned behavior, as visualized in Figure 6b. If the legacy device is still active and sending traffic into the network, injecting packets with the correct next-expected IP-IDs will result in detected duplicates when the device sends further packets. Therefore, the attacker must either drop the following packets sent by the legacy device or manipulate their IP-IDs, see Figure 6c. Overall, obfuscating the spoofing attack and remaining undetected is complex and time-consuming.

VI. Conclusion

Brownfield networks contain legacy devices with insufficient security features, necessitating compensating security measures. For this purpose, we present an anomaly detection mechanism for spoofed IPv4 traffic based on the IP-ID in the IPv4 header. The mechanism consists of a classification and a monitoring algorithm, utilizing the fact that IP-IDs in packets sent by an attack device likely differ from the classified IP-ID assignment behavior of the legacy device. The proposed mechanism applies to various IP-ID counter behaviors and configurations identified by the classification algorithm. The monitoring algorithm employs a window-based approach to observe if IP-IDs correspond to this classified behavior. In the PoC implementation, we detected any spoofing attempts, even in challenging use cases where IP-ID sequences showed gaps. The passive and application-independent nature of the proposed mechanism makes it applicable for use in various domains, including critical infrastructures.

References

[1] J. Postel, *RFC 791: Internet Protocol*, Sep. 1981.

[2] J. Touch, *RFC 6864: Updated Specification of the IPv4 ID Field*, Feb. 2013.

[3] M. West and S. McCann, *RFC 4413: TCP/IP Field Behavior*, Mar. 2006.

[4] K. Sandlund and G. Pelletier, *RFC 5225: RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite*, Apr. 2008.

[5] F. Salutari, D. Cicalese, and D. J. Rossi, "A Closer Look at IP-ID Behavior in the Wild," in *Passive and Active Measurements Conference*, 2018.

[6] X. Feng *et al.*, "PMTUD is not Panacea: Revisiting IP Fragmentation Attacks against TCP," in *Network and Distributed System Security Symposium*, 2022.

[7] G. Lyon, "TCP/IP Fingerprinting Methods Supported by Nmap," in *Nmap Network Scanning*. Nmap Project, 2009, ch. 8. Remote OS Detection.

[8] A. T. Al Ghazo and R. Kumar, "ICS/SCADA Device Recognition: A Hybrid Communication-Patterns and Passive-Fingerprinting Approach," in *IFIP/IEEE Symposium on Integrated Network and Service Management*, 2019.

[9] S. M. Bellovin, "A Technique for Counting NATted Hosts," in *ACM SIGCOMM Workshop on Internet Measurment*, 2002.

[10] S. Mongkolluksamee, K. Fukuda, and P. Pongpaibool, "Counting NATted Hosts by Observing TCP/IP Field Behaviors," in *IEEE International Conference on Communications*, 2012.

[11] W. Chen *et al.*, "Exploiting the IPID field to infer network path and end-system characteristics," in *Passive and Active Measurements Conference*, 2005.

[12] S. J. Templeton and K. E. Levitt, "Detecting Spoofed Packets," in *DARPA Information Survivability Conference and Exposition*, 2003.

[13] N. R. Rodofile, "Generating Attacks and Labelling Attack Datasets for Industrial Control Intrusion Detection Systems," Ph.D. dissertation, Queensland University of Technology, 2018.

[14] I. Frazão, P. H. Abreu, T. Cruz, H. Araújo, and P. Simões, "Denial of Service Attacks: Detecting the Frailties of Machine Learning Algorithms in the Classification Process," in *Critical Information Infrastructures Security Conference*, 2019.

[15] S. Adepu, N. K. Kandasamy, and A. Mathur, "EPIC: An Electric Power Testbed for Research and Training in Cyber Physical Systems Security," in *Computer Security: ESORICS International Workshops, CyberICPS and SECPRE*, 2018.

[16] iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design, *SWaT Testbed Technical Details*. 2020.