Assessing the Impact of Attacks on an Automotive Ethernet Time Synchronization Testbed

Mahdi Fotouhi*, Alessio Buscemi*, Abdelwahab Boualouache*, Florian Jomrich†, Christian Koebel†, Thomas Engel*
*Faculty of Science, Technology and Medicine, University of Luxembourg, †Honda R&D Europe (Germany) GmbH,
*{name.surname}@uni.lu, †{Name_Surname}@de.hrdeu.com,

***Abstract*—Time Sensitive Network (TSN) standards are gaining traction in the scientific community and automotive Original Equipment Manufacturers (OEMs) due their promise of deterministic Ethernet networking. Among these standards, Generalized Precision Time Protocol (gPTP) - IEEE 802.1AS - allows network devices to be synchronized with a precision far higher than other synchronization standards, such as Network Time Protocol (NTP). gPTP is a profile of Precision Time Protocol (PTP) which, due to its robustness to delay variations, has been designated for automotive applications. Nonetheless, gPTP was designed without security controls, which makes it vulnerable to a number of attacks. This work reveals a critical vulnerability caused by a common implementation practice that opens the door to spoofing attacks on gPTP. To assess the impact of this vulnerability, we built two real gPTP-capable testbeds. Our results show high risks of this vulnerability destabilizing the system functionality.**

***Index Terms*—Connected Vehicles, Time Sensitive Networking, Cybersecurity, Automotive Ethernet**

I. Introduction

Today's vehicles feature an increasing number of services for drivers and passengers, such as smartphone integration, infotainment, connectivity to infrastructure, Advanced Driver Assistance Systems (ADAS), and autonomous driving [1]. All of these services can only be made possible with effective in-vehicle communication systems. These systems, such as CAN, FlexRay, and CANopen have traditionally been developed to meet strict end-to-end latency and deterministic requirements for in-vehicle communications [2]. However, they lack flexibility in reconfigurability and struggle to meet the bandwidth requirements of future automotive applications. To cope with this, automotive Ethernet has been proposed as a in-vehicle communication system with a physical layer tailored to automotive services. Automotive Ethernet enables high bandwidth and higher flexibility to integrate with external services [3]. It is also empowered with a set of Time Sensitive Network (TSN) standards to guarantee tight end-to-end latency for meeting safety-critical functions. However, TSN has no security controls inherently embedded [4]. Creating a secure automotive TSN profile is critical for future vehicles in terms of safety, security, and comfort [5]. The IEEE 802.1DG group [6], who works on such a profile, recognizes the importance of security controls for TSN in automotive networks.

Accurate time synchronization, provided by Generalized Precision Time Protocol (gPTP) – IEEE 802.1AS [7] –, is an integral part of TSN. gPTP is a Precision Time Protocol (PTP) profile for Audio Video Bridging (AVB) and TSN released in 2011. PTP – IEEE 1588 [8] – is a protocol for highly accurate time synchronization, which relies on a master-slave architecture. The architecture consists of multiple nodes, also called *clocks* (Section 6 of [8]). A device with a single access to the network is referred to as an Ordinary Clock (OC). This device can either act as a master, i.e. the source of synchronization packets, or as a slave, i.e. the recipients of these messages. A Boundary Clock (BC) has numerous connections to the network and is able to precisely synchronize one network segment with another. Transparent Clock (TC), instead, simply relays messages to the other clocks.

The root timing reference clock is called the Grand Master (GM). The GM sends synchronization information which is propagated through the network. The GM is not fixed, but it is elected dynamically following the Best Master Clock Algorithm (BMCA). This algorithm ranks the clocks according to their features, such as the priority number, the class, etc. Iteratively the clock with the highest ranking becomes the GM. Following the election of the GM, a synchronization *master clock* is selected for each network segment in the system. A BC forwards accurate time to the other segments to which it is connected.

The management and synchronization in a PTP system is achieved with the exchange of the following PTP messages:

- **Event messages** – are the messages through which Boundary Clocks and Ordinary Clocks exchange time-related information with one another in order to synchronize clocks located throughout the network. The event messages are $Sync$, $Follow\_Up$, $Delay\_Req$ and $Delay\_Resp$.
- **General messages** – clocks make use of them in order to assess delays over the network and allow the system to correct for such delays. The messages are $Pdelay\_Req$, $Pdelay\_Resp$ and $Pdelay\_Resp\_Follow\_Up$.
- **Announce messages** – are used to establish a clock hierarchy and elect the GM, following the BMCA.

gPTP improves certain properties of PTP and restrains some options. When compared to PTP, gPTP is far more resistant to fluctuations in latency since it requires that every switch in the network support gPTP at the MAC layer. This means that we can expect a (almost constant) propagation delay, but no queueing delay. gPTP packets are sent via Link Layer

2023 IEEE Vehicular Networking Conference (VNC2023)

Discovery Protocol (LLDP) – a protocol for the link layer that nodes use to advertise their identity, capabilities, and neighbors in an asynchronous congested Local Area Network (LAN).

The capability of guaranteeing tight end-to-end delays and a sub-microsecond clock accuracy makes gPTP particularly suitable for in-vehicle communication. Nonetheless, PTP (and, therefore, gPTP) is not equipped with security controls by default and, thus, it is exposed to a number of attacks, which are described in RFC 7384 [9]. PTP version 2 (IEEE 1588–2008) includes an experimental security enhancement in Annex K; nevertheless, it has been demonstrated that the proposed modification is not adequate to prevent a number of attacks on the protocol [10]. These attacks affect also gPTP. In particular, the impact of successful attacks against gPTP is significant due to the fact that many safety-critical services, such as autonomous driving and Vehicle to Everything (V2X) applications are dependent on time assurances.

In this paper, we reveal a critical security breach found in a common implementation practice facilitating spoofing attacks. Then, we evaluate the effects of high-risk spoofing attacks, i.e. which have severe consequences, on real testbeds. The results achieved in this paper provide a deeper understanding of the security threats linked to the lack of security controls in TSN. The contributions of this paper can be summarized as follows:

- We set up two physical testbeds which includes time-aware endpoints with hardware timestamping capable network interfaces.
- We report a vulnerability caused by a common design practice that allows attackers to conduct spoofing attacks without accessing the network domain.
- To the best of our knowledge, we are the first to demonstrate and assess the impact of this attack on a real testbed.
- We discuss a set of countermeasures to mitigate the vulnerabilities presented in the paper.

The remainder of this paper is organized as follows. In Section II we discuss related work on gPTP security. In Section III we present our novel testbeds. The spoofing attacks and their impacts are described in Section IV and Section V respectively.Section VI lists a set of countermeasures. Section VII concludes the paper.

II. Background and Related work

IEEE 1588 (PTP) standard has been designed for precise clock synchronization in networked measurement and control systems. Since its first release in 2002, the standard has been revised twice to improve procedures and fix shortcomings found in the original version [8], [11]. In 2011, the gPTP profile was introduced aiming at becoming the standard for in-vehicle Time Synchronization.

RFC 7384 [9] identified a list of cybersecurity threats on time synchronization protocols and defined their threat models and security requirements. PTP security has been widely studied [12]–[14] and different attacks have been conducted, such as:

- **Rogue master** – the attacker has gained control of one node in the network and is granted the GM position via the BMCA by sending $Announce$ packets with high GM priority [15].
- **Packet manipulation** – an attacker with a Man-In-The-Middle (MITM) position manipulates packets sent by other nodes [16].
- **Packet delay manipulation** – an attacker with a MITM position forwards the received packets with a delay, but without modifying the packets' $CorrectionField$, a field which indicates the residence of the packet in the switch[17]–[19].
- **Spoofing** – an attacker without privileged position in the network sends spoofed PTP packets impersonating a legitimate node, with the aim of providing false time to the slave nodes, desynchronizing them or causing Denial of Service (DoS).

While rogue master, packet manipulation and packet delay manipulation attacks have been successfully performed on PTP, at the time of writing spoofing attacks have not been completely demonstrated. Han and Crossley [20] were the first to attempt spoofing attacks on a real PTP testbed. In their paper, the authors present three attacks scenarios:

- a node with a MITM position (between the GM and a TC) filters out, retains and modifies specific frames;
- a node in the same position overflows the network sending frames at a high rate, thus causing DoS;
- a node is granted GM position by sending $Announce$ packets with high GM priority.

However, we argue that, in accordance to RFC 7384, the presented attacks are packet manipulation, packet delay manipulation and rogue master attacks.

Given that gPTP is the chosen PTP profile for automotive TSN, its lack of security can harm significantly human lives and property. With the aim of raising further awareness regarding this threat, in this work we conduct spoofing attacks on realistic gPTP testbeds and assess their impacts.

III. Testbed

Network devices and equipment with gPTP capabilities are already available on the market. Currently, data centers use this technology to offer precise time synchronization throughout their networks. In addition many industries, such as the automotive, are considering implementing this network protocol. However, studies on gPTP security have been mostly conducted in a simulated environment. Since it has not been evaluated on a real testbed, the security of gPTP in the real world remains largely unknown. For this reason, in this work we chose to employ a physical testbed.

In this section, we describe the two testbeds that we set up to perform security analysis on gPTP. Specifically, we built a physical testbed based with real gPTP-capable switches and a testbed that uses a software implementation of gPTP on an Accelerated Processing Unit (APU) as network switch.

For the endpoint APUs of both testbeds, we have used LinuxPTP (version 3.1-00116-g24220e8), an open-source repository for the development of PTP on Linux [21]. Finally,

Figure 1. Picture of the testbed with real gPTP capable switches.

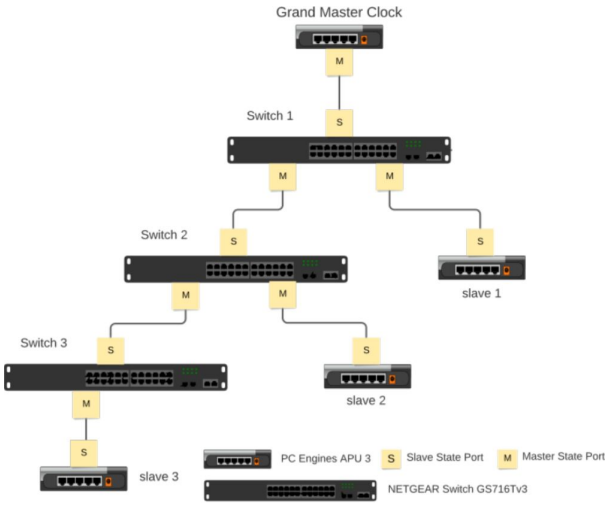

Figure 2. Architecture of the testbed with real gPTP capable switches.

to ensure that our evaluation of the attacks relies only on the internal clocks of the testbed, we have blocked the access of the devices to external time resources, such as Network Time Protocol (NTP).

A. Testbed with real gPTP-capable switch

The hardware components of the physical testbed are listed as follows:

- Three gPTP capable Netgear switches (GS716Tv3 ProSafe 16-port Gigabit Ethernet Smart Switch, 6.3.1.19-39, B1.0.0.4) [22].
- Four APUs apu2e4 [23] each with three Intel Ethernet Controller i210 and running Ubuntu 16.04. One of the APUs is dedicated to the attack tool as an attacking device. Other three APUs are considered as gPTP endpoints.

Figure 1 shows a picture of the aforementioned testbed, while Figure 2 describes its architecture.

B. Testbed with LinuxPTP software-based switches

To ensure that the results are not limited to a specific hardware, we created a parallel software-based testbed. This testbed contains the three PC Engine APUs (apu2e4) used in the physical testbed and one laptop as the attacker's device running LinuxPTP. While one of the APUs was configured to work as a gPTP relay instance, the second and third APUs were configured to work as the GM and the slave endpoints respectively. In Figure 3, we show the architecture of the testbed with LinuxPTP software-based switches.

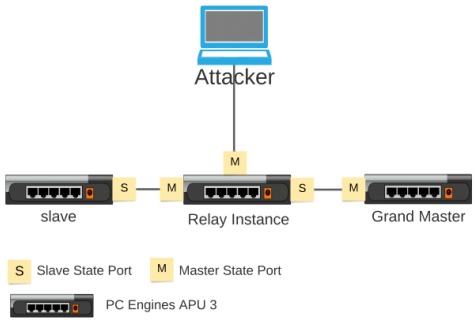

Figure 3. Architecture of the testbed with LinuxPTP software-based switches.

C. Monitoring

gPTP daemons only synchronize the PTP Hardware Clocks (PHCs). In order to monitor the impact of the attacks, we use the PHC time instead of the system clock on interfaces. To monitor the PHC time directly on each endpoint the following command is used:

```
$while :; do sudo phc_ctl [NIC] --get; done
```

*[NIC] = Network Interface Name

To observe clock offsets and delays, we use the real-time report generated by the LinuxPTP daemon at running time. Given that a slave node is considered synchronized with the GM if the clock offset is below 100 ns [24], we constantly monitor that this requirement is met.

Moreover, through the control panel of the switches, we keep track of the states of the ports (slave or master), as well as the joining status to the gPTP domain (port enabled/disabled), the propagation delay, and the gPTP packets dropped by the switch. Finally, Tcpdump [25] is used on each endpoint to sniff packets sent from or received by a device. The captured packets are then analyzed with Wireshark [26].

D. Attack Tool

To conduct attacks against the testbed, we require a trustworthy attack tool that can produce, alter, and process PTP packets. We implemented our attack tool, PTPAttack, in Rust, i.e. a system-level programming language specialized for low levels of network stacks and operating system access. PTPAttack can join the gPTP domain by delivering legitimate Pdelay responses with appropriate and stable latency. PTPAttack may also generate $Pdelay$, $Announce$, $Sync$, and $Follow_Up$ messages with arbitrary values by parsing PTP packets. PTPAttack

enables the attacker to execute several attacks from an internal injector point.

IV. Attack Description

In this section, we report the characteristics of the rogue master and spoofing attacks and the methodology followed to conduct them against the testbeds.

A. Threat Model

The threat model is the one of an internal attacker, i.e. an adversary who knows the encryption/authentication keys and/or who is capable of manipulating legitimate traffic in the network and/or generating its own traffic while making it appear legitimate to the attacked nodes. Since the protocol does not include any security control, the adversary can exploit this vulnerability by simply eavesdropping and analyzing the traffic to obtain sensitive information. To achieve such an advantage, the adversary must gain access to a segment of the network. Alternatively, the adversary can inject its own traffic into the network, which can be used in a variety of attacks, such as DoS, replay attacks, or impersonation attacks. In this work, we assume that the adversary has already gained access to the network. The methodology to perform such an access is out of the scope of this paper.

B. Rogue Master Attack

In the rogue master attack, the malicious slave pretends to be time-aware and joins the gPTP domain by responding to the $Pdelay\_req$ packets properly received from the switch. The malicious node keeps listening to the network for $Pdelay\_req$ packets and generates $Pdelay\_resp$ and $Pdelay\_resp\_follow\_up$ packets by modifying received $Pdelay\_req$ packets.

In order to join the domain successfully, it is essential for the attacker to have the switches calculating a low propagation delay. After joining the domain, the attack tool seeks to obtain the GM position. To make other nodes believe that it is the most accurate clock in the domain and, thus, be elected as the GM, the malicious node generates high-priority $Announce$ packets. Once it has achieved the status of GM, the malicious node completes the rogue master attack by sending arbitrary timestamps to the slaves in $Sync/Follow\_up$ packets, thus producing a false time in their clock and/or desynchronizing them.

C. Spoofing Attack

According to the gPTP protocol, $Sync/Follow\_up$ packets should be forwarded by a switch only if they are sent from a port in a master state. Therefore, it should not be possible for an adversary to send malicious $Sync/Follow\_up$ packets without being in a MITM position or being the GM, e.g. by conducting a Rogue Master attack [9]. However, in both the testbeds described in Section III we identified a vulnerability that permits spoofing attacks regardless of the position of the attacker. This vulnerability is linked to a wrong design choice made with respect to the $Ethertype$ field. The $Ethertype$ is a two-octet field in an Ethernet frame that indicates how the payload should be processed. The gPTP profile identifies $Ethertype$ as an essential field in the packets, but it does not provide detailed instructions on how to process it.

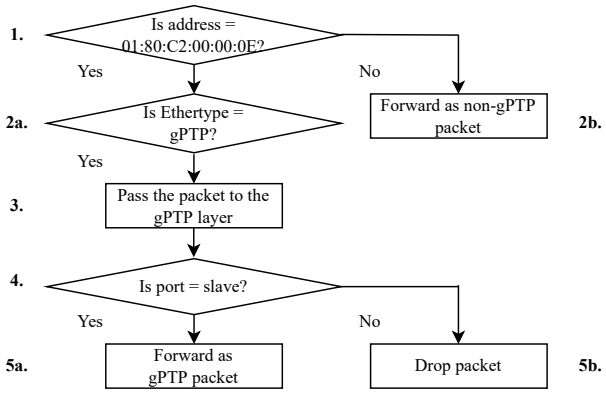

Figure 4. Implementation of security control for gPTP packet forwarding by LinuxPTP and Netgear. When the address check fails, the Ethertype control and further steps are bypassed and the packet is forwarded.

In Figure 4, we illustrate how the Netgear and the LinuxPTP-based switches process a gPTP packet.

gPTP packets are sent via LLDP Multicast MAC address $01:80:C2:00:00:0E$ (Step 1). Switches receiving packets with this address must check the $Ethertype$ field (Step 2a). If $Ethertype$ indicates a gPTP packet, the switch passes it to the gPTP layer (Step 3) and, subsequently, it checks the ingress port state (Step 4). If the port is in a slave state, the switch forwards the gPTP packet (Step 5a), otherwise it drops it (because the switch is not expected to receive $Sync/Follow\_up$ packets from a slave node). It is to be noted that if the packet is an $Announce$ packet with a higher priority in BMCA, the switch accepts that packet and changes the port state to the slave state. Moreover, if the packet is an $Pdelay$ packet the switch will process but not forward it as the $Pdelay$ mechanism in gPTP is hop by hop.

We found that both the Netgear and the LinuxPTP software-based switches do not check $Ethertype$ correctly, thus identifying gPTP packets only by considering the destination address (Step 1). As a consequence, it is sufficient for an attacker to change the destination MAC address to an arbitrary address, so that the switch does not identify the packet as a gPTP packet and forwards it to its destination without further processing (Step 2b).

This vulnerability implies that, in order to conduct a successful spoofing attack against the gPTP, the attacker only needs to find the $ClockIdentity$ of each master port. This can be achieved:

(a) Through path trace Type, Length, Value (TLV) values in multicasted $Announce$ messages.
(b) By eavesdropping.
(c) By scanning the network for MAC addresses and building the clocking identity of each node using its own MAC address.

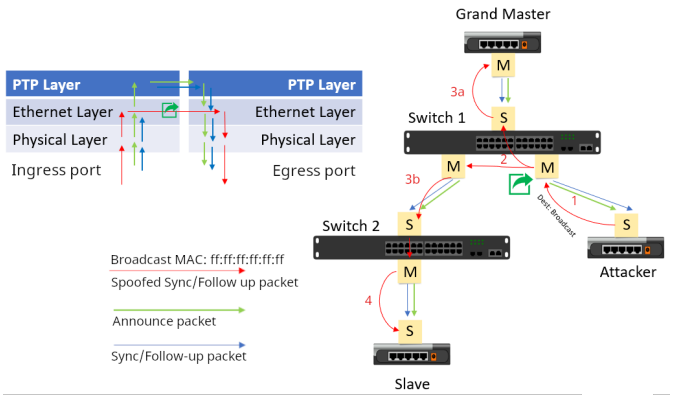

Figure 5. Example of spoofing attack with Broadcast MAC address. The flow of the spoofed packets is coloured in red. The switch is not considering the Ethertype value in the packet header and gives priority to the MAC address for detecting the gPTP packets. As a consequence, the attacker can bypass the gPTP port state limits and send spoofed sync & follow up packets with arbitrary MAC address.

Subsequently, to bypass the master or the disabled ports on the switches, the attacker sets the packet destination MAC address to an arbitrary address – except for the LLDP MAC Address. For example, the attacker can use the Broadcast MAC address $FF:FF:FF:FF:FF:FF$ to send the packet to all connected endpoints or the MAC address of a single endpoint for a selective attack.

This causes the switch to forward $Sync/Follow\_up$ messages to other nodes. Since gPTP endpoints do not match the PTP $Ethertype$ and $LLDP$ MAC address, the packet is considered as a valid gPTP packet. Figure 5 illustrates the details of this attack.

D. Attack Scenarios

Once an adversary has ensured that the malicious traffic can propagate through the network by following the rogue master or the spoofing attack mentioned in Section IV-B and Section IV-C respectively, the attack can be injected. The attacker can generate new $Sync/Follow\_up$, $Announce$ or $Pdelay$ packets or simply modify the timestamp, destination address, clock identity, port number and sequence id of the packets received from the GM and broadcast them. In particular, the attack can aim at achieving three objectives:

(a) **False Time** - the attacker changes the clock of the slaves to any arbitrary date and time by sending spoofed $Sync/Follow\_up$ packets.

(b) **Desynchronization** - the attacker makes the slaves desynchronized from the GM node or from each other. This is achieved by either unicasting different spoofed $Sync/Follow\_up$ packets to each slave or broadcasting highly variable, e.g. random, timestamps to all packets.

(c) **DoS** - The attacker can produce DoS by using the spoofing attack under various scenarios. In this work, we consider two scenarios: i) the $Announce$ packets of a legitimate GM are spoofed and sent with low priority values. ii) the adversary spoofs the $Announce$ packets of a slave node containing high priority values.

It is to be noted that when desynchronization is carried out, the adversary also achieves false time. For instance, if the adversary continuously sends $Sync/Follow\_up$ packets with a tampered time, he can firstly achieve false time and then desynchronization. Vice versa, false time does not necessarily imply desynchronization. In gPTP, the messages are sent as multicasts by default. So, when an attacker spoofs the GM, $Sync$ and $Follow\_up$ messages are multicasted to all nodes. This means that they can all be synchronized at the same tampered time.

V. Impact Evaluation

As described in Section IV-D, rogue master and spoofing attacks have similar capabilities in achieving false time, desynchronization and DoS. In this section, we evaluate the impact of the performed attacks by analyzing false time, desynchronization and DoS independently.

A. False Time

In LinuxPTP, the amount of time needed by the slaves to adjust their clocks depends on the specific configuration chosen. For instance, in the default options of LinuxPTP, the offset and clock frequency gradually increase until the new target time is reached. However, if LinuxPTP is set with the option $step\_threshold = n$ – where n is the maximum tolerable difference that can occur between the current clock and the requested clock (in seconds) – and the threshold is overcome, the PHC clock is adjusted to the attacker's arbitrary time [24].

We observed that when the GM is an endpoint, the slaves continuously change their time to the spoofed time and then reset to the legitimate GM clock time. Also, decreasing the spoofed packets sending intervals increases the attack efficiency. Additionally, the attacker can produce a certain time difference between the slaves and the GM clock by adding desired time difference to the timestamp sent by the GM.

In Figure 6, we show the impact that a multicast spoofing attack aiming at false time has on the target slaves. The target slaves abruptly change their timestamp to follow the information contained in the spoofed $Sync/Follow\_up$ messages. The two slaves node keep being synchronized to each other, but to the wrong time. It is to be noted that, in the case of a spoofing attack, the attacker does not manage to keep the slaves continuously synchronized to the desired time, due to the presence of the original GM which keeps sending its own $Sync/Follow\_up$ packets. By contrast, in a rogue master attack, being the malicious node the official GM, the slaves are uninterruptedly synchronized to the desired time.

We argue that this attack does not only impact gPTP, but TSN as whole. In fact, slaves synchronized at a false timestamp might be dropped in the case of high priority traffic, thus causing DoS.

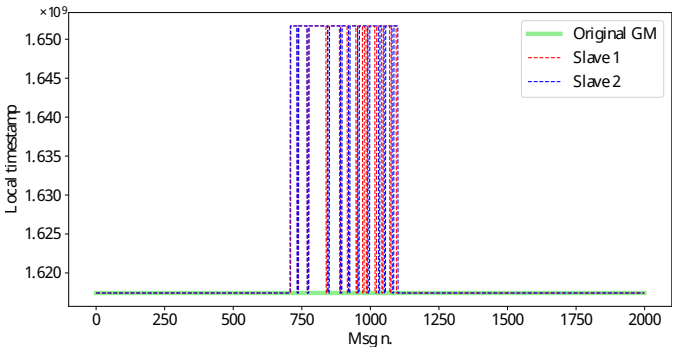

Figure 6. Example of multicast false time attack conducted on our testbed against Slave 1 and 2. The x-axis represents the sequence of $Sync$ messages collected by the daemons in the slaves, while the y-axis represents the timestamp to which each node is synchronized. Following the attack, which in our example occurs after the 700th $Sync$ packet, the two slaves are synchronized to a spoofed timestamp. However, the synchronization of the slaves to the new timestamp is not constant due to the presence of packets from the original GM. When the attack is over, the clocks are synchronized again to the original GM. It is to be noted that, due to the high granularity of the plot, the timestamps to which the clocks are synchronizing appear constant, but they are actually increasing.

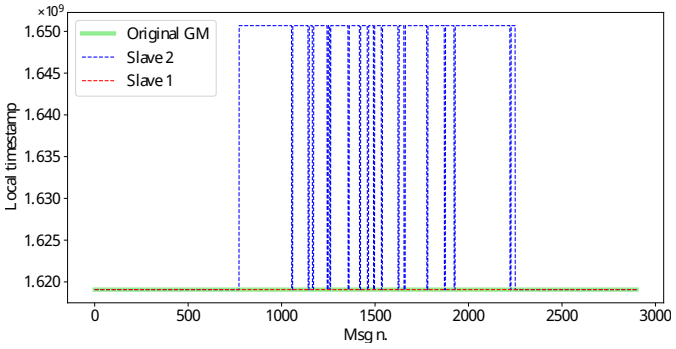

Figure 7. Example of unicast desynchronization attack conducted on our testbed against Slave 2. The x-axis represents the sequence of $Sync$ messages collected by the daemons in the slaves, while the y-axis represents the timestamp to which each node is synchronized. Following the attack, which in our example occurs after the 700th $Sync$ packet, only Slave 2 is synchronized to a spoofed timestamp and, thus, desynchronized from the rest of the network. Similarly to Figure 6, the synchronization of Slave 2 to the new timestamp is not constant due to the presence of packets from the original GM. When the attack is over, the clock of Slave 2 is synchronized again to the original GM. It is to be noted that, due to the high granularity of the plot, the timestamps to which the clock are synchronizing appear constant, but they are actually increasing.

B. Desynchronization

The attacker can send different times to the nodes. For instance, a node can receive the fabricated time $T1$, while another node receives the fabricated time $T2$. In this scenario, slave clocks can no longer remain synchronized, and each follows a unique time.

The impact of the desynchronization not only depends on the attacking scenario, i.e. unicast or multicast, but also on the configuration of the daemon running on each slave clock (such as the $step\_threshold = n$ option for LinuxPTP).

In Figure 7, we demonstrate the impact of a unicast desynchronization attack on a target slave, which is the only one being desynchronized. Similarly to the false time attack, the attacker struggles to keep the target slaves continuously synchronized to the desired time, as the original GM keeps sending its own $Sync/Follow\_up$ packets.

C. DoS

While rogue master and spoofing attacks follow a similar logic for what concerns False Time and Desynchronization, different degrees of DoS can be achieved by these two attacks.

Spoofing attacks can aim at sending false $Announce$ packets. The slaves receive $Announce$ packets with different priority values from two sources, i.e. the original GM and the adversary, and they get stuck in the master selection mode. As a consequence, they cannot process the received time packets anymore. Figure 8 illustrates an example of this DoS attack conducted on our testbed. Alternatively, the adversary can spoof $Announce$ packets of a legitimate GM with low priority values. This scenario causes the slaves to choose the wrong node (the spoofed slave) as the GM while no $Sync/Follow\_up$ packets are sent. In particular, in the first testbed, when the switch receives no $Sync/Follow\_up$ packets, it generates new packets and sends them through the egress ports in the

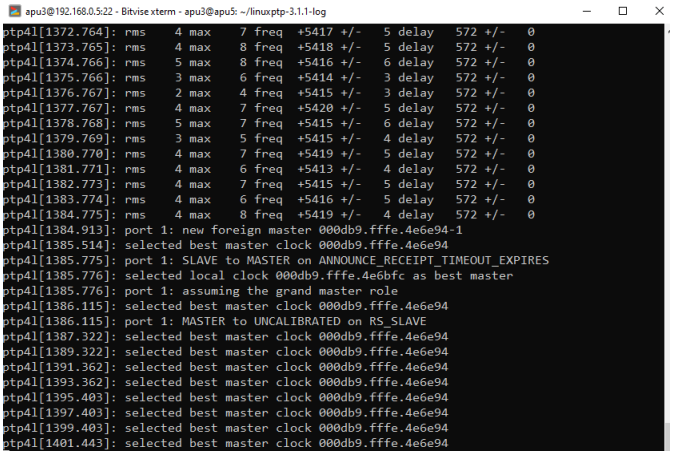

Figure 8. Example of the impact of an $Announce$ packets-based DoS attack on a slave node. When the attack starts, the slave gets in master selection mode and does not process $Sync/Follow\_up$ packets anymore.

master state. In this context, the $Follow\_up$ packets generated by the switch contain a constant value – the last received timestamp – in the $preciseOriginTimestamp$ field, while the $CorrectionField$ reports a wrong value. Furthermore, we observed that the slaves discard the received $Follow\_up$ packets after 16 s. Subsequently, they reset to the time when the attack started. This mechanism prevents the slaves from running free, but it causes false time.

In the case of a rogue master attack, the adversary can split the gPTP domain into sub-domains using the path route TLV in $Announce$ packets. In this attack, the attacker chooses a switch as a segmentation point and puts its clock identity as the 8th clockIdentity in the route path TLV of $Announce$ packets (see Figure 9). In the switch, this causes a change in the ingress port state from the slave to the master state. Therefore, the

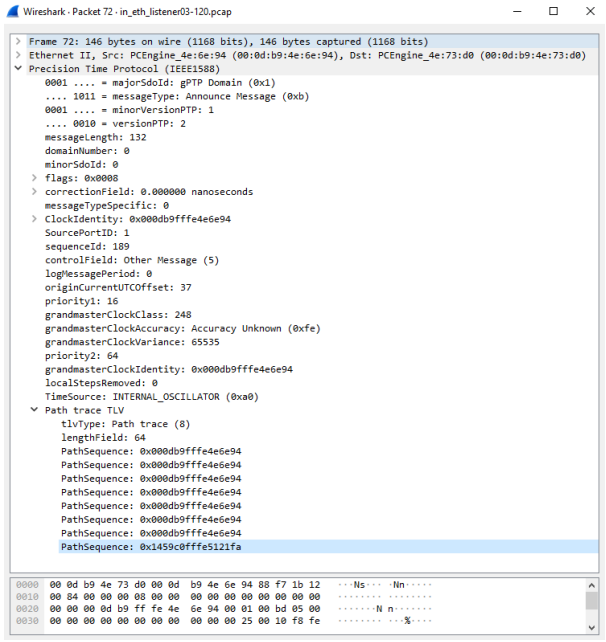

Figure 9. *Announce* packet with malformed route path TLV caused by a rogue master attack. The adversary fills out the path trace TLV with arbitrary data except for the last record (highlighted in blue), which is set to the victim's *clockIdentity*. As a consequence, the victim node rejects all $Sync/Follow\_up$ packets because they appear coming from the same node.

switch not only rejects all the $Sync/Follow\_up$ packets, but also the *Announce* packets even when they have high priority. The consequence is that the domain is split into two different segments, each having a specific GM. Also, the attacker can make more segments by generating customized *Announce* packets for each switch.

D. Impact on TSN and on the vehicles

Attacking time protocols makes delay measurement non-deterministic, increase the consumption of resources, and cause data disruption in TSN [27]. In automotive scenarios, the consequences of these attacks might hinder the proper functioning of the vehicle, be critical for the safety of the passengers, and produce financial loss and damage.

Inaccuracies in time synchronization within the automotive TSN can have critical consequences. A timing-based attack can potentially lead to synchronization issues, communication failures, up to safety concerns. In particular, such an attack may potentially cause the failure of the braking or steering systems of a vehicle that rely on time-sensitive communication and lead to an accident. However, the further exploration of such automotive threat scenarios, including safety-related ones, is not in scope of this work. Additionally, false time may result in incorrect certificate validation and potential bypassing of security controls ensuring the integrity and security of digital certificates and secure communication, such as SSL/TLS encryption.

VI. Countermeasures

In this section, we discuss how to reduce the risks associated with the attacks presented in Section IV.

A. Configuration changes

Typically, the GM is elected dynamically through the BMCA. However, the gPTP protocol allows the GM to be pre-configured and unchangeable. Clearly, relying on a single node for the time synchronization consistently reduces the fault tolerance of the network, but it is an effective prevention mechanism against rogue master attacks.

B. gPTP Implementation Improvement

As previously discussed, many gPTP implementations seem to be fundamentally based on PTP, and are adapted to run gPTP by inputting some configuration parameters. This practice increases the security risks, as an attacker may lead to a PTP state that is unknown in the gPTP state machine. To protect the gPTP network from such threats the gPTP-capable device manufacturers and gPTP stack designers should perform a thorough check to verify that their products fully comply with gPTP.

In particular, to protect the network from the discussed spoofing attacks, the gPTP packets not only should be limited to the LLDP MAC address and gPTP Ethtertype. Rather, the Ethertype should be checked before the MAC address to make sure that all gPTP messages are identified correctly. If not, the MAC address should be dropped. This should be done both in gPTP-capable switches and endpoint devices.

C. Other Security Mechanisms

Other security control mechanisms can be added to gPTP in order to increment its robustness against a variety of threats, including those We identify four categories of security control mechanisms:

- Integrated Security Mechanisms – AUTHENTICATION TLV are appended to the PTP messages in order to provide source authentication and message integrity, and prevent replay attacks [8]. For instance, spoofing attacks from external attackers on gPTP could be prevented with authentication and integrity checks. In this case, the attacker would not be able to produce valid gPTP packet, thus making malicious packets to be dropped.
- External Security Mechanisms – IEEE 1588 [8] suggests using some security mechanisms, such as IEEE 802.1AE MACSec [28] and/or IPSec [29], which were not originally included in PTP, but can be used to address some security requirements.
- Architectural Security Mechanisms – PTP offers architectural solutions for threat detection and mitigation, such as redundancy of time and paths.
- Key Management – key management is currently not covered by PTP. It would be required to manage and distribute keys and other security parameters in the authentication and message verification process.

D. Intrusion detection

Intrusion Detection Systems (IDSs) could also be used as countermeasures complementary to the previously mentioned security mechanisms. IDSs allow the detection of any suspicious deviation from normal behaviors. For example, IDSs could monitor devices' clocks to detect desynchronizations. However, advanced internal attacks may require more sophisticated IDSs. Here Machine Learning (ML) is a key enabler in detecting such behaviors [30]. More specifically, ML models could be built based on normal data and tuned to detect malicious behaviors.

VII. Conclusion

In this paper, we have described a security vulnerability on automotive Ethernet caused by the incorrect check of the $Ethertype$ field in gPTP packets by switches. This vulnerability seems a common design flaw in the implementation of the gPTP protocol. This security breach can allows adversaries to easily bypass the gPTP layer and conduct high-risk spoofing attacks even without joining the gPTP domain.

Based on two real testbeds, we tested different attacking scenarios resulting from this vulnerability. Our tests highlighted the major negative impact that this vulnerability has on gPTP with respect to time accuracy, synchronization and service availability.

This work shows that the poor design of gPTP-enabling features in network switches raises consistently the risk of attacks. Given that gPTP is the chosen PTP profile for automotive applications, these attacks raise concerns regarding the health of the vehicles and their passengers.

Future work includes assessing the impact of a diverse set of attacks on real testbeds, with the aim of providing fundamental information useful for the implementation of security controls for gPTP.

Acknowledgment

This work was supported by the BRIDGES grant, funded by the Luxembourg National Research Fund (FNR), and by Honda R&D Europe (Germany) GmbH.

References

[1] M. Traub, A. Maier, and K. L. Barbehön, "Future automotive architecture and the impact of IT trends," *IEEE Software*, vol. 34, no. 3, pp. 27–32, 2017.

[2] M. L. Sichitiu and M. Kihl, "Inter-vehicle communication systems: a survey," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 2, pp. 88–105, 2008.

[3] P. Hank, T. Suermann, and S. Müller, "Automotive Ethernet, a holistic approach for a next generation in-vehicle networking standard," in *Advanced Microsystems for Automotive Applications 2012: Smart Systems for Safe, Sustainable and Networked Vehicles*, Springer, 2012, pp. 79–89.

[4] D. Ergenç, C. Brülhart, J. Neumann, L. Krüger, and M. Fischer, "On the security of IEEE 802.1 time-sensitive networking," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2021, pp. 1–6.

[5] "Draft Standard for Local and metropolitan area networks — Time-Sensitive Networking Profile for Automotive In-Vehicle Ethernet Communications," *IEEE Draft Std P802.1DG/D1.4 Dec 2020*, 2020.

[6] IEEE. "P802.1DG – TSN Profile for Automotive In-Vehicle Ethernet Communications." (), [Online]. Available: https://1.ieee802.org/tsn/802-1dg/ (visited on 03/08/2023).

[7] "IEEE Draft Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks," *IEEE Draft Std P802.1AS/D2.0 Feb 2008*, 2008.

[8] IEEE Std 1588™-2019, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE SA, Standard, 2019.

[9] M. Mizrahi, "RFC 7384: Security requirements of time protocols in packet switched networks," *Tools. ietf. org (online) https://tools. ietf. org/html/rfc7384 (accessed 26 Sep 2020)*, 2014.

[10] E. Itkin and A. Wool, "A security analysis and revised security extension for the precision time protocol," *IEEE Transactions on Dependable and Secure Computing*, pp. 22–34, 2017.

[11] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems - Redline," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002) - Redline*, pp. 1–300, 2008.

[12] G. Gaderer, A. Treytl, and T. Sauter, "Security aspects for IEEE 1588 based clock synchronization protocols," in *Proc. IEEE Int. Workshop Factory Commun. Syst.(WFCS)*, Citeseer, 2006, pp. 247–250.

[13] J. Tsang and K. Beznosov, "A security analysis of the precise time protocol (short paper)," in *International Conference on Information and Communications Security*, Springer, 2006, pp. 50–59.

[14] A. Treytl and B. Hirschler, "Security flaws and workarounds for IEEE 1588 (transparent) clocks," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, IEEE, 2009, pp. 1–6.

[15] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. A. Wojciak, and S. Guendert, "Impact of Cyberattacks on Precision Time Protocol," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 5, pp. 2172–2181, 2020.

[16] W. Alghamdi and M. Schukat, "Precision time protocol attack strategies and their resistance to existing security extensions," *Cybersecurity*, vol. 4, no. 1, pp. 1–17, 2021.

[17] J. Neyer, L. Gassner, and C. Marinescu, "Redundant schemes or how to counter the delay attack on time synchronization protocols," in *2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, IEEE, 2019, pp. 1–6.

[18] Q. Yang, D. An, and W. Yu, "On time desynchronization attack against IEEE 1588 protocol in power grid systems," in *2013 IEEE Energytech*, IEEE, 2013, pp. 1–5.

[19] R. Annessi, J. Fabini, F. Iglesias, and T. Zseby, "Encryption is futile: Delay attacks on high-precision clock synchronization," *arXiv preprint arXiv:1811.08569*, 2018.

[20] M. Han and P. Crossley, "Vulnerability of IEEE 1588 under Time Synchronization Attacks," in *2019 IEEE Power & Energy Society General Meeting (PESGM)*, 2019, pp. 1–5.

[21] linuxptp, *ptp4l*. [Online]. Available: http://linuxptp.sourceforge.net/.

[22] Netgear. "A New Generation of Gigabit Smart Switches." (2019), [Online]. Available: https://www.downloads.netgear.com/files/GDC/datasheet/en/GS716Tv3-GS724Tv4-GS748Tv5.pdf?_ga=2.213794839.222651013.1652102686-21276228.1652102686.

[23] P. Engine, *apu2e4*, May 2016. [Online]. Available: https://pcengines.ch/apu2e4.htm.

[24] Avnu Alliance, *TSN Documentation Project for Linux*. [Online]. Available: https://tsn.readthedocs.io/timesync.html.

[25] Tcpdump Group. "Tcpdump and Libcap." (), [Online]. Available: https://www.tcpdump.org/ (visited on 11/26/2021).

[26] Wireshark. "Wireshark - Go deep." (), [Online]. Available: https://www.wireshark.org/ (visited on 11/26/2021).

[27] E. Grossman, T. Mizrahi, and A. Hacker, "Deterministic Networking (DetNet) Security Considerations," RFC Editor, Tech. Rep. RFC9055, Jun. 2021, RFC9055.

[28] "802.1AE: MAC Security (MACsec)," *IEEE Standard Association*, 2018.

[29] S. Kent, "IP encapsulating security payload (ESP)," Tech. Rep., 2005.

[30] A. Boualouache and T. Engel, "A Survey on Machine Learning-based Misbehavior Detection Systems for 5G and Beyond Vehicular Networks," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2023.