Evaluation of TLS/SSL Implementations on Raspberry Pi for Secure IoT Communication

LALANDE LUCAS LEV MICHEL, University of Twente, The Netherlands
SUPERVISOR: DR. ING. MOHAMMAD ELHAJJ, University of Twente, The Netherlands

Transport layer security (TLS)/Secure socket layer (SSL) is a security protocol implemented on the internet aiming to provide communications that respect the Confidentiality, Integrity, Availability (CIA) triad . The Internet of Things (IoT) is a growing industry that requires encrypted communications to respect the consumer's privacy. IoT devices are devices with computing capabilities but that are restricted in computing power and memory. Several research report that TLS/SSL is too energy consuming and has too much overhead which makes the connections inefficient on those devices. TLS/SSL is defined by the Internet Engineering Task Force (IETF) a security protocol that can be modified with several parameters , the Cipher suite, the transport Protocol and handhsake optimisations such as session resumption and Pre-Shared Key (PSK). It is known that the the Cipher Suite can be modified by choosing cryptography functions with a low computational complexity such as curves from the Elliptic Curve Cryptography (ECC) family. Another aspect TLS/SSL depends on is the transport protocol. Two interesting are TCP and UDP protocol that are by design oriented towards a connection-oriented oriented protocol and connection-less protocol respectively, there is no consensus on which is faster in terms of bandwidth while the handshake process can also be altered for instance using PSK or session resumption. One last aspect that has been investigated in research are performance improvement over the software to accelerate computing processes. Each of these step can be optimised. Previous research indicate that as is TLS/SSL is not suitable for IoT devices as is.This calls for a comprehensive study that assess whether this claim is true for a setup that is heavily optimised towards embedded systems. The goal of this research is to assess optimisations and their influence on speed of connection for the TLS handshake as well as throughput and in throughput for the hash functions. Secondly a security analysis must be carried to assess possible dangerous setups that could be vulnerable to attacks such as TLS downgrade, Man In The Middle (MITM) and vulnerabilities exploiting holes in the TLS 1.0 to 1.3 protocols and cipher suites. This comprehensive study has for objective to provide information on the speed of different TLS setups and their security properties in the context of IoT devices. The following section provides more context on the necessity of this study and introduces the main research question

Additional Key Words and Phrases: TLS, SSL, CIA, IoT, TCP, UDP, PSK, MITM, IP, AEAD, ARP, IETF, ECC

1 INTRODUCTION

Society is increasingly including products integrating computers and micro controllers within their household, those devices when connected to the internet are also known as the IoT . The amount of IoT devices has grown by 18% in 2022 and is forecasted to have a growth of at least 15% from 2023 as mentioned in [1]. The information transmitted by smart devices must be kept secure to forbid malicious attackers from eavesdropping and breaching the privacy of users. **Transport Layer Security (TLS)** and **Secure socket layer (SSL)** are protocols that are designed to guarantee security over the internet and guarantee the CIA triad. TLS is an upgraded version of SSL historically and the terms TLS/SSL are used interchangeably [2]. TLS/SSL is a security protocol that runs over a transmission protocol, some transmission protocols are TCP/IP, UDP ,both protocols have the same functionality which is transmitting packets over the network but are implemented and designed differently , specifically TLS is designed for reliable direct communications and is widely used on the internet some applications are web-browsing, email and text messaging . As for UDP it is designed for real-time data transmission and is designed for speed. It is generally considered that UDP is faster than TCP in terms of bandwith however TCP and UDP have been compared in [3] and in the specific scenario they compare UDP is indeed faster. In the context of this research the DTLS 1.3 protocol is investigated, which is TLS over UDP. The study will be using the terms DTLS for TLS over UDP and TLS for TLS over TCP. TLS has been proposed in 1999 by the IETF (Internet Engineering Task Force) and the first version of the protocol has been published in 1999 [4]. It is a communication protocol that consists of a handshake with multiple steps that depend on the TLS version. The standard nowadays is TLS version 1.3. TLS guarantees the following properties for a communication **Encryption, Authentication, Integrity** [4]. Researchers have been trying to achieve fast and secure communications but the Status Quo is that the best-established standard which is TCP/SSL is not efficient and consumes too much power, in [5] the results demonstrate that encrypted communications consume 10 times more the number of bytes transmitted per minute is also diminished when TLS/SSL is activated. A concerning issue is that the state-of-the-art protocol TLS/SSL for encrypted communications is considered secure but has been proven to not be optimal for constrained devices due to its energy consumption, high complexity and high overhead [5]. TLS is made of several components,[6] enumerates key concepts of TLS, the relevant ones are **Handshake, Cipher suite, Certificate, and Session**. Handshake describes the protocol of messages that must be exchanged in for a secure connection to be established and can differ based on the implementation. The cipher suite describes the combination of an encryption, a hashing, and a key exchange algorithm that will be used during the TLS protocol as mentioned in [7]. It is then possible to trade security for faster hashing execution by choosing algorithms that are less computationally expensive, many are available but a promising direction are ECC algorithms. RSA and ECC have been compared in [8] and the result conclude that ECC is more performant than RSA regarding operational efficiency and security. The Certificate is a document that contains a public key and is signed by a CA(Certificate Authority) which is a mutually trusted entity that pledges the integrity of a user's identity. Finally, the Session is a concept that represents a period of communication

TScIT 37, July 8, 2022, Enschede, The Netherlands

between a server and a client. These sessions can accept different kinds of parameters. One famous optimisation technique is called session resumption. It consists of not re-negotiating a session on every subsequent connection. The results when activating session resumption from this source [9] show that the amount of handshakes required is halved. To conclude, it has been established that standard TLS without optimisations is too computationally expensive and slow in terms of throughput for IoT devices without any optimisations. That is why it is necessary to investigate and provide a comprehensive study of the optimisation techniques within these protocols to make a step forward towards an encrypted IoT.
The rest of this paper is organized as follows: Section 2 states the problem statement and describes the concrete research questions. Section 3 provides an overview of related works that are used as reference to compare different TLS/SSL setups and also provide more in-depth knowledge about the mechanisms that will be analysed in this study. Section 5 is a performance analysis of TLS/SSL using the wolfssl library in the context of IoT devices. The section 6 Is a comprehensive security analysis that provides an analysis on setups to avoid and how to test for vulnerabilities in TLS implementations. In section 8 conclusions are drawn based on results in the performance and security analysis.

2 PROBLEM STATEMENT

This section aims to analyse the motivation and reasons driving the development of fast and efficient SSL/TLS implementations and define a specific problem to solve. It is clear from previous work mentioned in 3 that any secure and encrypted communication protocol is slower than its un-encrypted version. Specifically, it has been shown that TLS provides secure communication but is more energy-consuming and reduces greatly the latency of communications due to it's handshake requirement [5], making it unattractive to adopt especially in devices with hardware constraints such as limited memory and limited battery autonomy. Due to the nature of the encryption algorithms, some components cannot be optimised without significant security/speed trade-offs such as the minimal amount of handshakes or a too simple cipher suite [6]. However, these protocols can be optimised for constrained devices with different techniques such as picking an appropriate transportation protocol, varying the cipher suites and techniques such as session resumption as well as investigating an alternative underlying transport protocol for example User Datagram Protocol (UDP). Several attempts to investigate the following aspects(protocols, cipher suites, handshakes) in the context of TLS in an IoT environment. However the aforementioned investigations do not always take into account a combination of these techniques. And some have not been performed on IoT devices [10] namely this research that compares cipher suites but does not have figure for IoT devices but only for a virtual machine. This sparks the need for a comprehensive study that would analyse different TLS setups on a Raspberry Pi such that it is possible to run it efficiently over constrained devices altering specific steps within the protocol. The consensus is that TLS as is, is not a fitting solution for constrained devices. This requirement of efficiency on constrained devices requires investigating different TLS optimisation techniques to allow TLS to be widely used within IoT environments. This need for a comprehensive study is the driving factor introducing the following research question : **What influence do parameters such as Cipher suites, Handshake parameters such as PSK and Session resumption, and transport layer such as Transmission Control Protocol (TCP) and UDP have on the performances of an TLS/SSL connection and its security posture?**
his research question is split into 3 more concrete sub-questions that are answered in the context of this study

(1) *How do different configurations of TLS/SSL implementations on Raspberry Pi devices affect the security posture of IoT deployments, particularly in terms of resistance against common security threats such as man-in-the-middle attacks and protocol vulnerabilities?*
(2) *What is the impact of the transport layer of different TLS-based protocols, specifically TLS/SSL against DTLS, including session resumption and their respective decryption processes on the performance of Raspberry Pi devices in IoT environments, considering factors such as communication latency, throughput, under varying payload sizes ?*
(3) *How effective are optimization techniques such as elliptic curve cryptography (ECC) and choice of cipher suite in improving the performance of TLS/SSL implementations on Raspberry Pi for secure IoT communication?*

In the following section starts the performance analysis, analysing and describing setups aimed to speed up a TLS/SSL connection

3 RELATED WORK

In order to get resources on possible option this paper includes possible related works that have been used as guidance for designing the experiment and finding possible optimisations it is also to explain in more detail keywords used in previous section. The following research from [10] has proposed a benchmark of different TLS cipher suites. Namely ECDHE-ECDSA and RSA. In the cited paper the results indicate that on average ECDHE-ECDSA performs faster than RSA on Ubuntu systems. [8] also compares RSA and ECC. They conclude that ECC outperforms RSA regarding operational efficiency and security with lesser parameters, it is also concluded that ECC is more suitable for resource constrained devices. The choice of cipher suite is crucial in term of security aspects. In the following study they [11] observe a dataset of TLS connections and they see that only 50% of connections of the dataset implement TLS 1.3 and can be considered as secure as TLS 1.3 is now the standard and older ciphers are considered weak. It is generally agreed that UDP outperforms TCP in terms of throughput as mentionned in the results of [3]. This information could let one think that DTLS 1.3 should be faster than TLS 1.3. However concerning DTLS 1.3 and TLS 1.3 it is not so clear as they both implement a handshake in their protocol so even though the transmission of packets is faster it is not clear whether DTLS 1.3 performs better. There is little resources providing benchmarks of TLS and DTLS in the context of IoT devices.[12] compares the performance of the protocols TLS and DTLS from their version 1.2 to their respective version 1.3 and benchmarks results. The figures from this study show that for the same cipher suite about 80 byte are added [12] DTLS comparing the

TLS row with the same cipher suite. Another study [13] oriented for medical IoT devices reports that for mobile networks TLS increases the response time by 6.5% whereas DTLS increases the response time by 11%. Based on these research, it seems that in most cases DTLS adds more overhead than TLS. However there is not extensive material and no studies benchmark clearly DTLS against TLS for the same version in a general IoT context.Another possible optimisation is handshake resumption it can speed up the connection time for TLS/SSL by resuming a previous connection.This article [9] compares handshakes for TLS and session resumption and indicates that session resumption requires twice as little RTT than without it. [13] explains the handshake in more details. Resuming a session would allow a 1 RTT handshake. It is also possible to achieve a 0 RTT within TLS 1.3 in the case of the use of a PSK [14]. In order to speed up the connection process it is possible to accelerate the hashing step by choosing specific hashes or use PSK.The following article [15] highlights that comparing DHE-PSK-AES128-CBC-SHA256 with DHE-RSA-AES128-SHA256 results in an almost halved average connection time, in addition the following source [12] indicates that PSK outperforms normal TLS from a performance point of view and energy consumption. [16] Compares different pre-shared keys against public-key exchange mechanisms. They find that plain PSK performs better than any public key based mechanism, They also found that for DHE-PSK it performs better than RSA only when using small key sizes. Concerning TLS it is known to be vulnerable to so-called MITM attacks. One general guideline is to pass encrypted traffic as this already prevents the MITM from sniffing . It is said that strong client authentication in compliance with server invariance prevents TLS MITM attacks [17] in the following source. Considering the following previous research this study will carry on a comprehensive analysis and compare the results with previous literature.

4 PROPOSED SOLUTION

The analysis is carried on a server client architecture with both server and client running on the same device. This is also known as a loopback interface. The goal of doing it this way is to remove the network conditions from the equation and have a more accurate measure of connection time for TLS/SSL. In order to carry out this analysis the setup is to implement a TLS/SSL client/server pair with different setups. Specifically to verify TLS 1.3 with DTLS 1.3 alongside with TLS 1.3 with PSK and TLS 1.3 with Session resumption. The experiment will measure the average connection time over variable connection times that can be seen in Figure 4next, there is a test with the same setup but to test throughput. Throughput for the RX and TX buffers of the client and the serve will be logged for variable payload sizes that can be seen in Figure 3. The goal of this experiment is to answer RQ2. In addition different TLS 13 cipher suites are measured with different Elliptic curve groups by running the wolcrypt benchmark with software acceleration t. This will allow us to answer RQ3.

5 PERFORMANCE ANALYSIS

In the follwoing section the performance analysis will be carried the goal of this section is to answer the questions asked in research question 2 and 3 .concerning RQ 3 this section aims to answer whether the choice of elliptic curve and the of cipher suite can speed up the connection process by comparing them. Finally concerning RQ 2 the goal is to compare TLS/SSL setup and TLS 1.3 against DTLS 1.3

5.1 Hardware setup

In order to carry out the experiment the setup described in Figure 1 is used. In the following setup the raspberry pi is connected to the WiFi. For figure 2,3 and 4 both client and server were ran on the same raspberry pi on loopback interface. The client/server are the example the wolfssl library that can be modified accordingly with options to modify TLS/SSL setup. The setup of the pair is running the selected options TLS 1.3, Private Shared Keys and Session Resumption are using TCP/Internet Protocol (IP) transmission protocol.Network connections do not vary in this case since ran on same machine, this setup is called loopback traffic and can be monitored on Wireshark by selecting the loopback interface on the same machine.The connection time is bound to Processor speed and TLS Stack size. This allows to measure the connection times more accurately disregarding network conditions and evaluate solely the metrics influenced by the TLS/SSL implementation and optimisations. This setup also can be used to measure accurately the influence of hardware acceleration on arbitrary servers run on the machine.

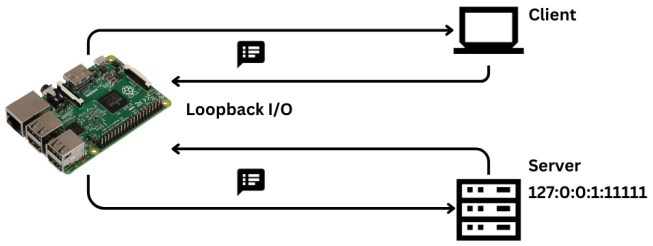

Fig. 1. Hardware setup for wolfssl client/server examples

5.2 Software setup

For the research and generating client/server pairs wolfssl-5.7.0 was used. Wolssl is a lightweight SSL/TLS library implemented in C with targeted to be ran on hardware devices IoT because of its size, speed and feature set. To setup wolfssl you must first install it then configure it with wanted parameters and then build it. More detailed steps on how to generate the data can be found in the github [18]. The library implements SSL protocols such as TLS 1.0 - 1.3 and supports a handpicked list of cipher suites compliant with the according TLS protocol. For the cryptographic tests wolfcrypt library was used which comes shipped with wolfssl. Wolfcrypt is a lightweight crypto library written in ANSI C and targeted for application in Embedded devices. This benchmark is measured in MB/s representing the throughput of the hash function for encryption for a block of 2^{20} bytes.

5.3 Wolcrypt and software acceleration

As discussed in earlier sections. The choice of cryptography matters significantly in the performances of TLS/SSL to uncover which setups might be promising it is possible to analyse the speed a cryptographic function executes in . The following figure demonstrates all supported wolfssl symmetric hash functions. One more detail is that those cryptographic fuctions can be accelerated by using mathematical libraries. In the following figure you can see different hash functions performed each with either **no optimisation, fastmath or fastHugeMath**.

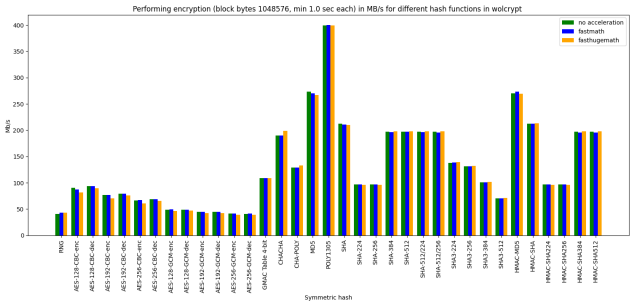

Fig. 2. TLS 13 symmetric ciphers benchmarks

From the result an observation can be made that the Symmetric hash function with the highest throughput out of the ones available by the wolfcrypt library is POLY1305, HMAC-MD5 and MD5. One more observation is that even though the math acceleration improves it does not seem to be a significant improvement. In addition in certain functions fastmath and fasthugemath are slower than the setup not including any software acceleration. This observation indicates that a math acceleration library does not mean it will necessarily be quicker on average for any symmetrical hash algorithm but it depends on which one thus when implementing it must be taken in consideration and a user must use the appropriate math library.

5.4 Comparison of TLS 1.3 cipher suites

As mentioned in Problem statement. The choice of Cipher Suite has a strong influence on the connection speed of TLS/SSL. To uncover which setups might be promising it is possible to analyse the speed a cryptography function executes the connection in. In 4 the average connection time over infinite connection attempts within 15 seconds is measured and Figure 3 the total amount of bytes transmitted over a TLS 1.3 connection with the following TLS 1.3 cipher suites list and with elliptic curves of different groups is graphed the metrics average connection time is chosen to be able to see the influence of TLS on the connection part itself and throughput is to see the effect of setup over the packet transmission.

`TLS13-AES128-GCM-SHA256,`
`TLS13-AES256-GCM-SHA384,`
`TLS13-CHACHA20-POLY1305-SHA256,`

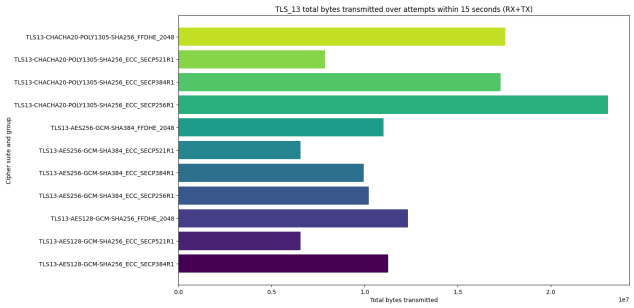

Fig. 3. TLS 13 cipher suites Amount of bytes transmitted over 15 s

The results from the experiment show that the most performant suite is the cipher suite TLS13-CHACHA20-POLY1305-SHA256 with an elliptic curve that belongs to the SECP256R1 group. In certain cases the cipher suites elliptic curve group in certain cipher suites affect the number of bytes transmitted, even when the same cipher suite for the rest of the operation is used . This shows that the choice of elliptic curve group is crucial for an implementation with superior performance. In TLS 1.3 a lot of previously available cipher suites have been pruned compared to TLS 1.2 in order to leave only the ones that implement the TLS 1.3 handshake and only ciphers. In addition the list of symmetric algorithms has been filtered and the remaining ones implement Authenticated Encryption Associated Data (AEAD) algorithms in wolfssl [19]. These measures guarantee protection against many potential attacks exploiting weaknesses in ciphers and non-AEAD ciphers. These results indicate that the choice of elliptic curve and elliptic curve group can greatly improve the connection time if chosen appropriately. These results are displayed in Figure 3

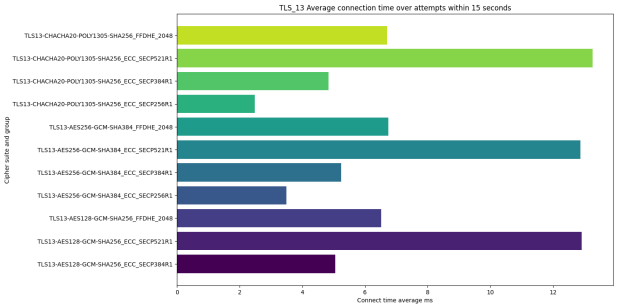

Fig. 4. TLS 13 cipher average connection time for connections over 15 s

The results for the average connection time follow the results from Figure 4 are similar to 3. They are correlated since connection time has a direct influence on throughput and amount of bytes transmitted in finite time.

5.5 Comparison of TLS/SSL optimisations

In the following sections the setups mentioned in Research question 2 and DTLS 1.3 against TLS 1.3 and state their results in comparison to each other. The objective is to find which setup is the fastest in terms of average connection time and throughput.

5.5.1 PSK. The results obtained show that PSK is slower than resuming a session but resumption does not happen for every connection. It is also significantly quicker than DTLS 1.3. This matches the findings from previous studies. It is advised in [20] to use ECDH or DH in combination with PSK in order to guarantee forward secrecy [21]. This can be seen in the graph of Figure 5

Concerning the results for throughput, For the RX buffer of server and client PSK performs worse on small values or as well as DTLS and performs as good as Session resumption for large value . The same observations can be made for the TX buffer in Figure 6.

This validates the results obtained in [12] as it also shows that from a performance point of view PSK has a significantly lower connection time than the other compared setups.

5.5.2 DTLS. The results seem to indicate that DTLS is slower for larger values and quicker for less than 100 subsequent connections. When running the benchmarks the DTLS server logs would stop for a few seconds while waiting for packets. One reason is that UDP being stateless, residual information from previous connections interferes with the subsequent ones thus slowing down a client that tries to connect thousands of times to the same server depending on packet timing. This indicated that the DTLS 1.3 implementation of wolfssl is slower than TLS for multiple subsequent connections . It is displayed in this from Figure 5.

Concerning the results for throughput, the RX buffer of server it performs on small values similarly to PSK and for large values similarly to TLS 1.3. For the Client RX buffer it performs better from 100, 1000, 2500 and worst 5000 and 10000 than other protocols. As for the TX buffer of the server and client it performs better than other stups except for small values in TX buffer of client. It seems that the throughput is not increased for the Receiving buffer but is improved for the transmitting buffer for DTLS compared to other setups. this can be verified in Figure 6 .

Comparing this to [3] it is clear that even though UDP outperforms TCP, DTLS 1.3 does not outperform TLS 1.3 in terms of throughput and is much slower for large amount of connections.

5.5.3 Session resumption. From the results the Session resumption is the method that give the quickest average connection time in Figure 5. However in practice the resumption is not happening at every connection, hence on the field this values will depend on the amount of allowed resumptions.

From 6 Session resumption has similar performances to TLS 13 and seems to be quicker for larger quantity of bytes transmitted for RX buffer for client and server and RX buffer of server

These results match the results found in [9] as it shows that Session resumption reduces the average amount of RTT by half. In our results the average connection time is more than halved. It is mentioned that in practice it should not be allowed to resume an infinite amount of connections. However it is clear from the results than resuming a session is more than twice as fast than establishing a new connection via TLS 1.3.

6 SECURITY ANALYSIS

Implementing TLS/SSL on IoT devices requires some optimisations described in previous sections. However certain combinations of

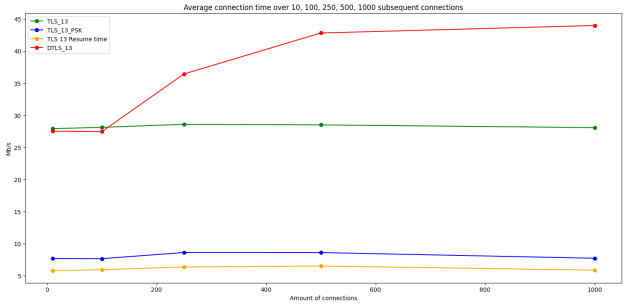

Fig. 5. Average connection time for 10, 100, 250, 500, 1000 subsequent connections

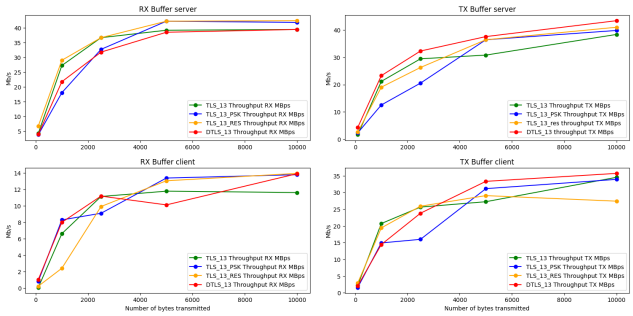

Fig. 6. Throughput of receiver and transmitter buffer of client and server

these optimisations can leave users of the servers vulnerable to certain types of attacks. This section aims to provide an answer to Research question 1 and provide an overview of vulnerable security postures and resistance against MITM attacks. When implementing SSL/TLS on IoT device possible vulnerabilities that could be introduced by the choice of the TLS setup must be taken into account. Factors that can influence the security posture are the choice of cipher suites, The protocol version and the chosen communication protocol. In the following sections go other possible attacks one can expose himself if choosing a vulnerable combination of ciphers.Tools are provided that can help a user implementing his own TLS/SSL server to verify whether his implementation is vulnerable to different kind of attacks by carefully avoiding the following pitfalls. This can be done with tools such as a TLS fuzzy test suite which tests for known TLS/SSL vulnerabilities. The following section covers those topics in more details.

6.1 Hardware setup

In order to perform MITM a raspberry pi on local network connected to WIFI was used. This setup is designed in such a way to have access to the local network and can act upon devices within it. The following section goes over the software components used in this analysis.

6.2 Software setup

In order to perform the analysis various tools for penetration testing have been looked into . Bettercap is a Command Line Interface tool that allows to perform various attacks such as Man In The Middle, TCP Proxies, Address Resolution Protocol (ARP) spoofing and also recognise devices on the network and sniff traffic. The ones that are interesting for the scope of this research are MITM attacks, TCP Proxy. To perform the attacks you need to install bettercap and follow this guide [22] for the TCP proxy you can follow the guide from bettercap docs [23] the code in order to launch the attack is the bettercap CLI with the script that is in this guide. This allows us to modify packets .

To test ssl implementations fuzzy tests are available open source. For instance the tls fuzzer implemented in python [24]. This library can be tested on openssl servers and any other kind of TLS/SSL connections. The following section will explain with more details the intricacies of these attacks in the next sections.

6.3 MITM with bettercap and TLS Proxy

MITM has been described as a common vector of attack used on TLS/SSL. This section gives concrete tools that can be used to perform a MITM attack on a TLS client/server pair. The initial and target architecture are described next in order to become man in the middle they give an indication on how an attacker would perform the attack. Usually a network setup for a local network looks like the following figure

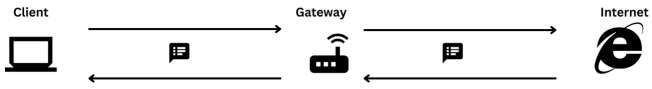

Fig. 7. Normal flow of client - router connection

How a man in the middle attack is performed with the bettercap module is by first sniffing target IP on the local network. Then choosing the target and the router the device pretends to be the router and sends the messages.It is now capable of sniffing traffic and if a request is made with http for instance the device will log it. This also allows for packet manipulation with more elaborated attacks however they need to be protocol specific and they are more elaborate and time consuming to perform. One interesting module is the TLS proxy which allows to perform the same operation but on an client and server and also allows a script that can be added and that can be modified and used to alter packets. It is possible to override packets. No concrete attacks with packet manipulation have been attempted due to lack of time but this could be for future works. The whole process is illustrated in Figure 8.

6.4 TLS fuzzer

To test TLS/SSL servers a method of testing called Fuzzy testing consists of testing multiple standard attacks or provide a script that can test for a potential attack.

Some well known attacks are TLS downgrade also known as SSL stripping which consists of negotiating a lower version of TLS than the advertised one by the server. It is often implemented with a

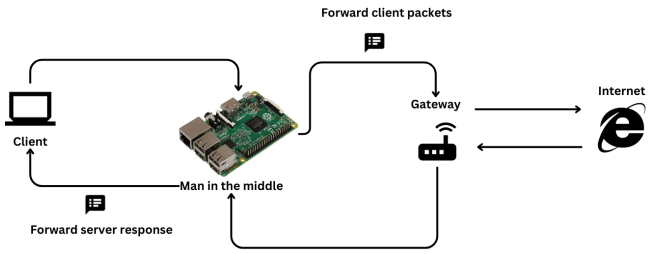

Fig. 8. MITM flow hardware setup

MITM setup. Since our sample servers were TLS 1.3 and DTLS 1.3 most of the common attacks on ciphers have failed on our servers. In some cases it would also fail due to difference in implementation of TLS/SSL libraries for wolfssl as it in . By lack of time it was not possible to get the TLS fuzzer libraries compatible with wolfssl to test everything It was possible to test those attacks on openssl servers like in the docs [24] of the repository.

7 DISCUSSIONS

Based on the results presented in the performance analysis specifically in section 5.5 it is clear how DTLS 1.3, TLS 1.3 and Setups with PSK and Session perform. This paper also has compared the available cipher suites and functions for TLS 1.3 in wolfcrypt and found ways to accelerate crypto functions with software. Previous research have mentionned that the drawbacks of TLS/SSL were great in terms of energy consumption and speed. From this study it is shown that it is possible to make an application-specific custom client server architecture that would meet the specific requirements of the application for speed by applying optimisations with the math library, the choice of cipher suite and the TLS/SSL stack but also by choosing a lightweight library such as wolfssl that is adapted for the application. This can improve the connection time problem for IoT devices using SSL/TLS that do not require autonomy. Since power consumption has not been measured it is not possible to make an analysis of the power consumption it is thus not possible to draw any conclusion on devices that require autonomy.DTLS 1.3 and TLS 1.3 have been compared, even though it seems that at the moment DTLS 1.3 performs worse on large value it has comparable results and could be a viable solution in the future. It needs to be investigated more as it is not a standard in the industry and is quite recent there is little literature about it so this could deserves attention in the future. The last section of this paper makes a security analysis and gives general guidelines on how potential attacks could be triggered, how to use fuzzy testing to test an SSL/TLS connection implementation and parameters. One direction for this is to extend the security analysis and focus on the DTLS 1.3 protocol as well as properly implementing attacks at the byte level that have not been done due to time constraint.

8 CONCLUSION

TLS/SSL is commonly used on the internet in HTTPS and TLS encrypts most of the internet. It requires optimisations as it has been shown to not be recommended with plain TLS 1.3 for IoT devices.

Optimisations are possible by using altering the transmission protocol, the cipher suite, and adding session resumption or PSK. In addition IoT devices are a potential target for attackers and requires a comprehensive analysis on potential vectors of attack. This paper demonstrates a comprehensive performance analysis and properties TLS/SSL in the context of IoT devices and answers the aforementioned research questions. More specifically it has been shown that the cipher suite choice when used in context of secure connections based on previous results has a direct effect on the speed of connections as can be seen in Figure 1 where cipher suites throughput is compared, it is also shown that it is possible to accelerate this process with software acceleration. The results in previous paper also show that ECC cipher suites perform better than RSA and are more adapted for constrained devices. In this study TLS 1.3 cipher suites are compared and the results in Figure 4 indicate that the elliptic curve group choice has a significant impact, in some cases for the same cipher suite the connection time is halved. Concerning handshake optimisations the results in 5 indicate that PSK and session resumption have a positive effect on reducing the average connection time . In the same results TLS and DTLS are compared . Previous paper [3] indicates that UDP outperforms TCP in terms of bandwidth which could indicate that DTLS 1.3 would be faster than TLS 1.3. In this study it is shown that it is not the case in 6 as the bandwidth is not superior and in some cases is performing worse than TLS 1.3 it is also shown in 5 that for large amount of subsequent connections DTLS is much slower than TLS. In addition to this this paper provides a comprehensive analysis and guidelines on how an attacker would approach attacking a connection via attack vectors and tools such as bettercap by performing using MITM. This paper provided a tool to test implementations of TLS/SSL in order to protect a connection from the most common vulnerabilities that are known.These results have for goal to give a global overview on how to implement a fast and secure SSL/TLS connection in the context of IoT devices and how to protect from common vulnerabilities. From the analysis of potential attacks in this section it is recommended to use the latest protocol version of TLS which is 1.3 as it prunes all weak cipher suites and the handshake is more robust as it addresses flaws in previous versions however it can still be vulnerable depending on the configurations that is why it is necessary to carry on fuzzy testing on the server in order to test the configuration for robustness.

REFERENCES

[1] "Number of connected IoT devices growing 16% to 16.7 billion globally." [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/

[2] "SSL and TLS: Theory and Practice, Third Edition - Rolf Oppliger - Google Books." [Online]. Available: https://books.google.nl/books?hl=en&lr=&id=TOnNEAAAQBAJ&oi=fnd&pg=PP1&dq=ssl+tls+history&ots=8asB5Rh50v&sig=lC02-Vd6GJpJjx918phj55FFofU&redir_esc=y#v=onepage&q=ssl%20tls%20history&f=false

[3] F. Taha Al-Dhief, N. Sabri, M. Albadr, F. Taha AL-Dhief, N. M. Abdul Latiff, N. Noordini Nik Abd Malik, M. Abbas Abbood Albader, M. Abed Mohammed, R. Noori AL-Haddad, Y. Dawood Salman, M. Khanapi Abd Ghani, and O. Ibrahim Obaid, "Performance Comparison between TCP and UDP Protocols in Different Simulation Scenarios," *Article in International Journal of Engineering & Technology*, vol. 7, pp. 172–176, 2018. [Online]. Available: https://www.researchgate.net/publication/329698255

[4] P. Cui, "Comparison of IoT Application Layer Protocols," 4 2017. [Online]. Available: https://etd.auburn.edu//handle/10415/5713

[5] I. L. B. M. Paris, M. H. Habaebi, and A. M. Zyoud, "Implementation of SSL/TLS Security with MQTT Protocol in IoT Environment," *Wireless Personal Communications*, vol. 132, no. 1, pp. 163–182, 9 2023. [Online]. Available: https://link.springer.com/article/10.1007/s11277-023-10605-y

[6] "Fortifying MQTT Communication Security With SSL/TLS | EMQ." [Online]. Available: https://www.emqx.com/en/blog/fortifying-mqtt-communication-security-with-ssl-tls

[7] "An Introduction To Cipher Suites | JSCAPE." [Online]. Available: https://www.jscape.com/blog/cipher-suites

[8] "(1) (PDF) RSA and ECC: A comparative analysis." [Online]. Available: https://www.researchgate.net/publication/322558426_RSA_and_ECC_A_comparative_analysis

[9] "TLS Session Resumption: Full-speed and Secure." [Online]. Available: https://blog.cloudflare.com/tls-session-resumption-full-speed-and-secure/

[10] "(1) (PDF) Performance Analysis of SSL/TLS Crypto Libraries: Based on Operating Platform." [Online]. Available: https://www.researchgate.net/publication/359906722_Performance_Analysis_of_SSLTLS_Crypto_Libraries_Based_on_Operating_Platform

[11] L. Universitet, S.-L. +++, S. Frisenfelt, and E. Kjell, "Characterization of cipher suite selection, downgrading, and other weaknesses observed in the wild Karaktärisering av cipher suite val, nedgradering och andra svagheter som observerats i det vilda." [Online]. Available: www.liu.se

[12] G. Restuccia, H. Tschofenig, and E. Baccelli, "Low-Power IoT Communication Security: On the Performance of DTLS and TLS 1.3," *2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks, PEMWN 2020*, 11 2020. [Online]. Available: https://arxiv.org/abs/2011.12035v2

[13] "Benchmarking SSL Performance." [Online]. Available: https://www.haproxy.com/blog/benchmarking-ssl-performance

[14] "Whats new with TLS 1.3. Recently TLS 1.2 got updated to TLS… | by Robert van Rijn | Medium." [Online]. Available: https://medium.com/@vanrijn/what-is-new-with-tls-1-3-e991df2caaac

[15] "When to use Pre Shared Key (PSK) Cipher Suites – wolfSSL." [Online]. Available: https://www.wolfssl.com/when-to-use-pre-shared-key-psk-cipher-suites-2/

[16] F. C. Kuo, H. Tschofenig, F. Meyer, and X. Fu, "Comparison studies between pre-shared and public key exchange mechanisms for transport layer security," *Proceedings - IEEE INFOCOM*, 2006.

[17] N. Karapanos, S. Capkun, and E. Zürich, "On the Effective Prevention of TLS Man-in-the-Middle Attacks in Web Applications." [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/karapanos

[18] "SSL-TLS/research-main/src at main · pandasansgains/SSL-TLS." [Online]. Available: https://github.com/pandasansgains/SSL-TLS/tree/main/research-main/src

[19] "TLS 1.3 Protocol Support | Documentation – wolfSSL." [Online]. Available: https://www.wolfssl.com/docs/tls13/

[20] "TLS 1.3 Performance Part 3 – Pre-Shared Key (PSK) – wolfSSL." [Online]. Available: https://www.wolfssl.com/tls-1-3-performance-part-3-pre-shared-key-psk/

[21] "Tls 1.3 PSK with ECDH (Page 1) — wolfSSL — wolfSSL - Embedded SSL Library." [Online]. Available: https://www.wolfssl.com/forums/topic2051-tls-13-psk-with-ecdh.html

[22] "Man In The Middle Attack Using Bettercap Framework | HackerNoon." [Online]. Available: https://hackernoon.com/man-in-the-middle-attack-using-bettercap-framework-hd783wzy

[23] "tcp.proxy :: bettercap." [Online]. Available: https://www.bettercap.org/modules/ethernet/proxies/tcp.proxy/

[24] "tlsfuzzer/tlsfuzzer: SSL and TLS protocol test suite and fuzzer." [Online]. Available: https://github.com/tlsfuzzer/tlsfuzzer