Missed Opportunities: Measuring the Untapped TLS Support in the Industrial Internet of Things

Markus Dahlmanns*, Johannes Lohmöller*, Jan Pennekamp*,
Jörn Bodenhausen*, Klaus Wehrle*, Martin Henze§,‡

*Communication and Distributed Systems, RWTH Aachen University, Aachen, Germany
§Security and Privacy in Industrial Cooperation, RWTH Aachen University, Aachen, Germany
‡Cyber Analysis & Defense, Fraunhofer FKIE, Wachtberg, Germany
{dahlmanns, lohmoeller, pennekamp, bodenhausen, wehrle}@comsys.rwth-aachen.de · henze@cs.rwth-aachen.de

ABSTRACT

The ongoing trend to move industrial appliances from previously isolated networks to the Internet requires fundamental changes in security to uphold secure and safe operation. Consequently, to ensure end-to-end secure communication and authentication, (i) traditional industrial protocols, e.g., Modbus, are retrofitted with TLS support, and (ii) modern protocols, e.g., MQTT, are directly designed to use TLS. To understand whether these changes indeed lead to secure Industrial Internet of Things deployments, i.e., using TLS-based protocols, which are configured according to security best practices, we perform an Internet-wide security assessment of ten industrial protocols covering the complete IPv4 address space.

Our results show that both, retrofitted existing protocols and newly developed secure alternatives, are barely noticeable in the wild. While we find that new protocols have a higher TLS adoption rate than traditional protocols (7.2 % vs. 0.4 %), the overall adoption of TLS is comparably low (6.5 % of hosts). Thus, most industrial deployments (934,736 hosts) are insecurely connected to the Internet. Furthermore, we identify that 42 % of hosts with TLS support (26,665 hosts) show security deficits, e.g., missing access control. Finally, we show that support in configuring systems securely, e.g., via configuration templates, is promising to strengthen security.

CCS CONCEPTS

• **Networks → Security protocols**; • **Security and privacy → Security protocols**.

KEYWORDS

industrial communication, network security, security configuration

1 INTRODUCTION

Traditionally, industrial networks, i.e., networks used for process and factory automation, were explicitly designed to be isolated from other computer networks [11, 33, 64]. Consequently, industrial protocols, e.g., Modbus, were specified without any security measures, e.g., encryption, authentication, or integrity protection [15, 21, 65]. However, similar to paradigms like the Internet of Things (IoT) which increasingly interconnects consumer hardware, also industrial networks converge with other networks resulting in an Industrial Internet of Things (IIoT) [6, 86, 87]. While this convergence enables a plethora of novel functionality [51, 56, 77, 87], it also significantly increases the number of attack vectors and thus demands tighter security requirements [86], especially end-to-end secure communication and authentication. Most importantly, traditional (insecure) industrial protocols without any security measures must not be used anymore, especially for communication via the Internet.

Indeed, related work confirms the availability of such traditional and insecure protocols on the Internet: While more than 60,000 publicly accessible systems use traditional protocols [64] to provide their services, the protocols are also heavily used by Industrial Control Systems for communication via the Internet [70]. Communication between such unsecured industrial endpoints is not protected against attacks such as eavesdropping or message alternation, allowing attackers to retrieve confidential business information or even control production lines [11, 39, 45, 79, 95]. Such attacks can have a significant impact on the environment, machinery, or even human life, as evidenced by NotPetya and other attacks [39, 95], as well as are expected to be more frequent in the future [22].

As a countermeasure, traditional protocols have been retrofitted (as of 2013) to account for these new security requirements: They now rely on TLS to end-to-end protect the communication and allow for client authentication as well as access control [47, 49, 50, 66, 72]. Complementing these efforts, modern protocols such as AMQP, MQTT, and CoAP, specifically targeting IIoT communication but also commonly used in the IoT, have been explicitly designed to provide security via TLS [3, 91]. However, to capitalize on their promised security benefits, these protocols need to be deployed on Internet-facing appliances and configured securely, e.g., w.r.t. used cryptographic ciphers and authentication methods [89].

While a first Internet-wide analysis of the secure-by-design industrial protocol OPC UA identified thousands of *theoretically* secure Internet-facing deployments, a surprisingly high number of 92 % exhibited *practical* security issues, mostly due to configuration deficits [17]. Given that TLS is the de-facto standard for secure communication, e.g., on the Web, the question arises whether similar

issues prevail for TLS-based (I)IoT protocols. As little is known whether these retrofitted and modern (I)IoT protocols relying on TLS for security are currently deployed in the wild and, if so, configured securely, we set out to shed light on these issues.

In this paper, we thus study to which extent (I)IoT deployments rely on TLS-based protocol variants to end-to-end secure their communication and whether they are configured securely, i.e., we look into today's attack surface. To this end, we actively scan the entire IPv4 address space (~3.7 billion IP addresses) on 20 ports to identify TLS-enabled (I)IoT deployments, assess their configuration relying on well-known best practices [8, 28, 31, 61, 89], and derive guidelines to increase the security of future deployments.

Contributions: Our main contributions are as follows.

- We identify ten TLS-based (I)IoT protocols and scan the entire IPv4 address space to quantify TLS adoption among these protocols. While indicating a higher TLS adoption rate of 7.2 % on hosts relying on secure-by-design protocols, our results show that TLS adoption on hosts employing protocols offering a retrofitted variant is comparatively low (only 0.4 % offer TLS support). Thus, the majority of 967,551 found (I)IoT deployments remain insecure and do not benefit from the security offered by TLS.
- For TLS-enabled systems, we assess whether their configuration in practice indeed results in a sufficient security level. To this end, we analyze whether the found deployments adhere to common TLS configuration guidelines [31, 61, 89]. Here, 42 % of TLS-enabled deployments suffer from configuration issues that impact security, ranging from deprecated TLS versions over selection of insecure ciphers to massive certificate reuse.
- To foster open science and allow for reproducibility of our results, we release our implementations of TLS-based industrial protocols for zgrab2 [14] and our anonymized dataset [18].

2 SECURE INDUSTRIAL COMMUNICATION

Automated industrial production relies on communication protocols enabling plant components to exchange control commands and sensor values [33, 65]. In this section, we discuss the evolution of industrial protocols for isolated networks to secure (I)IoT protocols. While these protocols, in theory, are able to provide confidential, integrity-protected, and authenticated communication, we deliberate that the actual security level depends on their configuration.

Evolution of Industrial Communication: Relying on bus communication, e.g., via RS-485, first industrial networks were designed for isolated communication where, e.g., SCADA terminals communicated with local Programmable Logic Controllers (PLCs) to control production processes and visualize their status [33]. To communicate via these buses, vendors of industrial devices each specified their own protocol, e.g., Siemens developed S7, Allen-Bradley (Rockwell Automation) formalized EtherNet/IP [15], and Modicon (Schneider Electric) established Modbus [67]. We provide a concise overview of relevant industrial communication protocols and their respective evolution over time in Table 1.

Even though these protocols differ among other things in terms of packet layout, they share the same core features: reading and writing values of plant components [15, 49, 65, 100], e.g., to get sensor readings or control processes. To simplify the communication with office networks managing productions, vendors ported

	Protocol	Type	Bus	IP	TLS
Traditional	Modbus	C/S	1979 [65]	2006 [65]	2018 [66]
	DNP3	C/S	1993 [21]	2014 [47]	
	IEC 104	C/S	1995* [48]	2000 [49]	2014 [50]
	EtherNet/IP	C/S	–	2001 [15]	2015 [72]
	Siemens S7	C/S	≤ 2004+	≤ 2004+,† [85]	2016+
	Tridium Fox	C/S	–	≤ 2004+	≤ 2013+
Modern	AMQP	PubSub	–	2003+	2007+
	OPC UA	C/S & PubSub	–	2006 [73, 74]	
	MQTT	PubSub	–	2013 [3]	
	CoAP‡	C/S	–	2014 [91]	

* IEC 101.   + No official specification available.   † ISO-on-TCP.   ‡ UDP/DTLS-based.

Table 1: Types (Client/Server (C/S) & Publish/Subscribe (PubSub)) and specification years of industrial protocol variants.

their protocols for usage via Ethernet, Internet Protocol (IP), and in most cases TCP (cf. Table 1). Hence, PLCs nowadays act as servers allowing clients to read and write values or commands. We refer to these *client/server (C/S)*-based protocols as *traditional protocols*.

Recently, a string of *modern protocols* surfaced focusing on a small resource footprint (CoAP) [91] or implementing the modern *Publish/Subscribe (PubSub)* communication pattern (AMQP, MQTT) [3, 37]. In the latter, sending and receiving entities connect to a central broker instance, facilitating many-to-many communication by distributing received data to all clients that previously subscribed to a specific topic. Implementing both of these communication concepts, OPC UA [73, 75] accounts for diverse needs.

More importantly, to address the need for security, especially in Internet-facing deployments, OPC UA provides attested [29] security measures by design [76], using specialized security paradigms but also with a TLS-based variant. While modern protocols were designed with security in mind to account for security and safety needs of modern (I)IoT deployments, traditional protocols underwent a retrofitting process. Since 2013, the specifications of traditional protocols have been updated to enable TLS communication.

As secure communication is an inherent building block for the safe operation of (I)IoT appliances to prevent attacks leading to production outages or harm to humans, the question arises whether these variants arrived in practice at Internet-reachable deployments.

Pitfalls in Secure Communication: TLS, the predominantly used security protocol on the Internet, and DTLS provide confidential, integrity-protected, and authentic communication for TCP and UDP connections [82, 83], respectively (we use (D)TLS as an abbreviation for TLS or DTLS in the remainder of this paper). However, the achieved level of communication security depends on the (D)TLS *version* and *cipher suite* negotiated during the handshake.

Since specific (D)TLS versions and cryptographic primitives included in cipher suites lose their security promises from time to time, servers should not rely on deprecated versions and should not offer cipher suites with problematic security primitives [89]. Hence, operators are responsible to keep the configuration of their servers up to date, which is especially challenging with long-life industrial hardware, but essential to uphold security and safety guarantees.

Apart from confidentiality and integrity protection, authentication is an inherent part of secure communication. The exchange of certificates allows both server and client to deny connections for unwanted identities ensuring a reliable access control [15, 49, 65]. While self-signed certificates force operators to manually safelist certificates, CA-signed certificates simplify the key management

and include a public entity in the issuing process (whenever a public CA is used). The use of CAs is common practice on the Web.

However, cryptographic primitives within issued certificates must be kept updated to retain their security promises [31]. Hence, to ensure secure deployments and prevent impersonation attacks, operators need to replace certificates relying on insecure primitives.

Takeaway: With the evolving communication needs in (I)IoT deployments, protocols increasingly adopt security and authentication best practices from the Web. Protocols are either retrofitted with TLS support or secured communication by design. However, available configuration options influence the security of ((I)IoT) appliances. Thus, investigating whether deployments are really secure and conform to today's security recommendations is important.

3 RELATED WORK

Our Internet-wide assessment of the security configuration of TLS-secured industrial deployments is motivated by related work on the security of industrial deployments, assessments of the security of Internet-reachable web services, and the combination of both.

Security of Industrial Deployments: Although the security issues in industrial deployments are well understood [11, 39, 45, 79, 86, 95], and security incidents can be catastrophic [70], to date, they occur comparably seldom [63]. Still, deployments are often insecure: Traditional industrial protocols are heavily used for unprotected communication via the Internet [70]. Additionally, more than 60,000 protocol deployments are connected to the Internet [64], identified as real PLCs [32] and robots [19] not restricting access, enabling everyone on the Internet to control them. Furthermore, the number of exposed deployments increases continuously [101].

Different Internet scan services, e.g., Shodan [92] or Censys [23], collect and share meta-information on Internet-facing services, including industrial deployments [38, 57], often listed quickly after deployment [5], but by far do not see all deployments communicating via the Internet [4]. Still, such meta-information serves as the basis for security assessments [84] of traditional industrial services, finding many affected by known vulnerabilities [10, 36, 52].

Other researchers [7, 27, 46] collect their data via active Internet measurement tools, e.g., ZMap [24]. While such tools typically do not interfere with the operation of industrial equipment [13], malicious attacks can. As researchers expect that the number of attacks against Internet-connected deployments increases [22], and exploits allow enslaving PLCs as network proxy to internal networks using their native programming language [53], securing these Internet-facing deployments is of utmost importance. Thus, the need to rely on secure (I)IoT protocols is unquestionable. However, until now, an analysis on deployments relying on these secure (I)IoT protocols and whether they are configured securely is still missing.

Security-related Internet Measurements: The configuration of security protocols was subject to research basing on active and passive Internet measurements. Especially TLS and PKI usage were in the scope of these works: The TLS and certificate configuration of communication services [42] and Internet-facing embedded devices [16] showed deficits. Furthermore, related work showed that security flaws in key generation [40], fails in Diffie–Hellman [2] and TLS implementations [94] are widespread, as well as analyzed the shift of HTTP servers to TLS 1.3 [44]. Focusing

on certificates of HTTP hosts, different works investigate their configuration [43], validity [12], wrong issuance [55], prevalence in certificate transparency logs [34], and insecurities induced by cross-signing [41].

With a slightly different focus, Internet measurements also cover SSH configurations [35], cloud usage, and communication security of IoT devices [80], as well as compromised IoT devices [60, 71, 88].

Internet-wide Industrial Security Assessments: While the security of deployments in other areas is often widely assessed, the configuration of industrial deployments relying on security features was rarely covered by research. Focusing on the secure-by-design OPC UA protocol, related work showed that operators frequently fail to configure deployments securely [17], possibly due to constrained protocol implementations [26], and presented mechanisms for a more secure device provisioning [54].

Takeaway: While security on the Web increases, Internet measurements have shown that many industrial deployments are connected to the Internet insecurely due to usage of insecure protocols or misconfiguration. However, works assessing industrial appliances left out TLS-based deployments, leaving open whether they improve security.

4 METHODOLOGY AND DATASET

To identify potential attack vectors of (I)IoT deployments relying on TLS to theoretically secure their communication, we select active Internet measurements—a suitable tool to assess such deployments—to study the security of services relying on retrofitted and modern TLS-based (I)IoT protocols. We describe our measurement methodology in Section 4.1 and our resulting dataset in Section 4.2.

4.1 Actively Finding Industrial Deployments

Prior to our Internet scans, we first surveyed relevant protocols.

4.1.1 Selection of Industrial Protocols. The collection of our dataset compiled using active Internet measurements requires the selection of retrofitted industrial and (I)IoT protocols to scan. Our selection process is a three-step approach (which we elaborate in more detail in Appendix A). **First**, we thoroughly analyze several related work in the area of Internet measurements with industrial background [4, 17, 32, 64, 101] and compose a list of 30 traditional industrial and (I)IoT protocols incl. specified standard ports subject in their analyses. **Second**, we employed the IANA Port Number Registry to find corresponding (retrofitted) secure variants. From the 30 detected protocols, we find 18 having an entry in the port number registry and 9 entries indicating a (D)TLS-secured variant: Modbus, DNP3, IEC 104, Siemens S7, EtherNet/IP, AMQP, OPC UA, MQTT, and CoAP. **Third**, we survey the protocol specifications and technical guidelines looking for (D)TLS support and find EtherNet/IP (EnIP) and Tridium Fox (TF; always offered together with Fox Platform (FP)). Overall, we compile a set of ten TLS-securable industrial protocols and their standard port numbers (cf. Table 2 (left)), which we target in our active Internet-wide scans.

4.1.2 Internet-wide Measurements. To find (I)IoT deployments, we scan the complete IPv4 address space for services running on the identified ports. Apart from the TLS-secured protocol variant, we also measure the standard variant (for comparisons). Furthermore, to eliminate short-term influences on our measurements, we

| | | | | *Section 4.2.1 (Validation Process)* | | | *(D)TLS* | | | *Section 4.2.2 (Deployments)* | | *Distinct* | |
Protocol	Port	Variant (Port iff diff.)	Date (2021)	Hosts	Transport	Valid	Auth. OK	Succ.	Valid	Total (opt. TLS)	% (D)TLS	ASes	Cert. CNs
Modbus	502†	Standard TLS	06-28	2,856,848	50,367 (1.8%)	— / 6153 (12%)	6153	6083	30,083 / 0	30,083	0.0%	2285	—
	802‡	TLS	06-27	3,124,856	36,660 (1.2%)	8279 (23%)	8279	8211	0	0 (0)		0	0
DNP3	20000†	Standard TLS	07-19	3,989,523	235,653 (5.9%)	— / 43,134 (18%)	41,697	1836	503 / 3	503	1.0%	91	—
	19999‡	TLS	07-18	3,080,949	242,876 (7.9%)	41,415 (17%)	41,408	536	2	5 (1)		3	3
IEC 104	2404†	Standard TLS	06-08	3,639,366	18,760 (0.5%)	— / 5958 (32%)	5954	5878	4484 / 0	4484	0.0%	383	—
	19998‡	TLS	06-07	3,455,835	139,390 (4.0%)	8999 (6.5%)	8994	8920	1	1 (1)		1	1
EtherNet/IP (EnIP)	44818†	Standard TLS	04-27	3,436,057	5120 (0.1%)	— / 1816 (35%)	1816	1816	1613 / 0	1613	6.4%	282	—
	2221‡	TLS	04-26	4,285,705	6904 (0.2%)	6904 (100%)	6904	6904	111	111 (16)		56	1
Siemens S7	102†	Standard TLS	05-11	2,790,596	25,957 (0.9%)	— / 6125 (24%)	6125	6060	6507 / 0	6507	0.0%	806	—
	3782‡	TLS	05-10	3,383,646	34,958 (1.0%)	5555 (16%)	5555	5497	1	1 (0)		1	1
Tridium Fox (TF)	3011†	Standard (1911)	05-17	4,380,362	32,723 (0.7%)	—			21,995	Standard 23,297	0.5%	1565	—
		Platform			189,620 (4.3%)	—			23,055				
		TLS (4911)			27,598 (0.6%)	11,972 (43%)	11,971	5453	26	TLS 123 (37)		52	55
		Plat.-TLS (5011)			163,820 (3.7%)	142,271 (87%)	142,269	92,529	22				
	4911‡	Standard (1911)	04-18	3,680,729	12,257 (0.3%)	—			4143	Platform 23,447	0.3%	1564	—
		Platform (3011)			19,076 (0.5%)	—			3791				
		TLS			33,788 (0.9%)	12,885 (38%)	12,882	4293	115	Plat.-TLS 82 (21)		31	36
		Plat.-TLS (5011)			31,076 (0.8%)	11,023 (35%)	11,021	3187	70				
AMQP	5672†	Standard TLS	06-14	5,478,838	302,585 (5.5%)	— / 147,280 (49%)	146,901	106,762	130,485 / 6299	130,485	16%	5194	—
	5671‡	TLS	06-13	5,544,602	203,859 (3.7%)	173,531 (85%)	173,464	130,281	19,466	23,908 (12,347)		781	2357
OPC UA	4840†	Binary TLS	07-26	2,781,223	21,991 (0.8%)	— / 5927 (19%)	5921	4681	2193 / 0	2193	0.0%	481	—
	4843‡	TLS	07-24	2,705,968	31,956 (1.2%)	6193 (19%)	6185	4898	0	0 (0)		0	0
MQTT	1883†	Standard TLS	07-06	3,678,615	488,626 (13%)	— / 171,912 (35%)	171,905	27,849	291,344 / 3420	291,344	12%	5285	—
	8883‡	TLS	07-05	4,252,939	632,221 (15%)	540,079 (85%)	540,055	168,617	35,652	39,061 (18,082)		2318	19,068
CoAP	5683†	Standard DTLS	06-22	479,861	425,387 (89%)	— / 3 (0.0%)	3	1	420,780 / 0	420,780	0.0%	2821	—
	5684‡	DTLS	06-21	52,696	4563 (8.7%)	702 (15%)	702	583	76	76 (48)		19	23

(Left section columns under Section 4.1.1 (Selected Protocols): Protocol, Port, Variant (Port iff diff.), Date (2021).)

Table 2: *Left*: Selected protocols and their variants (†standard and ‡secure port), measurement dates (in 2021), and the number of Internet-facing hosts subject to our step-by-step protocol validation process. *Right*: Number and spread of hosts implementing a protocol with and w/o (optional) TLS-support. PubSub protocols have the highest TLS adoption.

repeated them twice and ensured that the results do not differ significantly[1]. Overall, the design and execution of our measurements were subject to comprehensive ethical considerations and best practices for Internet-wide active measurements (cf. Appendix B).

Our Internet measurements consist of three steps: (i) identifying systems having a specific port open, (ii) when measuring (D)TLS-enabled protocols variants, verifying that the server indeed responds with (D)TLS messages, and (iii) validating the application layer protocol. Technically, our measurements rely on zmap [24] to detect Internet-facing systems behind each of the ports, i.e., to find open TCP or UDP ports with replying services. To subsequently retrieve (configuration) data from active systems, we use zgrab2 [102], which we extended with DTLS support (via [78]). Using this setup, we collect detailed information on the (D)TLS configuration of hosts, e.g., their certificates or selected cipher suites. Therefore, we perform four (D)TLS handshakes with each server.

To validate protocol support on the application layer, we extended ZGrab2 with implementations for AMQP (basing on [62, 96]), CoAP, EtherNet/IP, IEC 104, MQTT ([25]), and OPC UA ([14]). Basing on these implementations, we perform protocol-compliant handshakes whenever a (D)TLS connection could be established.

As enabling (D)TLS often is independently configurable from the port [58, 99], we assume to find (D)TLS deployments on the

port reserved for the standard variant. Hence, when scanning ports for standard variants and cannot verify the expected protocol, we also try to perform (D)TLS handshakes analog to the secure port.

4.2 Composing Our Analysis Data

Since servers can provide different services on ports originally reserved for (I)IoT protocols, we must validate all findings. Thus, we next report on how we identified them and labeled their type.

4.2.1 Validating Responses. Table 2 (center) guides through our five-step approach identifying valid protocol deployments, i.e., proving (D)TLS-support and verifying (I)IoT protocol answers. **First**, out of all hosts signaling an open port during our ZMap scan (column *Hosts*), we select systems that do not close the TCP connection immediately and actually respond (*Transport*). Here, we filter out a significant number of IP addresses, which was expected due to background noise that related work reported [42], i.e., up to 1 ‰ of the IPv4 address space reply to SYN packets but do not complete a handshake. During our UDP measurements, a large share of hosts responded with invalid packets, e.g., wrong length fields, which we do not consider. Notably, we detect that ports for the modern protocols AMQP, MQTT, and CoAP have a comparably large fraction of hosts completing a handshake and responding with data, already indicating a larger number of protocol deployments. However, it also comprises port numbers easy to remember (19999 and 20000).

---

[1]Only for MQTT (port 1883), we noticed 160,645 newly reachable brokers in South Korea without TLS support in comparison to our measurement on **2020**-08-31.

The next three steps are only relevant for protocol variants relying on (D)TLS: *Second*, when performing a (D)TLS handshake, we only select hosts that reply with a valid `Server Hello` (column *(D)TLS Valid*). Behind the ports where we expected to find (D)TLS deployments, we identified between 100 % of the responding hosts (EtherNet/IP) and only 6.5 % (IEC 104) as (D)TLS deployments. Interestingly, we also find significant numbers (up to 49 % for AMQP) of (D)TLS hosts behind the protocol's standard ports. These numbers show that operators indeed use ports originally reserved for standard protocol variants for their (D)TLS-enabled deployments.

Third, we focus on hosts not denying our connection due to failed client authentication (column *(D)TLS Auth. OK*). Specifically, we select hosts that do not refuse our connection after receiving our self-signed client certificate, as we would not be able to validate the protocols running on these hosts. Instead, we manually revised the server certificates to get an intuition for the type of running services and protocols. Here, we do not find any subjects inside the certificates indicating industrial use, e.g., 'scada' or well-known industrial manufacturers. However, for the PubSub protocols, we find 22,749 hosts (MQTT) and 15,121 hosts (AMQP) delivering a certificate with accompanying product names, e.g., 'rabbitmq' (AMQP broker implementation), usage description, e.g., 'messaging', or protocol names. On both ports reserved for DNP3 we recognize certificates indicating Orange Livebox router devices in 1334 cases. Since Internet routers typically do not offer a DNP3 interface, they are not pertinent for our analysis. This observation indicates that operators relying on secure-by-design protocols more often make use of security features, although specifications of retrofitted industrial protocols explicitly mention client authentication [50, 66].

While an effective method to prevent unwanted clients, e.g., attackers, to connect—especially in the area of (I)IoT deployments—we find the usage of client authentication comparably low: Only 11 % of all TLS deployments (29,646 hosts) request a client certificate during the handshake. Notably, 7445 of these hosts (25 %) still accept our connection and hence do not validate client certificates.

Fourth, we only select hosts completing the (D)TLS handshake without any error (column *(D)TLS Succ.*). Next to denying a connection due to rejection of our client certificate, we recognized 722,825 hosts aborting the handshake for other reasons without clearly stating why (generic handshake error), e.g., due to an incomplete implementation or an extravagant form of access control.

Finally, and relevant for measurements with and without established (D)TLS channel, we send protocol-conform messages and validate whether the response message of the server is also protocol compliant (column *Valid*), i.e., parsable and having protocol fields, e.g., the length field, valid. In total, we find 967,551 (I)IoT services, of which only 63,368 (6.5 %) are (D)TLS-secured (with a strong focus on modern PubSub protocols). While our measurements of traditional insecure variants show an increase from 2016 [64], they also show that the number of (D)TLS-secured deployments is sparse.

4.2.2 Identifying Single Deployments and Characteristics. For our subsequent security assessment and an indication of its impact, we revise our dataset and describe the characteristics of deployments.

As hosts offering TLS-endpoints with the same configuration on both the standard and TLS port would distort our assessment, we consider such services as one. Furthermore, several operators (especially for PubSub) decide to provide only optional TLS support by offering an insecure and secure endpoint on the respective ports simultaneously (counted as TLS-adopting). Table 2 (right) denotes such behavior (column *Total (opt. (D)TLS)*) and shows the share of deployments relying on (D)TLS (*% (D)TLS*). Here, we identify three groups: (i) protocols with *small TLS-adoption*, i.e., no hosts rely on the TLS variant of Modbus and only fewer than ten deployments on DNP3, IEC 104, Siemens S7, and OPC UA. (ii) We find protocols with a *medium TLS-adoption*, i.e., EtherNet/IP, Tridium Fox, and CoAP, and (iii) *large TLS-adoption* for PubSub (AMQP and MQTT).

Although leaving the choice for secure communication to the client, modern PubSub deployments rely on TLS more often than deployments with traditional protocols and CoAP. Hence, such variants are likely not in use by operators due to deficit configuration, the long lifetime of devices not implementing such new variants, and keeping the resource footprint on constrained devices low.

To analyze the spread of found deployments over the Internet and supervision of different operators, we also report on the distinct number of ASes and Certificate Common Names on TLS-based services in Table 2 (right). While PubSub protocol deployments are generally widely distributed over the Internet and present certificates with many distinct common names, MQTT brokers are more widely distributed than AMQP brokers, where a significant number is hosted by cloud providers, e.g., Microsoft and CloudAMQP. Deployments of protocols with medium TLS-adoption are mainly located in distinct ASes and, for Tridium Fox, Fox Platform, and CoAP, presenting different common names. The single common name of all EtherNet/IP devices backtracks to Rockwell Automation FactoryTalk Linx deployments, indicating that the manufacturer specifies the common name. Lastly, all minor used protocols are test deployments (except for two DNP3 deployments indicating an affiliation with a South Korean company in their certificate).

Takeaway: Only 63,368 *(I)IoT deployments (*6.5 %*) secure their communication, with a focus on employing modern PubSub instead of retrofitted protocols (*7.2 % *vs.* 0.4 %*). Hence, the majority of deployments is open for attacks, jeopardizing a secure and safe operation.*

5 BREAKING DOWN CONFIGURATIONS

Although TLS-adoption is generally improvable in the (I)IoT, examining the security configurations of today's deployments is crucial. To this end, we perform an Internet-wide security assessment. For a benign analysis, we focus on protocols with more than 50 deployments, i.e., we focus on MQTT, AMQP, Tridium Fox and Fox Platform, EtherNet/IP, as well as CoAP. For the detected deployments, we analyze foundational parameters for (D)TLS in Section 5.1, advanced configuration of certificates and PKI in Section 5.2, and security configuration of the protocol (if available) in Section 5.3.

5.1 TLS Foundations

Since (D)TLS is only secure when configured correctly, we assess whether operators set parameters of the (D)TLS deployments sufficiently for secure communication. To this end, we use official guidelines listing adequate TLS versions [28, 68] and cipher suites [89].

5.1.1 Negotiated Protocol Version. TLS deployments should not use any TLS version prior to 1.2 [28], as all of them are nowadays
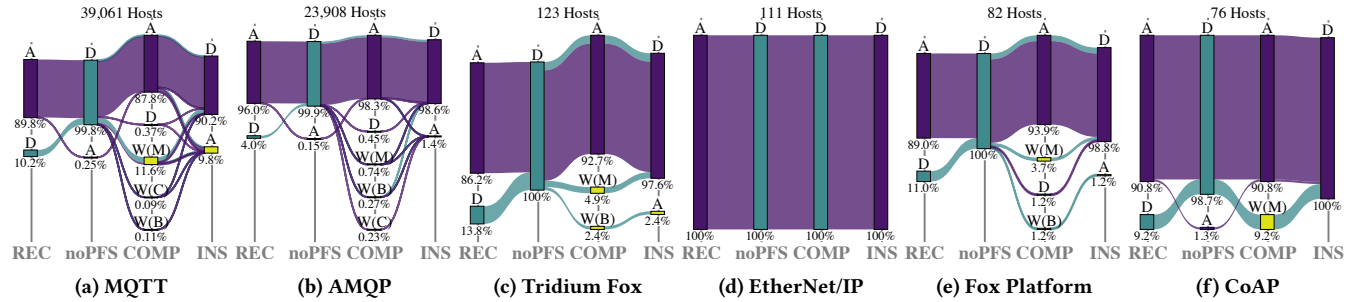
Figure 1: Most hosts act securely (violet) independently of the protocol and cipher suite sets (REC, noPFS, COMP, INS), i.e., accept connections (A) under a secure suite and deny (D) otherwise. Some hosts deny connections of clients not impacting their security (green), and only a few hosts act insecurely (yellow) accepting connections under weak cipher suites (W(•)).

Protocol	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3*
MQTT	252	61	38,748	13,087
AMQP	55	0	23,853	2960
DNP3	2	0	3	2
Tridium Fox	5	0	118	0
Fox Platform	2	0	80	7

* indicated via known byte sequence in the server random number when falling back to TLS 1.2 [81]

Table 3: Most deployments rely on (D)TLS 1.2 (Protocols where all deployments negotiate (D)TLS 1.2 are omitted).

deprecated and insecure due to their reliance on SHA1 [68, 90]. While the majority of industrial deployments use TLS 1.2, we also find deployments falling back to older TLS versions and see deployments already indicating support for the future-proof TLS 1.3 [81].

Table 3 lists the number of protocol deployments relying on a specific TLS version. Altogether, 377 deployments fall back to insecure TLS versions (0.6 %), inherently weakening their communication security by allowing for impersonation attacks. Looking into the certificates' NotBefore dates, we notice that hosts relying on older TLS versions are operated this way for long periods (Mann-Whitney-U test: U=66,690,676; p≈1.05×10^{-180}). Hence, they were not updated in light of changes in the security landscape, i.e., no secure TLS version is enabled. 45 AMQP hosts relying on TLS 1.0, i.e., allowing for impersonation or downgrading attacks, deliver a certificate issued by AppServerGeneratedSBCA (as per certificate's Issuer CA). While we are not entirely sure which software generates these certificates, we assume an association with Microsoft's Windows Service Bus after reviewing websites mentioning the CA's name. As software providing this bus is deprecated, these deployments likely are old and not updated to recent alternatives, which now impacts their security. Contrarily, we note that server software revealing their version (98 % of deployments falling back to TLS version <1.2 provide version information) in all but one case were published after major TLS libraries added TLS 1.2 support. Thus, operators must have set the maximum TLS version manually to a deprecated TLS version, e.g., to reduce compatibility problems with ancient clients. This procedure inherently weakens the communication security with clients that otherwise would choose TLS 1.2.

Deployments indicating support for TLS 1.3 are predominantly located at cloud hosters or in the case of Fox Platform and DNP3 proxied via their infrastructure, covering the findings of related work where the rise of TLS 1.3 was attributed to such providers [44]. These hosts already profit from significant security and performance advantages, e.g., entirely encrypted and 0-RTT handshakes.

5.1.2 Security of Offered Cipher Suites. While a recent TLS version is an important foundation, selecting a secure cipher suite is equally crucial to ensure secure communication. Servers should not select cipher suites with deprecated cryptographic primitives, especially not when clients indicate support for completely secure suites. To analyze this aspect, we performed four subsequent handshakes with servers indicating support for different cipher suites, i.e., **REC** covering cipher suites recommended by official guidelines [31, 61, 89], **noPFS** containing recommended cipher suites to be used when no perfect forward secrecy (PFS) can be implemented (due to hardware limitations; thus enabling attackers to recover information from data encrypted with older key material) [89], **COMP** offering cipher suites the standard ZGrab2 implements for maximum compatibility, and **INS** consisting of insecure ciphers (cf. Appendix C).

Figure 1 shows the share of hosts for every protocol accepting (A) or denying (D) a connection under the cipher suite sets indicated by boxes on respective vertical lines — for MQTT, 90 % accept a connection under REC. For COMP, the figure denotes the reason for the insecurity, i.e., due to the included cipher (W(C)), MAC (W(M)), or both (W(B))). Shapes between the cipher suite sets allow to trace dependencies between host decisions, e.g., which share of hosts accept a connection under COMP but deny a connection under INS — for MQTT, the largest share of hosts accepting a connection under COMP denies a connection under INS. The color of these shapes indicates the host decision of the host fraction on the REC set — for MQTT, most hosts accepting a connection under COMP, but denying under INS, already accepted a connection under REC.

Over all protocols, most hosts accept a connection when clients only offer recommended cipher suites (REC), i.e., such clients can communicate securely. Still, 8.4 % of all servers deny a connection in this setting, i.e., they do not support recommended suites, weakening the communication security. This negligence comprises all but one server using TLS 1.0 and all servers relying on TLS 1.1, showing that outdated configurations also have an impact on the cipher suite selection rendering the communication even more insecure.

Besides, independently of the protocol, most servers deny a connection when clients only request recommended ciphers w/o perfect forward secrecy (noPFS), most likely, as the respective cipher suites are typically excluded from the server's suite set by (OpenSSL's) default. While not negatively influencing the security, resource-constraint (I)IoT clients cannot connect to these servers.

When clients request one of a broad range of suites (COMP), servers accept most connection attempts and predominantly choose
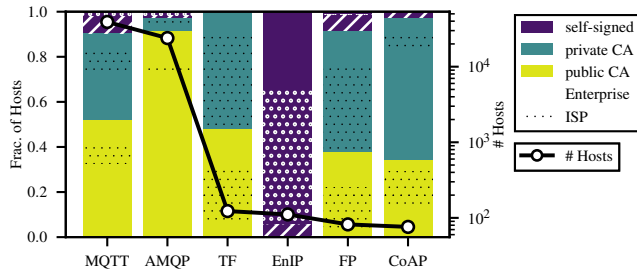
Figure 2: Most certificates are issued by (private) CAs. For PubSub, public CAs are highly frequented by cloud providers and large companies operating enterprise networks.

secure cipher suites. Still, a few hosts of each protocol, except for EtherNet/IP, choose weak cipher suites. Most notably, this also comprises servers previously accepting a connection when only recommended suites are offered by the client, indicating that these servers (i) do not support all recommended cipher suites or prioritize weaker cipher suites over more secure ones, and (ii) are configured for backward compatibility, i.e., not denying connections when clients request no recommended suite. While the selection of a suite relying on a weak MAC enables attackers to alter communication, potentially disturbing production processes, weak ciphers allow for eavesdropping of possibly sensitive data.

Hence, servers should not allow clients to connect via insecure ciphers (INS), and indeed, over all protocols, most servers deny such attempts. However, 4292 servers (6.0 %) still accept insecure connections, significantly weakening the communication security. Notably, for CoAP, Tridium Fox, and Fox Platform, servers accepting a connection with these insecure settings also accepted connections with weak cipher suites when clients offer a mix of recommended and insecure ciphers (COMP). However, CoAP servers also accept connections when clients only indicate support for recommended cipher suites, indicating that these servers do not have valid security policies or are intentionally configured for backward compatibility.

Notably, hosts relying on EtherNet/IP stand out as all of these hosts only accept a cipher from the recommended cipher suite set, i.e., have a very restrictive selection likely induced by their vendor.

Takeaway: 0.6 % of all deployments rely on deprecated TLS versions allowing for impersonation attacks, and 6.0 % of servers allow clients to connect via insecure ciphers easing eavesdropping or alteration of sensitive (I)IoT data. Both result from outdated configurations today impacting the security and safety of affected deployments.

5.2 Authentication Based on Certificates

Another important aspect for secure communication is server authentication relying on certificates and optionally a PKI to prevent attackers from performing Man-in-the-Middle (MitM) attacks. To understand whether operators protect their deployments against MitM attacks, we analyze whether operators make use of PKIs, inspect the age and lifetime of certificates to conclude for implemented strategies for certificate renewal, and whether the cryptographic primitives used to generate certificates are still secure.

5.2.1 Reducing Certificate Management Overhead. To analyze the trust anchor operators rely on (either using self-signed certificates, or utilizing Certificate Authorities (CAs)), we check whether certificates are validatable against root certificates in typical trust

stores[2]. Furthermore, we consider certificates where the issuer and common name are different but not validatable against one of the root certificates in the trust stores as signed by a private CA. We further mark certificates where these fields are equal as self-signed.

Figure 2 shows the trust anchor distribution of hosts split by protocol and indicates whether hosts are located in enterprise networks or networks of Internet Service Providers[3]. All EtherNet/IP servers use self-signed certificates and are mainly located in ISP networks indicating that (i) these hosts do not belong to large companies which would operate own enterprise ASes and (ii) that authentication in automation networks often relies on trust lists manually maintained on every client, increasing the management overhead.

Most Tridium Fox and Fox Platform instances are also located in ISP networks but rely on certificates issued by private CAs. Hence, these small companies issue certificates themselves. While still not involving external entities in the certificate issuance, operators must deploy their own CA root certificate on connecting clients easing the setup of new deployments as no trust lists must be updated.

The majority of MQTT and AMQP brokers as well as CoAP servers rely on certificates issued by public CAs. Here, MQTT brokers are often hosted by cloud providers also taking care of valid certificates as part of their service. In contrast, AMQP brokers relying on certificates issued by public CAs are mostly located in company networks, indicating that operating companies have strong security policies. Notably, 45 % of MQTT and 8.7 % of AMQP deployments profit from certificates issued by CAs supporting automated certificate management, e.g., Let's Encrypt. Hence, automated management allows to decrease the management overhead in the (I)IoT to likely increase authentication security similar to the Web [1].

5.2.2 The Less Overhead the Less Lifetime. Limiting the validity period of certificates can help to improve the security as CA's reverify requesting identities, operators are forced to revisit their used cryptographic primitives, and contain the negative impact of compromised certificates. As general guidelines significantly decreased the recommended lifetime of certificates (generated after June 2016: lifetime ≤ 39 months, after February 2018: ≤ 825 days, after September 2020: ≤ 398 days) [8], the question arises whether operators act accordingly or whether servers present expired certificates.

Figure 3 shows the share of servers with a certain certificate lifetime for different trust anchors. Moreover, it denotes the share of hosts distributing expired certificates. The majority of hosts relying on public CA-signed certificates, independently of the protocol, deliver certificates with a lifetime of two years or less. Thereby, public CAs obey the aforementioned guidelines on certificate lifetimes. However, 3062 hosts rely on expired certificates (between 1.8 % (AMQP) and 38 % (CoAP)), showing that (i) these servers do not benefit from regularly replaced certificates and (ii) clients of such servers do not validate certificate expiry accepting potentially compromised certificates, both allowing for impersonation attacks.

Private CAs do not obey the guidelines on certificate issuance, i.e., 60 % of hosts relying on private CAs send certificates with lifetimes beyond the recommended timespans. Hence, operators do not plan to replace the certificates regularly, potentially affecting their

---

[2]Stores from iOS/MacOS, Windows, Android, OpenJDK, Mozilla NSS, and Oracle JDK.
[3]We rely on the AS's entry in PeeringDB (peeringdb.com) for classification (We count content providers to enterprise and (educational) network services to ISP).
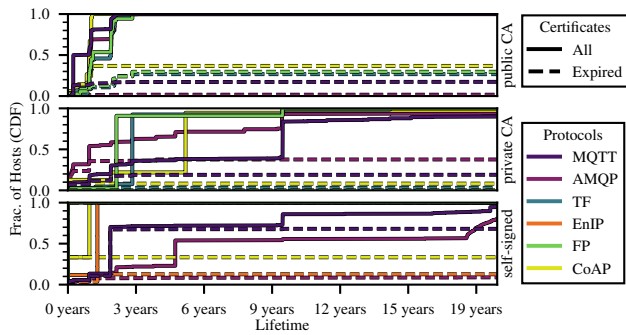
Figure 3: The lifetime of certificates issued by public CAs is significantly shorter, forcing operators to update certificates frequently. Still, many servers deliver expired certificates.

Figure 4: While all certificates issued by public CAs are secure, some others are not. Insecure parameters are hatched.

security on deprecation of used primitives. Furthermore, hosts use already expired private CA-signed certificates (up to 21 % (MQTT)), again showing that clients do not check on certificate expiry.

The lifetime of self-signed certificates is even longer, and certificates are more often expired (EtherNet/IP: 13 %, MQTT: 11 %, CoAP: 100 % of hosts deliver expired self-signed certificates having a lifetime longer than three years). Hence, operators replace certificates less likely when not forced by external entities, which does not necessarily impact the security now, but maybe in the future. Thus, public CAs enforcing update processes increase security.

5.2.3 Most Certificates Are Secure — Currently. While the regular renewal of certificates helps to update the cryptographic primitives certificates rely on, the question is whether operators do so. When operators still rely on certificates with too short public keys, the authentication is at stake independently of the trust anchor. Moreover, deprecated hash functions open doors for impersonation attacks on certificates signed by public or private CAs. Hence, guidelines recommend to not use certificates with an RSA key length lower than 2000 bit, alternatively to RSA, to rely on ECDSA [31, 61], and to not use SHA1 and MD5 for signatures [30].

Figure 4 visualizes the share of hosts delivering certificates with a certain key type and length in the case of RSA, as well as the used hash function separated by their certificate's trust anchor and used (I)IoT protocol. All servers relying on public CA-signed certificates use cryptographic primitives considered as secure. Contrarily, servers relying on certificates issued by private CAs or self-signed certificates use too short keys and deprecated hash functions more frequently. Hence, these systems jeopardize their authenticity.

Altogether, 119 AMQP brokers (3.3 % of private CA authenticated and 11 % of self-authenticated brokers) rely on too short asymmetric keys, opening the door for attackers to eavesdrop and alter their communication. Furthermore, the certificates of 322 brokers (17 % of private CA authenticated brokers) rely on a deprecated hash function for signature generation easing impersonation attacks. Thereby, 31 brokers (2.2 % of private CA authenticated brokers) use certificates still relying on MD5. Similar results are visible for private CA and self-authenticated MQTT brokers, i.e., 344 MQTT brokers (1.4 % of private CA authenticated and 3.6 % of self-authenticated brokers) have a too short key included in their certificate. 11 % of these hosts use certificates generated in 2021, showing that operators still generate certificates with weak parameters.
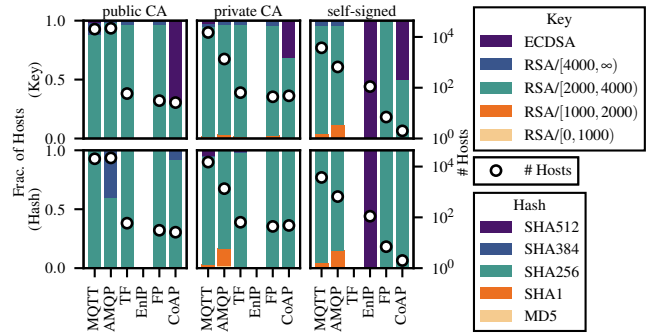
The same holds for deployments using retrofitted protocols, i.e., a minority relies on certificates with weak parameters for authentication not significantly older than strong certificates (25 % of weak certificates were generated in 2021). In contrast, all EtherNet/IP hosts use strongest cryptographic primitives, i.e., ECDSA and SHA512, indicating efficacy of well-applied security policies.

Takeaway: Public CAs force operators to update their certificates regularly, increasing the security of (I)IoT deployments by impeding impersonation. Still, 5.2 % of deployments (not relying on public CAs) use certificates with weak parameters putting their security at stake.

5.2.4 Large-Scale Leakage of Secrets. While using strong cryptographic primitives is a fundamental requirement for secure authentication, the correct handling of cryptographic secrets is even more important. Especially, the private key must not be leaked to any other entity or spread over numerous (I)IoT devices located on the field [17]. Thus, we analyze how many hosts use a single certificate and how widespread they are on the Internet, i.e., over how many autonomous systems (ASes) these hosts are distributed. On this basis, we conclude whether (compromised) certificates are in use.

To this end, we classify certificates into three categories according to how many hosts in how many ASes use them for authentication: (i) We consider certificates in a single AS behind one or two IP addresses (accounting for changes of dynamic IP addresses) as not reused. (ii) Certificates on more than two hosts in a single AS are *intra-AS* reused, and (iii) certificates on more than two hosts in different ASes are *inter-AS* reused. While intra-AS reuse can indicate load-balancing and their affiliation to a single operator, intra-AS reuse likely implies the reuse of key material across operators.

Figure 5 assigns the count of certificates to a number of hosts which use these certificates for authentication and the number of ASes the hosts are located in (nodes increase their size in log-scale). Furthermore, Figure 5 breaks down which share of hosts per protocol is affected by the reuse of certain certificates. While we detected 92 % of the certificates (21,807 of 23,595 certificates) used by hosts behind one or two IP addresses in a single AS, i.e., not reused, we discovered 1260 intra-AS and 528 inter-AS reused.

In both classes of reuse, we detected certificates delivered by AMQP or MQTT brokers indicating load balancing, e.g., hosted by major cloud providers like Amazon. We identified such certificates by their Common Name and Organization field and colored corresponding shares in Figure 5 green. This practice affects 67 certificates reused on in total 10,119 hosts in the same AS, and 12 certificates reused over 8893 hosts in different ASes (up to 5 ASes
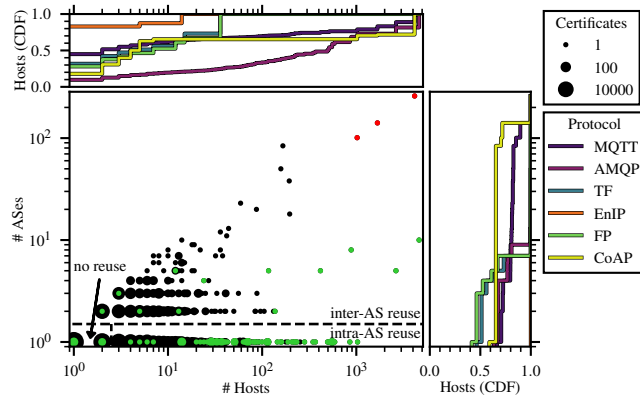
Figure 5: Certificates are reused over different IPs/ASes, indicating load balancing (●) or compromised secrets (● or ●).

for increased availability). As such hosts are typically installed in secure data centers and in control of the same entity, e.g., the cloud provider, these cases of reuse do not negatively impact authenticity.

Contrarily, we found certificates by MQTT and CoAP hosts operated by different entities. The three most reused certificates (red in Figure 5) are used by 6858 MQTT and CoAP hosts distributed over in total 348 ASes indicating that several entities are in possession of the certificate's private key, massively impacting their authenticity. All entities are able to perform impersonation attacks on each other to eavesdrop on confidential information or alter communication. On further investigation, we found these certificates and the corresponding private keys as an example in a GitHub repository of an MQTT broker software with CoAP extension[4]. Furthermore, the repository serves as the basis for a frequently used Docker image providing this functionality, which also includes the certificate and cryptographic secrets. This aspect shows that delivering exemplary cryptographic material inherently instigates operators to weaken their security. We contacted the broker developer via email and issue on GitHub suggesting to remove the example and achieved approval. However, the newest version of the certificate and the private key is still online and still part of the Docker Image.

Regarding (I)IoT protocols offside of modern PubSub protocols, we also identify practices of reuse. For EtherNet/IP, we found a single certificate used by 14 hosts at a smaller scale. After manual inspection of the payload data available on these hosts, we assume that project files, including the certificate and cryptographic secrets, have been copied between devices operating in similar appliances.

Takeaway: *30 % of hosts reuse certificates with (potentially) compromised secrets, e.g., included as examples in software. As partly available on the Internet, these secrets can be used by attackers to perform impersonation attacks to intercept communication.*

5.2.5 Templates to the Rescue? Reusing already available but compromised secrets is easy for operators but also puts authenticity at stake. Hence, ideally, operators receive support in generating cryptographic secrets. Guides providing instructions and predefined scripts can help to determine cryptographic parameters, e.g., a hash function. However, until now, their practical use and impact are unclear. Thus, we shed light on this uncertainty to give directions for best practices in templating and analyze their current influences.
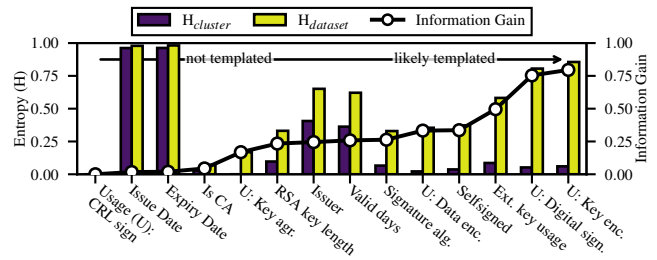
---
[4]https://github.com/emqx/emqx

Figure 6: Templates influence selected usage parameters and signature algorithm but not issue and expiry dates.

A manual revision of all received certificates shows widely distributed similar subject names, although subject names only have a few naming conventions. As this detail indicates a first coherence, we cluster all received certificates by their subject name and subsequently analyze the similarity of other parameters in each cluster. Our clustering relies on text mining approaches, i.e., (1,3)-grams, term frequency-inverse document frequency (TF-IDF), DBSCAN, and a manual review of all calculated clusters (cf. Appendix D).

On the basis of the subject name, we find 547 certificate clusters of up to 207 certificates (mean size: 6, median: 4). The largest cluster comprises certificates with the subject name leading to a bash script in the GitHub repository of OwnTracks, an open-source location tracking software often used in smart homes. Hence, operators also often use templates from the Web to generate their certificates, adopting potentially predefined security preferences. In contrast, the second-largest cluster includes 151 certificates. It can be assigned to a major cloud provider generating certificates for different MQTT broker instances showing that templating is also used in enterprise environments, most likely for fast and automated configuration. Further clusters relate to certificates, including names of large companies active in the communication, virtualization, and IIoT sector. However, we were not able to find any information on these templates on the Web, i.e., the templates used to generate these certificates are most likely company-internal.

Next, we analyze which certificate parameters other than the subject name are influenced by templates. To this end, we compare the entropy of single certificate parameters over the complete dataset with the average normalized entropy weighted by cluster size of each cluster and assume that a lower cluster entropy signalizes that templates set a respective parameter. Figure 6 visualizes both entropies and the gain of similarity of a single parameter within all clusters. Notably, templates specify used cryptographic primitives and their parameters, e.g., the RSA key length or the signature algorithm, more regularly than issuing or expiry dates, having a strong influence on the certificate's security.

To assess the impact of templating on the security of generated certificates, we analyze the occurrence of deprecated parameters in our clusters. Here, we identify 28 of 36 (78 %) detected certificates relying on MD5 signatures in a single cluster, i.e., generated using a single template. Notably, all of these certificates were generated since 2020, showing that old templates weaken the security of today's systems. Moreover, we found three clusters with in total 22 certificates relying on SHA1, implying that operators use three distinct templates still predefining outdated primitives.

Templating is already widely used in practice and helps operators to configure their system fastly and also can have a positive impact on security. However, our analysis shows that templates specifying deprecated parameters, e.g., SHA1 and MD5, are still used and were not updated. While regularly updated templates can influence the security of systems positively, templates with deprecated parameters can put security at stake.

Takeaway: Templates help operators to generate fresh key material and are widely used in practice. However, templates with deprecated settings weaken the security of systems at scale. Hence, operators must use regularly updated templates that take deprecation into account.

5.3 Open Door On PubSub Brokers

For now, our analysis focused on server authentication via their certificates for hosts that do not deny our connection attempt during the TLS handshake. Consequently, if these deployments do not enforce access control on the application layer, they are unprotected against malicious direct access. Especially the 111 hosts relying on the TLS-based variant of EtherNet/IP but also the other seven devices (today mostly test deployments) communicating via the retrofitted variants of S7, Modbus, IEC104, and DNP3 allow espionage or changes of machine behavior. While espionage puts the security of business secrets at stake, changing machine behavior allows attackers to disturb production processes.

For protocols with support for access control, the question arises whether operators make use of these mechanisms or leave their systems open for attackers. However, not all protocols support an analysis of found deployments for properly configured access control mechanisms under our ethical guidelines (cf. Appendix B). As CoAP does not provide any access control mechanism (to achieve a small resource footprint) [91] but leaves this task to downstream applications, no process is available on how to check for such mechanisms. Hence, an analysis would require systematically checking various URLs, leading to a significant overhead on these systems.

Tridium Fox requires access control of remote clients in every case [98]. As we are not aware of any default password, and we do not guess various passwords, we refrain from further analysis. Hence, we focus our analysis of activated access control mechanisms on AMQP, MQTT, and Fox Platform (relying on HTTP).

Fox Platform hosts, independently of using TLS for secure communication or not, reply with either HTTP error codes indicating the servers actively denying the connection, sending back empty responses, or login forms requesting credentials. Hence, we also consider all these devices as secure regarding access control.

Contrarily, several AMQP and MQTT brokers do not implement proper access control, allowing everyone on the Internet (including attackers) to connect and publish/retrieve data via/from these brokers, potentially gaining access to confidential production information and disturbing productions. The default configuration of RabbitMQ, a major AMQP broker implementation, already contains user credentials, initially limited to local network access. To understand operator behavior, i.e., whether operators change predefined credentials when offering services to the Internet, we check whether brokers accept these credentials. Indeed, 13 % of all 142,046 AMQP brokers (14 % of brokers w/o TLS, 1.4 % with TLS) allow access using these credentials. Thus, operators use default credentials

for login via the Internet that are also known by potential attackers. Hence, default configurations should not contain predefined secrets but must be generated for every deployment independently.

For MQTT, 255,857 of 312,323 found brokers (82 %) do not restrict access of any kind. This finding affects only 28 % of TLS-protected brokers but 84 % of non-TLS-protected brokers, indicating that operators more often have access control in mind when simultaneously configuring their communication security appropriately.

We connect to these systems to gain any intuition on the usage of affected systems and search contact information in payload data. Other than for AMQP, where retrieving payload data would disturb ongoing communication (as AMQP brokers do not duplicate messages), on MQTT, we subscribed to the root topic to receive all messages exchanged via the broker (in agreement with our institution's data protection officer, cf. Appendix B). Similar to related work [59], we find exchanged payloads to often cover privacy sensitive types, including feeds of private video surveillance, location services, smart homes, hospitals, and further (I)IoT deployments, underpinning how problematic the availability of such systems is. On these systems, we found 5092 email addresses, which we contacted to responsibly disclose our findings (cf. Appendix B).

Takeaway: 18 % of deployments using TLS for secure communication do not implement access control, i.e., attackers can access the devices to espionage confidential information or alter the system's state. Default configurations used by operators, including pre-configured credentials or disabling authentication, put the systems at risk.

6 DISCUSSION, LIMITATIONS & GUIDELINES

The outcome of our work is two-fold. First, we have seen that our Internet measurements are suitable to identify weaknesses in the configuration of Internet-facing (I)IoT deployments. Future work could look into approach-induced *limitations*, which we report in the following. Second, based on our analysis, we are able to derive *guidelines* to secure the configurations of (I)IoT deployments.

No Influence By Honeypots: Honeypots act like real production systems attracting attackers to perform their exploits. Such systems could, in theory, influence our analysis as operators might configure honeypots less securely. However, the lack of publicly-available honeypot implementations of retrofitted or modern (I)IoT protocols decreases the likelihood of such interference. Still, we manually inspected found systems (with a focus on sparse retrofitted protocol deployments) to find any abnormalities. While we did not find any deployments traceable to TLS-enabled standard honeypots, e.g., Conpot [69], we identified (on top of excluded testing deployments) 19 Tridium Fox devices and 5 EtherNet/IP hosts sharing an—otherwise—unique serial number, respectively. Based on the low number and lack of other anomalies, we conclude that, on a large scale, honeypots do not influence our analysis.

Limitations of Internet Scans: Like any other active Internet scans, we are not able to (i) detect (I)IoT protocols behind hosts denying our connection attempts, (ii) analyze deployments that are only reachable internally or via VPNs, and (iii) find deployments running on non-scanned ports. Thus, we might miss properly-secured hosts in our analysis. Still, the absolute volume of insecure systems is alarming on its own. However, long lifetimes of industrial equipment, generally slow protocol rollouts, the increasing number of

deployments relying on traditional protocols, and our presented results indicate that retrofitted variants are not widely in use today. To answer with certainty whether more properly-secured hosts exist (e.g., behind VPNs), future work should analyze factory, automation, and other (I)IoT networks on site and at large scale.

Guidelines to Improve (I)IoT Security: Our analysis shows that external entities must regularly force operators to revisit their security configurations to increase the provided security level (e.g., CAs mandating operators to re-create certificates). To ease secure configurations, secure-by-design templates are a useful approach, e.g., scripts to automatically generate certificates. Even more important, hardware and software vendors must not include any examples or defaults into their product as operators are otherwise tempted to use them in Internet-facing deployments. Simultaneously, for a secure operation, rigorous maintenance of templates and all security configurations is crucial. Finally, to reliably protect all connections and prevent downgrading attacks, operators may not provide an insecure endpoint simultaneously to a TLS-secured endpoint.

The Calm Before the Storm?: To analyze whether attackers target TLS-based deployments, we deployed seven low interaction honeypots (excluded from our own analysis). We operated our honeypots over six months (2021-01-20 till 2021-07-25) in five ASes providing services on all ports subject to our measurements and perform TLS handshakes on the respective TLS ports. Across all deployments, ports reserved for the secure protocol variant are between 17 % (Fox Platform) and 98 % (Siemens S7) less often subject to connection attempts than the port for the insecure variant, indicating that secure variants are not yet heavily subject to attacks and measurements. While most connection attempts concern modern protocols, operators should also urgently secure their retrofitted protocol deployments as they are still reachable and insecure.

7 CONCLUSION

The convergence of industrial appliances with the Internet requires rolling out end-to-end secure communication and access control to formerly isolated networks [86]. To assess whether secure protocols are indeed adequately used and securely configured in the (I)IoT, we perform active Internet measurements to find deployments relying on ten (I)IoT protocols specified for usage via TLS.

Our results show that TLS adoption in the (I)IoT is low: Only 6.5 % of 967,551 deployments protect their communication. Hence, a large share of deployments still communicates insecurely, enabling attackers to eavesdrop on confidential information or alter potentially safety-critical communication. Furthermore, we find that TLS adoption tends to focus on hosts using modern (I)IoT protocols designed with security in mind (7.2 %) rather than deployments relying on secure variants of traditional industrial protocols (0.4 %). Overall, the evolution towards secure industrial protocols is only barely noticeable, likely due to the long lifetimes of industrial devices.

To support operators in upgrading their systems for secure communication, we assess the security configuration of existing TLS-enabled systems. We reveal that 42 % of (seemingly secure) TLS-enabled deployments (26,665 appliances) are configured insecurely, resulting from outdated protocol versions (0.6 %), ciphers (6.1 %), certificates relying on deprecated primitives (2.2 %), reuse of compromised secrets (30 %), or disabled access control (18 %).

Moreover, we indicate that configuration templates can ease the secure configuration of industrial appliances, e.g., by predefining cryptographic parameters when generating certificates. While already in use and partly responsible for outdated configurations today, e.g., recently issued certificates relying on MD5, we see huge potential in regularly updated templates to assist operators in securely configuring industrial software and hardware.

To conclude, our work shows that the evolution of industrial protocols towards secure end-to-end communication is not widely reflected in real-world deployments. Even worse, when used at all, these protocols are often configured insecurely. To remedy this situation, operators need support in securely configuring (I)IoT deployments, thus requiring approaches to ease the secure configuration of industrial appliances for operators and practitioners.

REFERENCES

[1] Josh Aas et al. 2019. Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web. In *CCS*.
[2] David Adrian et al. 2015. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *CCS*.
[3] Andrew Banks and Rahul Gupta. 2014. MQTT Version 3.1.1. OASIS Standard.
[4] Giovanni Barbieri et al. 2021. Assessing the Use of Insecure ICS Protocols via IXP Network Traffic Analysis. In *ICCCN*.
[5] Roland Bodenheim et al. 2014. Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices. *IJCIP* 7, 2.
[6] Hugh Boyes et al. 2018. The industrial internet of things (IIoT): An analysis framework. *Comput. Ind.* 101.
[7] Olivier Cabana et al. 2019. Detecting, Fingerprinting and Tracking Reconnaissance Campaigns Targeting Industrial Control Systems. In *LNCS*.
[8] CA/Browser Forum. 2021. *Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates*. Version 1.7.8. CA/Browser Forum.
[9] CENELEC. 1996. Building automation and control systems (BACS) – Part 5:Data communication protocol. ISO 16484-5:2017.
[10] Joao M. Ceron et al. 2020. Online Discoverability and Vulnerabilities of ICS/SCADA Devices in the Netherlands. arXiv:2011.02019.
[11] Manuel Cheminod et al. 2013. Review of Security Issues in Industrial Networks. *TII* 9, 1.
[12] Taejoong Chung et al. 2016. Measuring and Applying Invalid SSL Certificates: The Silent Majority. In *IMC*.
[13] Kyle Coffey et al. 2018. Vulnerability Analysis of Network Scanning on SCADA Systems. *Sec. Commun. Netw.* 2018.
[14] COMSYS. 2021 (accessed 11/26/2021). ZGrab 2.0. https://github.com/COMSYS/zgrab2.
[15] ControlNet International and Open DeviceNet Vendor association. 2001. EtherNet/IP Specification – Release 1.0.
[16] Ang Cui and Salvatore J. Stolfo. 2010. A Quantitative Analysis of the Insecurity of Embedded Network Devices: Results of a Wide-Area Scan. In *ACSAC*.
[17] Markus Dahlmanns et al. 2020. Easing the Conscience with OPC UA: An Internet-Wide Study on Insecure Deployments. In *IMC*.
[18] Markus Dahlmanns et al. 2021. Dataset to "Missed Opportunities: Measuring the Untapped TLS Support in the Industrial Internet of Things". https://doi.org/10.18154/10.18154/RWTH-2021-10668
[19] Nicholas DeMarinis et al. 2019. Scanning the Internet for ROS: A View of Security in Robotics Research. In *ICRA*.
[20] David Dittrich and Erin Kenneally. 2012. *The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research*. Technical Report. U.S. Department of Homeland Security.
[21] DNP Users Group. 1993. DNP V3.0 Basic 4 Document Set.
[22] Michael Dodson et al. 2020. When will my PLC support Mirai? The security economics of large-scale attacks against Internet-connected ICS devices. In *eCrime*.
[23] Zakir Durumeric et al. 2015. A Search Engine Backed by Internet-Wide Scanning. In *CCS*.
[24] Zakir Durumeric et al. 2013. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *SEC*.

[25] Eclipse Foundation. 2014 (accessed 07/09/2021). Eclipse Paho MQTT Go client. https://github.com/eclipse/paho.mqtt.golang.
[26] Alessandro Erba et al. 2021. Practical Pitfalls for Security in OPC UA. arXiv:2104.06051.
[27] Claude Fachkha et al. 2017. Internet-scale Probing of CPS: Inference, Characterization and Orchestration Analysis. In NDSS.
[28] Federal Office for Information Security. 2013. ICS Security Compendium. Technical Report. Federal Office for Information Security (BSI).
[29] Federal Office for Information Security. 2017. OPC UA Security Analysis. Technical Report. Federal Office for Information Security (BSI).
[30] Federal Office for Information Security. 2021. Cryptographic Mechanisms: Recommendations and Key Lengths. BSI TR-02102-1.
[31] Federal Office for Information Security. 2021. Cryptographic Mechanisms: Recommendations and Key Lengths: Use of Transport Layer Security (TLS). BSI TR-02102-2.
[32] Xuan Feng et al. 2016. Characterizing industrial control system devices on the Internet. In ICNP.
[33] Brendan Galloway and Gerhard P. Hancke. 2013. Introduction to Industrial Control Networks. IEEE Commun. Surv. Tutor. 15, 2.
[34] Oliver Gasser et al. 2018. In Log We Trust: Revealing Poor Security Practices with Certificate Transparency Logs and Internet Measurements. In LNCS.
[35] Oliver Gasser et al. 2014. A deeper understanding of SSH: Results from Internet-wide scans. In NOMS.
[36] Béla Genge and Călin Enăchescu. 2016. ShoVAT: Shodan-Based Vulnerability Assessment Tool for Internet-Facing Services. Sec. and Commun. Netw. 9, 15.
[37] Robert Godfrey et al. 2012. AMQP Version 1.0. OASIS Standard.
[38] Adam Hansson et al. 2018. Analyzing Internet-connected industrial equipment. In ICSigSys.
[39] Kevin E. Hemsley and Ronald E. Fisher. 2018. History of Industrial Control System Cyber Incidents. Technical Report. Idaho National Laboratory.
[40] Nadia Heninger et al. 2012. Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. In SEC.
[41] Jens Hiller et al. 2020. The Boon and Bane of Cross-Signing: Shedding Light on a Common Practice in Public Key Infrastructures. In CCS.
[42] Ralph Holz et al. 2016. TLS in the Wild: An Internet-wide Analysis of TLS-based Protocols for Electronic Communication. NDSS.
[43] Ralph Holz et al. 2011. The SSL Landscape: A Thorough Analysis of the x.509 PKI Using Active and Passive Measurements. In IMC.
[44] Ralph Holz et al. 2020. Tracking the Deployment of TLS 1.3 on the Web: A Story of Experimentation and Centralization. ACM CCR 50, 3.
[45] Abdulmalik Humayed et al. 2017. Cyber-Physical Systems Security—A Survey. IoTJ 4, 6.
[46] Martin Husák et al. 2018. Assessing Internet-Wide Cyber Situational Awareness of Critical Sectors. In ARES.
[47] IEEE. 2010. IEEE Standard for Electric Power Systems Communications – Distributed Network Protocol (DNP3). IEEE Std. 1815-2010.
[48] International Electrotechnical Commission. 1995. Telecontrol equipment and systems - Part 5: Transmission protocols - Section 101: Companion standard for basic telecontrol tasks. IEC 60870-5-101:1995.
[49] International Electrotechnical Commission. 2000. Telecontrol equipment and systems - Part 5-104: Transmission protocols - Network access for IEC 60870-5-101 using standard transport profiles. IEC 60870-5-104:2000.
[50] International Electrotechnical Commission. 2014. Power systems management and associated information exchange - Data and communications security - Part 3: Communication network and system security - Profiles including TCP/IP. IEC 62351-3:2014.
[51] Wazir Zada Khan et al. 2020. Industrial internet of things: Recent advances, enabling technologies and open challenges. Comput. Electr. Eng. 81.
[52] Timo Kiravuo et al. 2015. Peeking Under the Skirts of a Nation: Finding ICS Vulnerabilities in the Critical Digital Infrastructure. In ECCWS.
[53] Johannes Klick et al. 2015. Internet-facing PLCs as a network backdoor. In CNS.
[54] Florian Kohnhäuser et al. 2021. On the Security of IIoT Deployments: An Investigation of Secure Provisioning Solutions for OPC UA. IEEE Access 9.
[55] Deepak Kumar et al. 2018. Tracking Certificate Misissuance in the Wild. In SP.
[56] Prasanth Lade et al. 2017. Manufacturing Analytics and Industrial Internet of Things. IEEE Intell. Syst. 32, 3.
[57] Éireann P. Leverett. 2011. Quantitatively Assessing and Visualising Industrial System Attack Surfaces. Master's thesis. University of Cambridge.
[58] Roger Light. 2021 (accessed 07/29/2021). mosquitto.conf man page. https://mosquitto.org/man/mosquitto-conf-5.html.
[59] Federico Maggi et al. 2018. The Fragility of Industrial IoT's Data Backbone: Security and Privacy Issues in MQTT and CoAP Protocols. Technical Report. Trend Micro Inc.
[60] Antonio Mangino et al. 2020. Internet-Scale Insecurity of Consumer Internet of Things: An Empirical Measurements Perspective. TIMS 11, 4.
[61] Kerry A. McKay and David A. Cooper. 2019. Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations. NIST SP 800-52 Rev. 2.
[62] Microsoft. 2019 (accessed 07/09/2021). Azure AMQP Go Library. https://github.com/Azure/go-amqp.
[63] Bill Miller and Dale Rowe. 2012. A survey SCADA of and critical infrastructure incidents. In RIIT.
[64] Ariana Mirian et al. 2016. An Internet-wide view of ICS devices. In PST.
[65] Modbus Organization. 2006. MODBUS Messaging Implementation Guide V1.0b.
[66] Modbus Organization. 2018. Modbus/TCP Security Protocol Specification. MB-TCP-Security-v21_2018-07-24.
[67] MODICON, Inc. 1996. Modbus Protocol Reference Guide. PI–MBUS–300 Rev. J.
[68] Kathleen Moriarty and Stephen Farrell. 2021. Deprecating TLS 1.0 and TLS 1.1. IETF RFC 8996.
[69] MushMush. 2021 (accessed 07/29/2021). Conpot. https://github.com/mushorg/conpot.
[70] Marcin Nawrocki et al. 2020. Uncovering Vulnerable Industrial Control Systems from the Internet Core. In NOMS.
[71] Nataliia Neshenko et al. 2019. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. COMST 21, 3.
[72] ODVA. 2015. CIP Networks Specification – Volume 8: CIP Security.
[73] OPC Foundation. 2006. OPC Unified Architecture — Part 1: Concepts. Version 1.00.
[74] OPC Foundation. 2006. OPC Unified Architecture — Part 6: Mappings. Release Candidate 0.91.
[75] OPC Foundation. 2017. OPC Unified Architecture — Part 14: PubSub. OPC 10000-14: OPC Unified Architecture.
[76] OPC Foundation. 2017. OPC Unified Architecture — Part 7: Profiles. OPC 10000-7: OPC Unified Architecture.
[77] Jan Pennekamp et al. 2019. Towards an Infrastructure Enabling the Internet of Production. In ICPS.
[78] Pion. 2018 (accessed 07/09/2021). Pion DTLS. https://github.com/pion/dtls.
[79] Davide Quarta et al. 2017. An Experimental Security Analysis of an Industrial Robot Controller. In SP.
[80] Jingjing Ren et al. 2019. Information Exposure From Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach. In IMC.
[81] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. IETF RFC 8446.
[82] Eric Rescorla and Tim Dierks. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246.
[83] Eric Rescorla and Nagendra Modadugu. 2012. Datagram Transport Layer Security Version 1.2. IETF RFC 6347.
[84] Linus Roepert et al. 2020. Assessing the Security of OPC UA Deployments. In ITSec.
[85] Marshall T. Rose and Dwight E. Cass. 1987. ISO Transport Service on top of the TCP Version: 3. IETF RFC 1006.
[86] Ahmad-Reza Sadeghi et al. 2015. Security and Privacy Challenges in Industrial Internet of Things. In DAC.
[87] Martin Serror et al. 2021. Challenges and Opportunities in Securing the Industrial Internet of Things. TII 17, 5.
[88] Farooq Shaikh et al. 2018. Internet of Malicious Things: Correlating Active and Passive Measurements for Inferring and Characterizing Internet-Scale Unsolicited IoT Devices. ComMag 56, 9.
[89] Yaron Sheffer et al. 2015. Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). IETF RFC 7525.
[90] Yaron Sheffer et al. 2015. Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS). IETF RFC 7457.
[91] Zach Shelby et al. 2014. The Constrained Application Protocol (CoAP). IETF RFC 7252.
[92] Shodan. 2013. Shodan. https://www.shodan.io.
[93] Siemens. 2019. SIMATIC S7-1500, ET 200MP, ET 200SP, ET 200AL, ET 200pro, ET 200eco PN Communication. Function Manual.
[94] Drew Springall et al. 2016. Measuring the Security Harm of TLS Crypto Shortcuts. In IMC.
[95] Ioannis Stellios et al. 2018. A Survey of IoT-Enabled Cyberattacks: Assessing Attack Paths to Critical Infrastructures and Services. COMST 20, 4.
[96] Sean Treadway. 2012 (accessed 07/09/2021). Go RabbitMQ Client Library. https://github.com/streadway/amqp.
[97] Tridium, Inc. 2012. NiagaraAX SSL Connectivity Guide. Technical Document.
[98] Tridium, Inc. 2019. Niagara 4 Hardening Guide. Guide.
[99] VMWare, Inc. 2021 (accessed 07/29/2021). Networking and RabbitMQ. https://www.rabbitmq.com/networking.html.
[100] Konrad Wolsing et al. 2021. IPAL: Breaking up Silos of Protocol-dependent and Domain-specific Industrial Intrusion Detection Systems. arXiv:2111.03438.
[101] Wei Xu et al. 2018. The Landscape of Industrial Control Systems (ICS) Devices on the Internet. In Cyber SA.
[102] ZMap Project. 2021 (accessed 07/30/2021). ZGrab 2.0. https://github.com/zmap/zgrab2.

A PROTOCOL SELECTION

Without any prior knowledge on the deployment of (D)TLS-based (I)IoT protocols, we need to systematically summarize (I)IoT protocols subject to our study. To select TLS-enabled (I)IoT protocols for which we assess deployments during our analysis, we follow a three-step process guided through in Table 4. First, we compiled a list of (I)IoT protocols (without TLS-support) subject in related work focusing on Internet measurements with an industrial background [4, 17, 32, 64, 101]. Here, we found 30 protocols subject to active and passive Internet-wide measurements indicating numerous deployments for many of the protocols.

Second, we inspected IANA port registrations for listed protocols and searched for counterparts indicating security within their name. To this end, we performed an automated full-text search on protocol names registered for standard ports of identified protocols and manually checked for results indicating a secure variant in their name, e.g., TLS, -sec, or secure. From 30 protocols, we found 18 with an entry in the IANA registrations associated with the standard (insecure) port and the industrial protocol. Out of these 18 entries, we identified 11 registrations with an associated secure protocol variant. These registrations also state a secure port to scan during our Internet-wide measurements to find and assess the security of deployments implementing these protocols.

Third, we investigate whether protocol specifications or technical guidelines and manuals of devices implementing these protocols indicate TLS support. While we find eight protocols specified via TLS [3, 37, 47, 50, 50, 66, 72, 76, 91], we also discover indications for TLS support in guidelines for two other protocols, i.e., Tridium Fox and Fox Platform—which is always offered in combination with Tridium Fox [97, 98].

A fundamental foundation for feasible active Internet measurements is a single port per protocol to perform scans on. Hence, we filter out protocols where we are not aware of any (single) standard port which could be subject to our measurements. As we are not able to find evidence for any standard port used by retrofitted BACnet protocol implementations, we refrain from analyzing this retrofitted protocol. Furthermore, we focus our analysis in this paper on (D)TLS-based protocols. Since we find no evidence for Zigbee IP secure relying on (D)TLS, we exclude it from our analysis.

As a result, we obtain a curated list of ten (D)TLS-based (I)IoT protocols subject to our Internet-wide analysis, i.e., we show for these protocols how large the TLS-adoption is and whether the deployments are configured securely.

B ETHICAL CONSIDERATIONS

As measurements of (I)IoT systems could have unintended implications, e.g., concerning information security, privacy, or safety, we take several ethical considerations as the basis for our research.

We follow widely recognized ethical research guidelines [20] as well as practices and procedures imposed by our institutions during the design, execution, and analysis of our research. Therefore, we handle all collected data with care as well as responsibly contact operators of systems not implementing access control, whenever contact information is available (cf. Appendix B.1). Furthermore, we adhere to accepted measurement guidelines [24] to reduce the impact of our measurements (cf. Appendix B.2).

B.1 Handling of Data & Responsibilities

During our measurements, in agreement with our institution's data protection officer, we only request publicly available data. Specifically, we never request any payload data when protected by access control mechanisms. Besides for AMQP, where we check whether deployments accept a pair of default credentials, we never bypass any security mechanisms. Most importantly, we immediately close connections to AMQP servers accepting the default credentials. Furthermore, we never alter the server's state, i.e., we never send any write or function execution requests to the servers. Still, our dataset might contain sensitive data of servers that do not implement access control (cf. Section 5.3). We store all data on secured systems to keep potentially included sensitive data private and prevent attackers from finding open or insecure systems by using our dataset. For the public release of our dataset, we removed retrieved payload data and replaced all identifiers, e.g., IP addresses and subject names in certificates, by consecutive numbers. While this restriction prevents others from independently reproducing our certificate templating results, we consider this decision to constitute a reasonable trade-off to protect affected users.

Responsible Disclosure: We analyzed the received data to identify server operators to inform them about their accessible systems whenever possible, i.e., we automatically searched the received payload data of MQTT brokers for email addresses. We found 5092 email addresses on 372 MQTT brokers as contact information. To reduce the overhead of our information campaign, we filtered out email addresses using a domain without any DNS MX record behind. Finally, we reached out to 2985 email addresses pointing out potential privacy, security, and safety issues. Although we received 877 automatically generated responses indicating that these collected email addresses did not exist, we received nine responses within one day. While two operators indicate that affected systems are test systems, the seven others report that they secured their systems after our message or intend to secure them soon. All in all, we received very positive feedback on our information campaign.

Unfortunately, other protocols subject to our study do not allow performing similar information campaigns. Other than MQTT, AMQP does not allow duplicating data on request, i.e., data would be lost when our scanner requests it, and none of the legitimate clients would receive the data. We refrain from requesting payload data because it is against our ethical considerations to change a system's state and alter ongoing communication. Furthermore, traditional (I)IoT protocols with and w/o TLS only allow to request single production values but do not provide any function to request contact information.

B.2 Reducing Impact of Measurements

To minimize the implication of our active Internet measurements, we follow well-established Internet measurement guidelines [24].

Measurement Coordination: We coordinate all of our measurements with our institutional Network Operation Center to reduce the impact on the Internet. Specifically, we ensure to answer and handle inquiries or abuse requests as fast as possible.

External Perception: We display the research intent of our scans to external operators by providing rDNS records for our scanning IP address and transmitting contact information in our client

Protocol	Related Work	Standard Port	IANA		(D)TLS mentioned		Secure Port
			Assigned	Secure Variant	Specification	Technical Guidelines	
BACnet	[4, 32, 64, 101]	47808	bacnet	—	[9]	—	—
DNP3	[4, 32, 64, 101]	20000	dnp	dnp-sec	[47, 50]	—	19999
Modbus	[4, 32, 64, 101]	502	mbap	mbap-s	[66]	—	802
Siemens S7	[4, 32, 64, 101]	102	iso-tsap	iso-tp0s	—	[93]	3782
TridiumFox	[4, 32, 64, 101]	1911	mtp*	—	—	[97, 98]	4911
Ethernet/IP	[4, 32, 64]	44818	EtherNet-IP-2	ethernet-ip-s	[72]	—	2221
HART-IP	[4, 32, 64]	5094	hart-ip	—	—	—	—
OPC UA	[4, 17]	4840	opcua-tcp	opcua-tls	[76]	—	4843
Automatic Tank Gauge	[4, 32]	10001	scp-config*	—	—	—	—
CodeSys	[4, 32]	2455	wago-io-system	—	—	—	—
General Electric SRTP	[4, 32]	18245−18246	—	—	—	—	—
IEC 104	[4, 32]	2404	iec-104	iec-104-sec	[50]	—	19998
Melsec-Q	[4, 32]	5006−5007	wsm-server*	—	—	—	—
OMRON FINS	[4, 32]	9600	micromuse-ncpw*	—	—	—	—
PC Worx	[4, 32]	1962	biap-mp*	—	—	—	—
ProConOS	[4, 32]	20547	—	—	—	—	—
Red Lion Crimson V3	[4, 32]	789	—	—	—	—	—
ANSI C12.22	[4, 64]	1153	c1222-acse	—	—	—	—
ICCP	[4, 64]	102	iso-tsap*	—	—	—	—
IEC 61850	[4, 64]	102	iso-tsap*	—	—	—	—
AMQP	[4]	5672	amqp	amqps	[37]	—	5671
ATG	[4]	10001	scp-config*	—	—	—	—
CoAP	[4]	5683	coap	coaps	[91]	—	5684
EtherCAT	[4]	34980	ethercat	—	—	—	—
FF HSE	[4]	1089−1091	ff-{annunc,fms,sm}	—	—	—	—
FL-net	[4]	55000−55003	—	—	—	—	—
MQTT	[4]	1883	mqtt	secure-mqtt	[3]	—	8883
PROFINET	[4]	34962−34964	profinet-{rt,rtm,cm}	—	—	—	—
Zigbee IP	[4]	11754−11755	zep, zigbee-ip	zigbee-ips	—	—	11756
CSPV4	[32]	2222	EtherNet-IP-1	ethernet-ip-s	—	—	2221

* IANA registration unrelated to (I)IoT protcol.   Protocols marked in grey are subject to our analysis.

Table 4: (I)IoT protocols subject in related work undergoing our selection process.

certificate as well as protocol messages where possible. Additionally, we provide a website behind our scanning IP address with detailed information on the scope and purpose of our research. Furthermore, we list opt-out instructions on the website to request exclusion of our scans. Based on such requests (also regarding other scanning projects of our institute), we exclude 5.8 M IP addresses (0.14% of the IPv4 address space).

Limiting Load: To not overload any autonomous system, we spread our scans over a timeframe of approximately 24 hours per protocol and rely on zmap's address randomization. More importantly, to not overload potentially (I)IoT devices during our subsequent TLS handshakes, we instruct our scanner module to wait 15 min between subsequent handshakes to one server. For MQTT, we further set a scanning time (30 min) and outgoing traffic (10 MB) limit per host, i.e., our scanner disconnects whenever the limit exceeds.

While (I)IoT protocols do not realize security by default, we consider it essential to know whether today's deployments benefit from recently introduced security features. To answer this question, we have taken sensible measures to reduce the risks introduced by active Internet measurements of industrial appliances, aiming to positively influence the security of (I)IoT deployments.

C (D)TLS CIPHER SUITE SETS

To reduce the load on deployments subject to our measurements, we refrained from performing a single (D)TLS handshake per possible cipher suite to analyze server support. Instead, we curate four sets of (D)TLS cipher suites and analyze which suite servers choose from each set.

Table 5 lists our four sets, i.e., *recommended*, *recommended without perfect forward secrecy*, *compatibility*, and *insecure*, as well as reports on the assigned (D)TLS ciphers.

Recommended: To check whether servers implement (at least one) secure and recommended cipher suite, we collect suites following basic recommendations and guidelines [31, 61, 89]. Servers denying a connection when our scanner presents this cipher suite set most likely do not implement any recommended suite indicating insecure or outdated configuration. While fundamental for secure communication, all the ciphers included in this set implement perfect forward secrecy.

Recommended w/o perfect forward secrecy: However, especially constrained (I)IoT devices might not be able to implement perfect forward secrecy. Hence, we also check whether deployments support less resource-intensive, still secure, but non-perfect forward secret cipher suites. We base the choice of ciphers in this set also on basic guidelines [31, 61, 89]. While still secure, disclosure of cryptographic key material used for communication allows attackers to decrypt all the communicated data.

Compatibility: Implementing a large variety of different combinations of ciphers and MAC algorithms, the standard ZGrab2 set has very high compatibility. The cipher suites include RSA and ECDSA key exchange mechanisms in combination with AES and RC4 ciphers in GCM or CBC mode, as well as MAC algorithms relying on SHA1 or SHA256. Thus, this set encompasses secure and weak cipher suites, depending on the included mechanism. Hence, we use this set to evaluate the security policy of deployments, i.e., assess whether deployments choose one of the secure or one of the insecure cipher suites.

Recommended: *Cipher suites recommended by recent guidelines.*

ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	DHE_DSS_WITH_AES_128_GCM_SHA256	ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE_RSA_WITH_AES_256_GCM_SHA384	ECDHE_ECDSA_WITH_AES_128_CCM	ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
DHE_RSA_WITH_AES_256_GCM_SHA384	DHE_RSA_WITH_AES_128_CCM	ECDHE_ECDSA_WITH_AES_128_CBC_SHA
DHE_DSS_WITH_AES_256_GCM_SHA384	ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	ECDHE_ECDSA_WITH_AES_256_CCM_8
ECDHE_ECDSA_WITH_AES_256_CCM	ECDHE_RSA_WITH_AES_256_CBC_SHA384	DHE_RSA_WITH_AES_256_CCM_8
DHE_RSA_WITH_AES_256_CCM	DHE_RSA_WITH_AES_256_CBC_SHA256	ECDHE_ECDSA_WITH_AES_128_CCM_8
ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	DHE_DSS_WITH_AES_256_CBC_SHA256	DHE_RSA_WITH_AES_128_CCM_8
ECDHE_RSA_WITH_AES_128_GCM_SHA256	DHE_RSA_WITH_AES_128_CBC_SHA256	ECDHE_ECDSA_WITH_AES_256_CBC_SHA
DHE_RSA_WITH_AES_128_GCM_SHA256	DHE_DSS_WITH_AES_128_CBC_SHA256	

Recommended w/o perfect forward secrecy: *Cipher suites recommended by recent guidelines when no PFS can be implemented.*

ECDH_ECDSA_WITH_AES_256_GCM_SHA384	DH_RSA_WITH_AES_128_GCM_SHA256	ECDH_ECDSA_WITH_AES_128_CBC_SHA256
ECDH_RSA_WITH_AES_256_GCM_SHA384	DH_DSS_WITH_AES_128_GCM_SHA256	ECDH_RSA_WITH_AES_128_CBC_SHA256
DH_RSA_WITH_AES_256_GCM_SHA384	ECDH_ECDSA_WITH_AES_256_CBC_SHA384	DH_RSA_WITH_AES_128_CBC_SHA256
DH_DSS_WITH_AES_256_GCM_SHA384	ECDH_RSA_WITH_AES_256_CBC_SHA384	DH_DSS_WITH_AES_128_CBC_SHA256
ECDH_ECDSA_WITH_AES_128_GCM_SHA256	DH_RSA_WITH_AES_256_CBC_SHA256	
ECDH_RSA_WITH_AES_128_GCM_SHA256	DH_DSS_WITH_AES_256_CBC_SHA256	

Compatibility: *Cipher suites used by default ZGrab2 for maximum compatibility.*

ECDHE_RSA_WITH_AES_128_GCM_SHA256	ECDHE_ECDSA_WITH_AES_128_CBC_SHA	RSA_WITH_AES_256_CBC_SHA
ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ECDHE_ECDSA_WITH_AES_256_CBC_SHA	ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
ECDHE_RSA_WITH_RC4_128_SHA	ECDHE_ECDSA_WITH_AES_256_CBC_SHA	RSA_WITH_3DES_EDE_CBC_SHA
ECDHE_ECDSA_WITH_RC4_128_SHA	RSA_WITH_RC4_128_SHA	
ECDHE_RSA_WITH_AES_128_CBC_SHA	RSA_WITH_AES_128_CBC_SHA	

Insecure: *Deprecated and insecure ciphersuites.*

NULL_WITH_NULL_NULL	DH_ANON_EXPORT_WITH_DES40_CBC_SHA	ECDH_ANON_WITH_RC4_128_SHA
RSA_WITH_NULL_MD5	DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA	RSA_WITH_DES_CBC_SHA
RSA_WITH_NULL_SHA	RSA_EXPORT1024_WITH_DES_CBC_SHA	DH_DSS_WITH_DES_CBC_SHA
RSA_WITH_NULL_SHA256	DH_ANON_EXPORT_WITH_RC4_40_MD5	DH_RSA_WITH_DES_CBC_SHA
ECDH_ECDSA_WITH_NULL_SHA	RSA_EXPORT1024_WITH_RC4_56_MD5	DHE_DSS_WITH_DES_CBC_SHA
ECDHE_ECDSA_WITH_NULL_SHA	RSA_EXPORT1024_WITH_RC4_56_SHA	DHE_RSA_WITH_DES_CBC_SHA
ECDH_RSA_WITH_NULL_SHA	DHE_DSS_EXPORT1024_WITH_RC4_56_SHA	DH_ANON_WITH_DES_CBC_SHA
ECDHE_RSA_WITH_NULL_SHA	RSA_EXPORT_WITH_RC4_40_MD5	DH_ANON_WITH_3DES_EDE_CBC_SHA
ECDH_ANON_WITH_NULL_SHA	RSA_WITH_RC4_128_MD5	ECDH_ANON_WITH_3DES_EDE_CBC_SHA
RSA_EXPORT_WITH_RC2_CBC_40_MD5	RSA_WITH_RC4_128_SHA	DH_ANON_WITH_AES_128_CBC_SHA256
RSA_EXPORT1024_WITH_RC2_CBC_56_MD5	DH_ANON_WITH_RC4_128_MD5	DH_ANON_WITH_AES_256_CBC_SHA256
RSA_EXPORT_WITH_DES40_CBC_SHA	DH_DSS_WITH_RC4_128_SHA	DH_ANON_WITH_AES_128_GCM_SHA256
DH_RSA_EXPORT_WITH_DES40_CBC_SHA	ECDH_ECDSA_WITH_RC4_128_SHA	DH_ANON_WITH_AES_256_GCM_SHA384
DH_DSS_EXPORT_WITH_DES40_CBC_SHA	ECDHE_ECDSA_WITH_RC4_128_SHA	ECDH_ANON_WITH_AES_128_CBC_SHA
DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	ECDH_RSA_WITH_RC4_128_SHA	ECDH_ANON_WITH_AES_256_CBC_SHA
DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	ECDHE_RSA_WITH_RC4_128_SHA	

Table 5: Cipher suite sets we use to analyze whether servers support cipher suites implementing different levels of security.

Insecure: Finally, we curate a list of cipher suites, including insecure cryptographic primitives, e.g., DES, MD5, and NULL encryption, to check whether servers allow clients to connect without a proper level of security. To this end, we manually selected cipher suites including unrecommended primitives (according to [30]) from the IANA registration list, which are officially standardized. Whenever servers select a cipher out of this set, it enables clients to communicate insecurely.

D CERTIFICATE CLUSTERING

To research whether operators use configuration templates predefining preferences when generating certificates, we cluster certificates and analyze whether the certificates in each cluster were likely generated using the same template.

We chose the certificate's subject name to find coherent certificates after a manual revision of our dataset showed many certificates distributed over different hosts all over the Internet using similar naming schemes. However, the name typically is a free text field with only a few conventions. To find related subject names, we calculate a similarity matrix based on methods from text mining and subsequently create clusters using DBSCAN. Since the subject name includes insignificant keys in front of every value and is comprised in every certificate, e.g., CN=, we first vectorize the subject name using single character (1,3)-grams and apply term frequency-inverse document frequency (TF-IDF) building one vector per term. This vector indicates the term's distribution in the subject name of all certificates (the more subject names include a term, the smaller its TF-IDF). Subsequently, using a sparse cosine similarity metric, we calculate a similarity matrix including the 500 most similar TF-IDF vectors (with a threshold of 0.5 to make the calculation feasible, directly performing DBSCAN over the vectors of all terms would be infeasible).

To build clusters of certificates most likely generated with the same template, we perform DBSCAN on the similarity matrix (with $\epsilon = 0.8$, cosine similarity as a metric, and a minimum cluster size of three). Here, we manually revisited the results to ensure that no obvious false positives were included.