TLS-Attacker: A Dynamic Framework for Analyzing TLS Implementations

Fabian Bäumer
Ruhr University Bochum
Bochum, Germany
fabian.baeumer@rub.de

Marcus Brinkmann
Ruhr University Bochum
Bochum, Germany
marcus.brinkmann@rub.de

Nurullah Erinola
Ruhr University Bochum
Bochum, Germany
nurullah.erinola@rub.de

Sven Hebrok
Paderborn University
Paderborn, Germany
sven.hebrok@uni-paderborn.de

Nico Heitmann
Paderborn University
Paderborn, Germany
nico.heitmann@uni-paderborn.de

Felix Lange
Paderborn University
Paderborn, Germany
felix.lange@uni-paderborn.de

Marcel Maehren
Ruhr University Bochum
Bochum, Germany
marcel.maehren@rub.de

Robert Merget
Technology Innovation Institute
Abu Dhabi, United Arab Emirates
robert.merget@tii.ae

Niklas Niere
Paderborn University
Paderborn, Germany
niklas.niere@uni-paderborn.de

Maximilian Radoy
Paderborn University
Paderborn, Germany
maximilian.radoy@uni-paderborn.de

Conrad Schmidt
Hackmanit GmbH
Bochum, Germany
conrad.schmidt@hackmanit.de

Jörg Schwenk
Ruhr University Bochum
Bochum, Germany
joerg.schwenk@rub.de

Juraj Somorovsky
Paderborn University
Paderborn, Germany
juraj.somorovsky@uni-paderborn.de

Abstract

TLS-Attacker is an open-source framework for analyzing Transport Layer Security (TLS) implementations. The framework allows users to specify custom protocol flows and provides modification hooks to manipulate message contents. Since its initial publication in 2016 by Juraj Somorovsky, TLS-Attacker has been used in numerous studies published at well-established conferences and helped to identify vulnerabilities in well-known open-source TLS libraries. To enable automated analyses, TLS-Attacker has grown into a suite of projects, each designed as a building block that can be applied to facilitate various analysis methodologies. The framework still undergoes continuous improvements with feature extensions, such as DTLS 1.3 or the addition of new dialects such as QUIC, to continue its effectiveness and relevancy as a security analysis framework.

Keywords

SSL, TLS, DTLS, Protocol State Fuzzing, Planning Based

ACM Reference Format:
Fabian Bäumer, Marcus Brinkmann, Nurullah Erinola, Sven Hebrok, Nico Heitmann, Felix Lange, Marcel Maehren, Robert Merget, Niklas Niere, Maximilian Radoy, Conrad Schmidt, Jörg Schwenk, and Juraj Somorovsky. 2024. TLS-Attacker: A Dynamic Framework for Analyzing TLS Implementations. In *Proceedings of Cybersecurity Artifacts Competition and Impact Award (ACSAC '24).* ACM, New York, NY, USA, 6 pages.

ACSAC '24, December 09–13, 2024, Waikiki, Hawaii
2024.

1 Introduction

Transport Layer Security (TLS) is arguably the most important cryptographic protocol. It provides authenticity, integrity, and confidentiality to any application-level protocol and thus can be used to secure communication to web, email, or FTP servers. The protocol has a long history, with the first developments starting in the 90s. Back then, it was initially developed as the Secure Sockets Layer (SSL) protocol. The protocol was adopted by the IETF and, in 1999, released as the TLS 1.0 standard [2]. Since then, TLS 1.1 [12], 1.2 [32], and 1.3 [31] were released. Along these standards, various protocol extensions were defined, which introduced new cryptographic primitives, features, or even completely new messages.

The high importance of TLS also attracted the security research community. Most notably, many new attacks were released at the beginning of the 2010s. In the TLS community, this era is also referenced as the golden age of TLS attacks. The attacks exploited the complexity of TLS and various TLS extension specifications and affected the protocol in different attacker models. For example, Rizzo and Duong showed how to exploit TLS compression for their CRIME attacks [33]. Alfardan and Paterson presented Lucky13, exploiting tiny timing side channels resulting from the padding used in the MAC-then-pad-then-encrypt scheme [1]. Beurdouche et al., and de Ruiter and Poll showed how TLS state machine implementations can be forced to process invalid messages, leading to state machine violations [6, 11]. Probably the most visible attack was, however, Heartbleed [25]. Heartbleed showed how a buffer overread vulnerability in a single implementation can lead to severe consequences throughout the entire TLS ecosystem, such as exfiltrating server private keys. These attacks triggered the research community to improve TLS implementations and search for TLS vulnerabilities in a systematic way.

The development in the area of TLS attacks also motivated our research and culminated in the development of TLS-Attacker [38]. By developing TLS-Attacker, we considered the developments of TLS attacks at that time and used them as the basis for defining the requirements for a flexible TLS evaluation. Most importantly, we observed that to implement TLS attacks and detect new ones, the framework has to cover more than cryptographic attacks (cf. Figure 1). First, the framework should allow for flexible protocol message modifications potentially triggering state machine vulnerabilities [6, 11]. This includes dynamically adding and removing custom protocol messages without negatively affecting the internal state. Second, the framework should be able to dynamically change custom cryptographic primitives and their underlying values. Such behavior is required, for example, to modify padding before performing the encryption of application messages [1]. Third, the framework should be able to trigger and detect vulnerabilities resulting from buffer overflows and overreads.

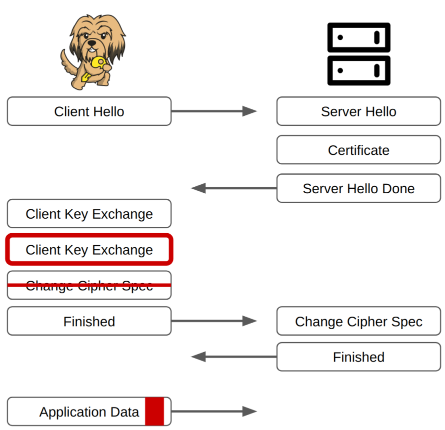

Figure 1: An illustration of a modified TLS handshake. Modifications, such as adding, removing, and manipulating message contents, are highlighted in red.

The TLS-Attacker project was first published alongside the paper "Systematic Fuzzing and Testing of TLS Libraries" by Juraj Somorovsky in 2016 [38] and released under the Apache 2.0 License as an open-source repository on GitHub.[1] At its core, TLS-Attacker implements the TLS protocol, supplying client and server functionalities. To achieve the requirements resulting from the related attacks, TLS-Attacker implements two mechanisms: *Workflow Traces* and *Modifiable Variables*. A Workflow Trace defines the protocol flow of a TLS session on a high level. It contains a series of messages TLS-Attacker sends to the analyzed TLS connection peer. Modifiable Variables can be integrated into a Workflow Trace to apply further nuanced modifications to individual message fields. The modifications can be applied at different stages, for example, before the computed values are encrypted.

Listing 1 shows an example of a protocol flow triggering the Heartbleed bug. The code implements a custom protocol flow, resulting in a crafted Heartbeat message being sent. The message contains a dynamic modification that manipulates its internal length field. The resulting message will claim to contain 2,000 bytes, while its actual content is significantly shorter. A vulnerable implementation would fail to verify the integrity of the length field and proceed to read 2,000 bytes from its memory to echo the contents back to the client. Defining the same message flow and manipulation is also possible through XML instead of Java code.

```
Config config = Config.createConfig();
WorkflowTrace trace = new WorkflowTrace();
trace.addTlsAction(new SendAction(new ClientHelloMessage()));
ServerHelloDoneMessage helloDone
    = new ServerHelloDoneMessage();
trace.addTlsAction(new ReceiveTillAction(helloDone));
trace.addTlsAction(new SendAction(new ClientKeyExchange()));
HeartbeatMessage heartbeat = new HeartbeatMessage();
heartbeat.setLength(Modifiable.explicit(2000));
trace.addTlsAction(new SendAction(heartbeat));
trace.addTlsAction(new ReceiveAction(new HeartbeatMessage()));
State state = new State(config, trace);
DefaultWorkflowExecutor executor
    = new DefaultWorkflowExecutor(state);
executor.executeWorkflow();
```

Listing 1: A protocol flow triggering the Heartbleed bug

Development. Since its release in 2016, TLS-Attacker has been continuously expanded to cover a range of protocol versions, from SSL 3.0 to TLS 1.3, as well as DTLS 1.0 and DTLS 1.2. The project further implements numerous TLS extensions and more than 330 cipher suites, including uncommon GOST and SM cipher suites specified by the Russian and Chinese authorities. More than 70 people, including researchers, students, and pen testers from the community, have contributed to TLS-Attacker. Since large-scale studies require automated tests rather than individual workflow traces, the TLS-Scanner project leverages the flexibility of TLS-Attacker to evaluate clients and servers of TLS libraries. It provides various *probes* to identify supported protocol versions and features and to test for known vulnerabilities. Figure 2 gives an overview of the main projects of the TLS-Attacker suite.

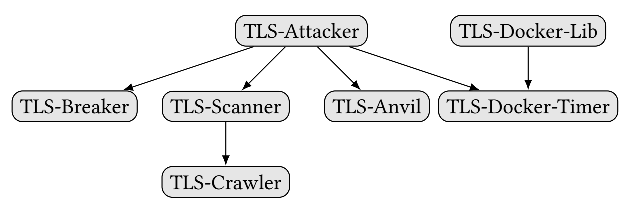

Figure 2: Overview of projects within the TLS-Attacker suite. All projects are accessible at https://github.com/tls-attacker/.

Maintenance. Currently, TLS-Attacker and its subprojects are actively maintained and developed by the Ruhr University Bochum (RUB), the Paderborn University (UPB), the Technology Innovation Institute (TII), and the Hackmanit GmbH.

[1] https://github.com/tls-attacker/TLS-Attacker

2 Impact

Academic Perspective. TLS-Attacker has been used in various studies. The use cases range from lab evaluations to IPv4-wide scans. Table 1 provides an overview of the publications that utilized TLS-Attacker. Below, we briefly describe the scope of the papers and how TLS-Attacker[2] was used in these works.

- In 2016, Juraj Somorovsky presented TLS-Attacker as an open-source TLS analysis framework [38]. Using fuzzing to create messages with TLS-Attacker, Somorovsky identified padding oracle vulnerabilities and buffer overflows in TLS libraries such as Botan and MatrixSSL.
- In 2017, Bozic et al. presented a planning-based test approach for TLS libraries [8]. Their work utilized TLS-Attacker to create TLS messages sent in pre-specified orders to conduct tests. Furthermore, Xian et al. used and extended TLS-Attack to show that side-channel attacks on encrypted communication to trusted CPU enclaves are possible [41]. Simos et al. built upon TLS-Attacker to create a framework for combinatorial testing of TLS servers [36]. The same year, Böck et al. presented the ROBOT attack [7], proving that Bleichenbacher oracle vulnerabilities still exist in modern libraries. They extended the TLS-Attacker framework with the ability to better identify vulnerable libraries.
- In 2018, Engelbertz et al. conducted a study on the security of eID endpoints [15]. They used TLS-Attacker to evaluate the TLS features supported by these endpoints and to conduct vulnerability tests. Also, in 2018, Simos et al. extended their research in combinatorial testing of TLS servers with even more capabilities for their TLS-Attacker-based tool.
- In 2019, Merget et al. systematically studied padding oracles in TLS based on a scan of publicly deployed hosts on the Internet [29]. They used TLS-Attacker to send messages with manipulated padding at different points in the TLS session. In the same year, Garn et al. applied combinatorial input sequence generation to fingerprint web browsers [21]. They utilized TLS-Attacker's server implementation to send malformed sequences of TLS messages, aiming to reveal unique response patterns. Furthermore, Calzavara et al. presented a quantitative security evaluation of TLS configurations from the Alexa Top 10k using TLS-Attacker as one of their evaluation tools [10].
- In 2020, Fiterau-Brostean et al. presented a study of eleven DTLS libraries based on protocol state fuzzing [17], in which they used TLS-Attacker to generate and parse protocol messages.
- In 2021, Brinkmann et al. present a study of cross-protocol attacks against different application protocols that utilize TLS with shared certificates [9]. In their work, they used TLS-Attacker to test which certificates are accepted by web browsers. In the same year, Merget et al. presented an attack on the Diffie-Hellman (DH) key exchange as used in TLS versions up to 1.2 [28]. They used TLS-Attacker to conduct timing measurements that revealed side-channel vulnerabilities, which enable an attacker to deduce the most significant bits of a DH shared secret. In the same year, Drees et al. built upon TLS-Attacker, creating a way to automatically scan for new side-channel attacks using machine learning [13]. Henn et al. analyzed German health websites [24] using TLS-Scanner as one of the considered test tools. Fu et al. [20] explored machine learning to detect malicious traffic. They used TLS-Scanner's vulnerability tests as part of their datasets.
- In 2022, McMahon Stone et al. [27] expanded upon the previous protocol state fuzzing studies by employing a gray-box approach with an extended input alphabet. Again, TLS-Attacker was used to construct and parse messages. In the same year, Maehren et al. [26] conducted a study applying combinatorial testing to TLS libraries. They used TLS-Attacker to generate test templates that evaluate the RFC compliance of TLS libraries for varying session parameters. Fiterau-Brostean et al. presented *DTLS-Fuzzer* [19], a DTLS protocol state fuzzing tool based on TLS-Attacker. Asadian et al. explored symbolic execution to test specification compliance of TLS implementations [3]. They used TLS-Attacker and the DTLS-Fuzzer mentioned above to generate test cases. Saatjohann et al. [34] used TLS-Scanner's vulnerability tests to analyze medical devices and hospital IT infrastructure. Garn et al. expanded upon previous work in the field of TLS-based web browser fingerprinting [22]. They used TLS-Attacker to send the fingerprinting message sequences.
- In 2023, Wu et al. studied the security of VPNs deployed at academic institutions [40]. They used TLS-Attacker to assess the server configuration of TLS-based VPN endpoints. In the same year, Hebrok et al. [23] presented a study of weaknesses in the TLS session ticket mechanism caused by insecure key generation. They used TLS-Attacker to collect session tickets in different protocol versions and test whether invalid session resumption attempts were rejected correctly. Erinola et al. further presented a study on the DTLS ecosystem [16]. Here, they used TLS-Attacker to evaluate the deployment of DTLS protocol features and to identify denial of service vulnerabilities. Berbecaru et al. proposed *TLS-Monitor* [5], a network monitoring tool for vulnerability detection. They used TLS-Attacker to build a testbed for the evaluation of their tool. Wang et al. proposed a testing framework for 5G network components [39] and used TLS-Attacker to verify potential findings. Fiterau-Brostean et al. presented an automated analysis of DTLS state machines based on model checking [18]. As in previous works, TLS-Attacker was used to send and parse the messages used to infer the state machine. Scott used TLS-Anvil [35] to compare a newly proposed TLS library to well-known open-source TLS libraries.
- In 2024, Dunsche et al. [14] presented a study of timing side-channel vulnerabilities in open-source TLS libraries. Here, TLS-Attacker was used to build various types of attack vectors to allow for side-channel measurements. Berbecaru et al. presented an intrusion detection tool called *Threat-TLS* [4]. Among other tools, they used TLS-Attacker to verify potential vulnerabilities in deployed hosts. Radoy et al. presented a study on Partitioning Oracles in TLS session ticket handling [30], using TLS-Attacker to collect and manipulate session tickets.

[2] Note that we use 'TLS-Attacker' to refer to all projects based on TLS-Attacker here.

Year	Venue	Author(s)	Paper
'16	CCS	Somorovsky	[38]
'17	CCS	Xiao et al.	[41]
'17	ICST	Simos et al.	[36]
'17	ICSTW	Bozic et al.	[8]
'18	(ePrint)	Böck et al.	[7]
'18	SQJ†	Simos et al.	[37]
'18	WOOT	Engelbertz et al.	[15]
'19	HotSoS	Garn et al.	[21]
'19	S&P	Calzavara et al.	[10]
'19	USENIX Security	Merget et al.	[29]
'20	USENIX Security	Fiterau-Brostean et al.	[17]
'21	(ePrint)	Drees et al.	[13]
'21	CCS	Fu et al.	[20]
'21	Computers & security	Garn et al.	[22]
'21	JDMIBE‡	Henn et al.	[24]
'21	USENIX Security	Merget et al.	[28]
'21	USENIX Security	Brinkmann et al.	[9]
'22	CCS	McMahon Stone et al.	[27]
'22	DuD	Saatjohann et al.	[34]
'22	ICST	Asadian et al.	[3]
'22	ICST	Fiterau-Brostean et al.	[19]
'22	USENIX Security	Maehren et al.	[26]
'23	(ePrint)	Scott	[35]
'23	ACNS	Wang et al.	[39]
'23	CCNC	Berbecaru et al.	[5]
'23	NDSS	Fiterau-Brostean et al.	[18]
'23	USENIX Security	Hebrok et al.	[23]
'23	USENIX Security	Erinola et al.	[16]
'23	USENIX Security	Wu et al.	[40]
'24	ARES	Berbecaru et al.	[4]
'24	ESORICS	Radoy et al.	[30]
'24	USENIX Security	Dunsche et al.	[14]

†: Software Quality Journal
‡: Annual Conference of the German Society for Medical Informatics, Biometry and Epidemiology

Table 1: Overview of academic papers using TLS-Attacker. Papers from independent authors are highlighted in gray.

Industry Projects. The TLS-Attacker project suite has been used in the following projects:

- In a study focusing on the security of the OpenSSL library, commissioned by the German Federal Office of Information Security (BSI project 154)[3].
- In a project contributing to the development of the Botan TLS library, commissioned by the German Federal Office of Information Security (BSI project 197)[4].
- In SIWECOs, a project aiming to help small and medium-sized businesses estimate the security of their websites and content management systems [5].
- In Future Trust [6], a project focusing on the security of eID and electronic signature services.
- In KoTeBi, a project focusing on the development of a TLS test suite for the analysis of RFC compliance [7] called TLS-Anvil [8]. The test suite is a finalist of the IT Security Award (IT-Sicherheitspreis 2024 [9]) funded by the Horst Görtz Stiftung.
- In the Botan TLS-library [10]. Botan uses TLS-Anvil in its automated testing environment [11].

3 Outlook

TLS is used in combination with various protocols, including HTTP, FTP, and SMTP, to provide confidentiality, integrity, and authentication for different use cases. To fully assess the security of a TLS-based application, an analysis of the interaction between TLS and the application protocol is required to cover each protocol's unique attack surface. With the lessons learned from building, extending, and maintaining TLS-Attacker, we are currently striving towards other security protocols like QUIC or SSH. We aim to develop a universal analysis framework that is extendable with TLS-independent protocols by reusing the techniques and tools created in the past years and only implementing protocol-specific components. By providing such a universal analysis framework, we aim to facilitate further and better research of cryptographic protocols and reduce the manual workload of real-world evaluations.

Acknowledgments

We thank all contributors of TLS-Attacker and its subprojects. Niklas Niere, Felix Lange, Fabian Bäumer, Marcel Maehren, Conrad Schmidt, and Nurullah Erinola were supported by the German Federal Ministry of Education and Research (BMBF) through the project KoTeBi (16KIS1556K, 16KIS1559). Sven Hebrok and Nico Heitmann were supported by the research project "North-Rhine Westphalian Experts in Research on Digitalization (NERD II)", sponsored by the state of North Rhine-Westphalia – NERD II 005-2201-0014. Marcus Brinkmann was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

References

[1] Nadhem J. Al Fardan and Kenneth G. Paterson. 2013. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In *2013 IEEE Symposium on Security and Privacy* (San Francisco, CA, USA). IEEE Computer Society, Los Alamitos, CA, USA, 526–540. https://doi.org/10.1109/SP.2013.42

[2] Christopher Allen and Tim Dierks. 1999. The TLS Protocol Version 1.0. RFC 2246. https://doi.org/10.17487/RFC2246

[3] Hooman Asadian, Paul Fiterău-Broştean, Bengt Jonsson, and Konstantinos Sagonas. 2022. Applying Symbolic Execution to Test Implementations of a Network Protocol Against its Specification. In *2022 IEEE Conference on Software Testing, Verification and Validation (ICST)* (Valencia, Spain). IEEE Computer Society, Los Alamitos, CA, USA, 70–81. https://doi.org/10.1109/ICST53961.2022.00019

[4] Diana Gratiela Berbecaru and Antonio Lioy. 2024. Threat-TLS: A Tool for Threat Identification in Weak, Malicious, or Suspicious TLS Connections. In *Proceedings of the 19th International Conference on Availability, Reliability and Security* (Vienna, Austria) *(ARES '24)*. Association for Computing Machinery, New York, NY, USA, Article 125, 9 pages. https://doi.org/10.1145/3664476.3670945

[5] Diana Gratiela Berbecaru and Giuseppe Petraglia. 2023. TLS-Monitor: A Monitor for TLS Attacks. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)* (Las Vegas, NV, USA). IEEE Computer Society, Los Alamitos, CA, USA, 1–6. https://doi.org/10.1109/CCNC51644.2023.10059989

[6] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. 2015. A Messy State of the Union: Taming the Composite State Machines of TLS. In *2015 IEEE Symposium on Security and Privacy* (San Jose, CA, USA). IEEE Computer Society, Los Alamitos, CA, USA, 535–552. https://doi.org/10.1109/SP.2015.39

[7] Hanno Böck, Juraj Somorovsky, and Craig Young. 2018. Return Of Bleichenbacher's Oracle Threat (ROBOT). In *27th USENIX Security Symposium (USENIX*

[3] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/OpenSSL-Bibliothek/DokumentationOpenSSL.pdf
[4] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Krypto/Projektzusammenfassung_Botan.pdf
[5] https://siwecos.de/en/
[6] https://pilots.futuretrust.eu/
[7] https://www.kotebi.de/en/
[8] https://github.com/tls-attacker/TLS-Anvil
[9] https://www.deutscher-it-sicherheitspreis.de/
[10] https://botan.randombit.net/
[11] https://github.com/randombit/botan/blob/master/.github/workflows/nightly.yml#L146

Security 18) (Baltimore, MD, USA). USENIX Association, Berkeley, CA, USA, 817–849. https://www.usenix.org/conference/usenixsecurity18/presentation/bock

[8] Josip Bozic, Kristoffer Kleine, Dimitris E. Simos, and Franz Wotawa. 2017. Planning-Based Security Testing of the SSL/TLS Protocol. In *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (Tokyo, Japan). IEEE Computer Society, Los Alamitos, CA, USA, 347–355. https://doi.org/10.1109/ICSTW.2017.63

[9] Marcus Brinkmann, Christian Dresen, Robert Merget, Damian Poddebniak, Jens Müller, Juraj Somorovsky, Jörg Schwenk, and Sebastian Schinzel. 2021. ALPACA: Application Layer Protocol Confusion - Analyzing and Mitigating Cracks in TLS Authentication. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Berkeley, CA, USA, 4293–4310. https://www.usenix.org/conference/usenixsecurity21/presentation/brinkmann

[10] Stefano Calzavara, Riccardo Focardi, Matus Nemec, Alvise Rabitti, and Marco Squarcina. 2019. Postcards from the Post-HTTP World: Amplification of HTTPS Vulnerabilities in the Web Ecosystem. In *2019 IEEE Symposium on Security and Privacy (SP)* (San Francisco, CA, USA). IEEE Computer Society, Los Alamitos, CA, USA, 281–298. https://doi.org/10.1109/SP.2019.00053

[11] Joeri de Ruiter and Erik Poll. 2015. Protocol State Fuzzing of TLS Implementations. In *24th USENIX Security Symposium (USENIX Security 15)* (Washington, D.C., USA). USENIX Association, Berkeley, CA, USA, 193–206. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/de-ruiter

[12] Tim Dierks and Eric Rescorla. 2006. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346. https://doi.org/10.17487/RFC4346

[13] Jan Peter Drees, Pritha Gupta, Eyke Hüllermeier, Tibor Jager, Alexander Konze, Claudia Priesterjahn, Arunselvan Ramaswamy, and Juraj Somorovsky. 2021. Automated Detection of Side Channels in Cryptographic Protocols: DROWN the ROBOTs!. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security* (Virtual Event, Republic of Korea) *(AISec '21)*. Association for Computing Machinery, New York, NY, USA, 169–180. https://doi.org/10.1145/3474369.3486868

[14] Martin Dunsche, Marcel Maehren, Nurullah Erinola, Robert Merget, Nicolai Bissantz, Juraj Somorovsky, and Jörg Schwenk. 2024. With Great Power Come Great Side Channels: Statistical Timing Side-Channel Analyses with Bounded Type-1 Errors. In *33rd USENIX Security Symposium (USENIX Security 24)* (Philadelphia, PA, USA). USENIX Association, Berkeley, CA, USA, 6687–6704. https://www.usenix.org/conference/usenixsecurity24/presentation/dunsche

[15] Nils Engelbertz, Nurullah Erinola, David Herring, Juraj Somorovsky, Vladislav Mladenov, and Jörg Schwenk. 2018. Security Analysis of eIDAS – The Cross-Country Authentication Scheme in Europe. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)* (Baltimore, MD, USA). USENIX Association, Berkeley, CA, USA. https://www.usenix.org/conference/woot18/presentation/engelbertz

[16] Nurullah Erinola, Marcel Maehren, Robert Merget, Juraj Somorovsky, and Jörg Schwenk. 2023. Exploring the Unknown DTLS Universe: Analysis of the DTLS Server Ecosystem on the Internet. In *32nd USENIX Security Symposium (USENIX Security 23)* (Anaheim, CA, USA). USENIX Association, Berkeley, CA, USA, 4859–4876. https://www.usenix.org/conference/usenixsecurity23/presentation/erinola

[17] Paul Fiterău-Broştean, Bengt Jonsson, Robert Merget, Joeri de Ruiter, Konstantinos Sagonas, and Juraj Somorovsky. 2020. Analysis of DTLS Implementations Using Protocol State Fuzzing. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Berkeley, CA, USA, 2523–2540. https://www.usenix.org/conference/usenixsecurity20/presentation/FiterÄČu-BroÅ§tean

[18] Paul Fiterău-Broştean, Bengt Jonsson, Konstantinos Sagonas, and Fredrik Tåquist. 2023. Automata-Based Automated Detection of State Machine Bugs in Protocol Implementations. In *30th Annual Network and Distributed System Security Symposium (NDSS 2023)* (San Diego, CA, USA). The Internet Society, Reston, VA, USA. https://www.ndss-symposium.org/ndss-paper/automata-based-automated-detection-of-state-machine-bugs-in-protocol-implementations/

[19] Paul Fiterău-Broştean, Bengt Jonsson, Konstantinos Sagonas, and Fredrik Tåquist. 2022. DTLS-Fuzzer: A DTLS Protocol State Fuzzer. In *2022 IEEE Conference on Software Testing, Verification and Validation (ICST)* (Valencia, Spain). IEEE Computer Society, Los Alamitos, CA, USA, 456–458. https://doi.org/10.1109/ICST53961.2022.00051

[20] Chuanpu Fu, Qi Li, Meng Shen, and Ke Xu. 2021. Realtime Robust Malicious Traffic Detection via Frequency Domain Analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 3431–3446. https://doi.org/10.1145/3460120.3484585

[21] Bernhard Garn, Dimitris E. Simos, Stefan Zauner, Rick Kuhn, and Raghu Kacker. 2019. Browser fingerprinting using combinatorial sequence testing. In *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security* (Nashville, TN, USA) *(HotSoS '19)*. Association for Computing Machinery, New York, NY, USA, Article 7, 9 pages. https://doi.org/10.1145/3314058.3314062

[22] Bernhard Garn, Stefan Zauner, Dimitris E. Simos, Manuel Leithner, Richard Kuhn, and Raghu Kacker. 2022. A Two-Step TLS-Based Browser fingerprinting approach using combinatorial sequences. *Computers & Security* 114 (2022), 102575. https://doi.org/10.1016/j.cose.2021.102575

[23] Sven Hebrok, Simon Nachtigall, Marcel Maehren, Nurullah Erinola, Robert Merget, Juraj Somorovsky, and Jörg Schwenk. 2023. We Really Need to Talk About Session Tickets: A Large-Scale Analysis of Cryptographic Dangers with TLS Session Tickets. In *32nd USENIX Security Symposium (USENIX Security 23)* (Anaheim, CA, USA). USENIX Association, Berkeley, CA, USA, 4877–4894. https://www.usenix.org/conference/usenixsecurity23/presentation/hebrok

[24] Frederic Henn, Richard Zowalla, and Andreas Mayer. 2021. The Security State of the German Health Web: An Exploratory Study. *Studies in Health Technology and Informatics* 283 (Sept. 2021), 180–185.

[25] Riku Hietamäki, Antti Karjalainen, Matti Kamunen, and Neel Mehta. 2014. CVE-2014-0160. Retrieved November 20, 2024 from https://heartbleed.com/

[26] Marcel Maehren, Philipp Nieting, Sven Hebrok, Robert Merget, Juraj Somorovsky, and Jörg Schwenk. 2022. TLS-Anvil: Adapting Combinatorial Testing for TLS Libraries. In *31st USENIX Security Symposium (USENIX Security 22)* (Boston, MA, USA). USENIX Association, Berkeley, CA, USA, 215–232. https://www.usenix.org/conference/usenixsecurity22/presentation/maehren

[27] Chris McMahon Stone, Sam L. Thomas, Mathy Vanhoef, James Henderson, Nicolas Bailluet, and Tom Chothia. 2022. The Closer You Look, The More You Learn: A Grey-box Approach to Protocol State Machine Learning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 2265–2278. https://doi.org/10.1145/3548606.3559365

[28] Robert Merget, Marcus Brinkmann, Nimrod Aviram, Juraj Somorovsky, Johannes Mittmann, and Jörg Schwenk. 2021. Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E). In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Berkeley, CA, USA, 213–230. https://www.usenix.org/conference/usenixsecurity21/presentation/merget

[29] Robert Merget, Juraj Somorovsky, Nimrod Aviram, Craig Young, Janis Fliegenschmidt, Jörg Schwenk, and Yuval Shavitt. 2019. Scalable Scanning and Automatic Classification of TLS Padding Oracle Vulnerabilities. In *28th USENIX Security Symposium (USENIX Security 19)* (Santa Clara, CA, USA). USENIX Association, Berkeley, CA, USA, 1029–1046. https://www.usenix.org/conference/usenixsecurity19/presentation/merget

[30] Maximilian Radoy, Sven Hebrok, and Juraj Somorovsky. 2024. In Search of Partitioning Oracle Attacks Against TLS Session Tickets. In *Computer Security – ESORICS 2024*, Joaquin Garcia-Alfaro, Rafał Kozik, Michał Choraś, and Sokratis Katsikas (Eds.). Springer Nature Switzerland, Cham, Switzerland, 320–340.

[31] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. https://doi.org/10.17487/RFC8446

[32] Eric Rescorla and Tim Dierks. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. https://doi.org/10.17487/RFC5246

[33] Juliano Rizzo and Thai Duong. 2012. The CRIME attack. Retrieved November 20, 2024 from https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU

[34] Christoph Saatjohann, Fabian Ising, Matthias Gierlings, Dominik Noss, Sascha Schimmler, Alexander Klemm, Leif Grundmann, Tilman Frosch, and Sebastian Schinzel. 2022. Sicherheit medizintechnischer Protokolle im Krankenhaus. In *GI SICHERHEIT 2022*. Gesellschaft für Informatik, Bonn, Germany, 143–158. https://doi.org/10.18420/sicherheit2022_09

[35] Michael Scott. 2023. On TLS for the Internet of Things, in a Post Quantum world. Cryptology ePrint Archive, Paper 2023/095. https://eprint.iacr.org/2023/095

[36] Dimitris E. Simos, Josip Bozic, Feng Duan, Bernhard Garn, Kristoffer Kleine, Yu Lei, and Franz Wotawa. 2017. Testing TLS Using Combinatorial Methods and Execution Framework. In *Testing Software and Systems*, Nina Yevtushenko, Ana Rosa Cavalli, and Hüsnü Yenigün (Eds.). Springer International Publishing, Cham, Switzerland, 162–177.

[37] Dimitris E Simos, Josip Bozic, Bernhard Garn, Manuel Leithner, Feng Duan, Kristoffer Kleine, Yu Lei, and Franz Wotawa. 2019. Testing TLS using planning-based combinatorial methods and execution framework. *Software Quality Journal* 27, 2 (jun 2019), 703–729.

[38] Juraj Somorovsky. 2016. Systematic Fuzzing and Testing of TLS Libraries. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 1492–1504. https://doi.org/10.1145/2976749.2978411

[39] Yong Wang, Rui Wang, Donglan Liu, Hao Zhang, Lei Ma, Fangzhe Zhang, Lili Sun, and Zhenghao Li. 2023. A Framework for TLS Implementation Vulnerability Testing in 5G. In *Applied Cryptography and Network Security Workshops*, Jianying Zhou, Lejla Batina, Zengpeng Li, Jingqiang Lin, Eleonora Losiouk, Suryadipta Majumdar, Daisuke Mashima, Weizhi Meng, Stjepan Picek, Mohammad Ashiqur Rahman, Jun Shao, Masaki Shimaoka, Ezekiel Soremekun, Chunhua Su, Je Sen Teh, Aleksei Udovenko, Cong Wang, Leo Zhang, and Yury Zhauniarovich (Eds.). Springer Nature Switzerland, Cham, Switzerland, 284–298.

[40] Ka Lok Wu, Man Hong Hue, Ngai Man Poon, Kin Man Leung, Wai Yin Po, Kin Ting Wong, Sze Ho Hui, and Sze Yiu Chau. 2023. Back to School: On the (In)Security of Academic VPNs. In *32nd USENIX Security Symposium (USENIX Security 23)* (Anaheim, CA, USA). USENIX Association, Berkeley, CA, USA, 5737–5754. https://www.usenix.org/conference/usenixsecurity23/presentation/wu-ka-lok

[41] Yuan Xiao, Mengyuan Li, Sanchuan Chen, and Yinqian Zhang. 2017. STACCO: Differentially Analyzing Side-Channel Traces for Detecting SSL/TLS Vulnerabilities in Secure Enclaves. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, TX, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 859–874. https://doi.org/10.1145/3133956.3134016