POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Empirical Use of Network Time Protocol in Internet of Things Devices: Vulnerabilities and Security Measures

BASMA CHANDID
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Empirical Use of Network Time Protocol in Internet of Things Devices: Vulnerabilities and Security Measures

présenté par **Basma CHANDID**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Samuel PIERRE, président
Alejandro QUINTERO, membre et directeur de recherche
David BARRERA, membre et codirecteur de recherche
Martine BELLAÏCHE, membre

DEDICATION

Life is an unexpected journey
outstanding is your guidance in it
vital is your role in my existence
everything I am is shaped by you
To my family...

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my research supervisor, Professor Alejandro Quintero, and my co-supervisor, Professor David Barrera, for their support throughout this project, and the opportunity to work on my thesis under their guidance. I also thank Professor Samuel Pierre and Professor Martine Bellaïche for being part of my thesis committee and taking time to assess my work. Finally, I would like to acknowledge the work done by Alarwi and al., that was made available to the public, on which we based parts of our analysis.

RÉSUMÉ

Network Time Protocol (NTP) est un protocole responsable de la synchronisation du temps dans les environnements de réseautique. Il est utilisé depuis plus de trente ans afin d'aider les appareils connectés à acquérir l'heure correcte depuis le réseau. Ce protocole sert entre autre à garantir les informations d'horodatage et la cryptographie des fichiers journaux. Il a également fait l'objet de nombreuses études visant à rendre ce pilier des réseaux et télécommunications plus sûr et plus robuste.

Ceci dit, la synchronisation du temps est l'un des composants essentiels au bon fonctionnement des réseaux d'objets connectés (IoT) aussi. Il est primordial pour la planification des tâches, la tenue des journaux, la surveillance des différents comportements et la mise à jour du temps lors de l'utilisation de protocoles de chiffrement ou de mise en file d'attente dans ces réseaux. L'importance de ces facteurs clés peut être ressentie lorsqu'ils sont rendu non valides à cause de la composante temps, par exemple si l'heure est décalée ou est incorrecte lors de l'utilisation de dispositifs précis dépendant du temps. Cela entraîne un retard ou une négligence totale de la tâche, ce qui peut être dangereux si ces appareils sont utilisés dans une chaîne d'action au service de la vie humaine (soins de santé ou circulation de véhicules).

Bien que la synchronisation du temps au moyen du protocole NTP et sa sécurité soient importantes dans les réseaux IoT, à notre connaissance, les recherches existantes en synchronisation du temps en IoT ne concernent que la création de nouvelles méthodes et protocoles sans l'analyse des défauts existant et de possibles mesures de sécurité orientées vers NTP. Cela amplifie l'hétérogénéité déjà présente au niveau de la panoplie des protocoles IoT et ne résout pas les problèmes de sécurité NTP des appareils intelligents existants.

Dans ce but, nous nous concentrons dans cette thèse sur l'étude des différentes façons dont NTP est utilisé dans les réseaux IoT. Nous effectuons également une analyse empirique de la méthode utilisée par les objets connectés afin de choisir leur port réseau NTP et les serveurs NTP publiques auxquels ils synchronisent leur horloge. Cette analyse nous donne une idée de la façon dont la synchronisation NTP peut être utilisée pour attaquer et tirer profit des appareils intelligents connectés. C'est également un moyen de recueillir des informations sur la variété des implémentations de ce protocole utilisées par différents fournisseurs.

Afin de compléter l'étude empirique de la synchronisation temporelle correcte à l'aide de NTP dans les réseaux d'objets connectés, nous avons mis en place différents environnements de test basés sur différents types de dispositifs IoT. Nous étudions leur utilisation normale de NTP à des fins de comparaison. Nous appliquons ensuite des attaques sur ces architectures

afin de prouver leur insécurité face aux décalages dans le temps. Cela présente également les possibilités d'utiliser ces attaques pour affecter les tâches humaines automatisées qui dépendent de la précision du temps, mettant ainsi en danger certains aspects de notre vie qui dépendent de ces appareils.

En se basant sur les vulnérabilités utilisées dans ces attaques, nous suggérons deux différentes architectures sécurisées pour répondre à ce problème. Elles dépendent de nouvelles implémentations de versions sécurisées de NTP étudiées dans les réseaux traditionnels. Ces architectures utilisent Network Time Security (NTS), une version authentifiée de NTP qui utilise TLS pour le tunneling. Dans notre première architecture, nous l'appliquons séparément de l'appareil IoT et dans la deuxième architecture, nous l'incluons directement dans l'appareil. Nous étudions la faisabilité des deux solutions et ce qui est nécessaire pour que l'objet connecté puisse établir une communication NTP sûre et sécurisée avec le serveur NTP.

En proposant ces solutions, nous atteignons nos trois objectifs de départ. Nous aurons ainsi étudié l'importance du temps dans les services fournis par les appareils IoT. Nous aurons, à travers des tests et une évaluation des ensembles de données, envisagé et utilisé les vulnérabilités des communications de synchronisation du temps en IoT. Enfin, nous aurons élaboré une proposition pour deux architectures sécurisées possibles qui traitent ce problème de deux manières différentes.

ABSTRACT

Network Time Protocol (NTP) is a protocol responsible for time synchronization in networking environments. It has been used for over three decades in order to help connected devices acquire the correct time over the network.It is implemented to ensure log file time stamp information and cryptography. It has also been the topic of many studies aiming at making this pillar of networking and telecommunications more secure and robust.

Time synchronization is one of the essential ingredients needed for Internet of Things (IoT) networks to function correctly. It is important for scheduling tasks, keeping logs, monitoring different behaviors and for keeping the time concept up to date when using encryption or queuing protocols in these networks. These key factors can be noticed when they are tampered with such as if the time is incorrectly shifted during the use of accurate time dependent devices. This leads to the task being delayed or overlooked completely which can be dangerous if these devices are used in life-dependent chains of action (healthcare or traffic for example).

Although NTP, time synchronization and their security are important in IoT networks, to our knowledge, the existing research in IoT time synchronization only concerns creating new methods and protocols. This adds to the heterogeneity of the field's components and does not solve the NTP security issues in existing smart devices.

For these reasons, we focus in this thesis on studying the different ways NTP is used in IoT networks. We also run an empirical analysis of how IoT devices pick their NTP network source port and the servers they choose to synchronize to. Doing this analysis gives us an idea about how NTP synchronization can be used to tamper with and take advantage of connected smart devices. It is also a way of gathering information about the variety of protocol implementations and policies used by different vendors.

In order to complete the empirical study of correct time synchronization using NTP in these networks, we put together different test environments of IoT devices. We study their normal use of NTP for comparison purposes. We then apply attacks on these devices in order to demonstrate their insecurity towards time shifting attacks. This also showcases the possibilities of using these attacks to affect human automated tasks that depend on time accuracy, thus endangering some life aspects for which these devices are needed.

With the same process of using existing regular networks' NTP attacks in IoT networks, we suggest two different secure architectures for this issue that depend on studied implemen-

tations of secure NTP. These architectures use Network Time Security (NTS) which is an authenticated version of NTP that uses Transport Layer Security (TLS) for tunneling. In our first architecture, we apply it separately than the IoT device and in the second architecture we include it directly in the device. We study the feasibility of both solutions and what is required in order for the device to complete a safe and secure NTP communication with the NTP server.

By suggesting these solutions we achieve our three research objectives. We have studied the importance of time in services provided by IoT devices. We have, through tests and data set evaluation, contemplated and used IoT time synchronization communications' vulnerabilities. And finally, we elaborated a proposition for two possible secure architectures that deal with this issue in two different ways.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

ANTP	Authenticated Network Time Protocol
API	Application Programming Interfaces
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CPU	Central Processing Unit
DNS	Domain Name System
DDoS	Distributed Denial of Service
DoS	Denial of Service
GPS	Global Position System
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IoT	Internet of things
KoD	Kiss of Death
MQTT	Message Queuing telemetry transport protocol
MTS	Maximum consensus-based Time Synchronization
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
NTS	Network Time Security
OS	Operating System
OSI	Open Systems Interconnection
PiTM	Person in The Middle
PTP	Precise Time Protocol
RTC	Real Time Clock
SCADA	Supervisory Control And Data Acquisition Systems
SNTP	Simple Network Time Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 General concepts

Today, Internet of Things (IoT) devices are considered a bigger part of our daily lives than they have ever been. They are expected to be even more present in the future. For a world population of a little over 8 billion people [1], it is estimated that we will have over 75 billion connected devices by the year 2025 [2]. Which averages more than 9 network-connected devices per person.

What is surprising about those numbers, other than how big they are, is how practicable they seem when we consider how we live our lives in the present time. From a smartphone to a connected watch, a smart lock, a door security camera, a smart vehicle, a voice controller at home and a connected baby monitor or a simple connected plug, we see an interesting diversity of daily used objects inside the same households. Many people have already reached 5 to 8 IoT devices without including the devices we share with people outside the comfort of our homes and vehicles.

These devices are now almost essential to our day to day interactions from playing music at home to paying electricity bills at the store using banking credentials that are saved on a smartwatch. It is no longer about the amount of data that we exchange over these devices, it is now about the severity and security of such data. An estimated 90 Zettabytes of data, billions of Terabytes, will be processed by IoT devices by 2025 [3]. Those devices will be exchanging information as important as identification, social security numbers, license numbers and banking information of not only individuals but also firms and companies. We can only wonder how safe it is for our information to be out there and if there are any measures implemented to keep it secure from current and future cyber threats.

The features which generally attract users to IoT devices are mobility and efficiency. Connected devices are mostly small if not integrated into bigger devices such as ovens, fridges or even cars. Their small size, a big selling-point, makes it so that the processing power and CPUs used to make them are a lot smaller since they can not fit. The downgrade of the CPU leads to the use of weaker protocols and security measures. In certain cases, the security aspect is only achieved by the first authentication and no other process.

This lack of good cyber protection in connected devices added to their network capability turns them into easy prey for attackers. These attackers do not only have access to IoTs but to the network they are connected to and all the critical data it contains. In which case these

attacks can be quite dangerous for the victim.

Another feature of IoT devices is how they highly depend on time precision and correctness for calibration needs or scheduling tasks. For example, a surveillance camera that needs the correct time to keep coherent logs of events. For scheduling task that depends on time correctness, we have, among other devices, a network-connected infusion pump that releases a certain drug into a patient's system at specific times during the day. If such devices were to have issues related to tampering with its time precision, the patient's life could be endangered due to either a big dose of medicine or not enough.

This feature is shared with "traditional/regular networking systems" such as laptops and routers. Their high reliance on the correct time for correct reliable network communication has been the subject of many studies. A number of which centered around the security of the time component and how it can become the weak link used by attackers to have access to the network if tampered with. The industry of these "traditional system", being much older, is well aware of some drawbacks in time-dependent services and has been producing ways of securing them such as encryption or more complex algorithms.

Another point of connection between IoT networks and traditional networking systems is the protocols used for their different services. Although IoT uses a larger spectrum of protocols, the network-oriented protocols are mostly dependent on the same ones used for traditional networks. They can be the exact same ones if they are of lower complexity or privately developed lighter versions of them.

Seeing how these two types of networks, traditional and IoT, connect on two features: protocols and time correctness importance, the time component for IoT devices can be used as a way to hack these networks especially if the protocol used is paired with less security. The cyberattacks used to tamper with regular time-dependent protocols in traditional networks could be used to tamper with IoT networks' critical services in such cases.

So far there is no collection of time-based attacks such as the ones studied for traditional networks. Their impact, just like the IoT field itself, is a grey area that is open for research. It is essential to venture into these attacks and how they can affect the daily services we use IoT devices for. Thus, leading to an open door for measures to counter these vulnerabilities and for the deployment of standardization of IoT security.

1.2 Elements of the problematic

From the foregoing, it seems that the new embracing of IoT networks and the ease of their use have been made to the detriment of the security component. As a result, it is now necessary to guarantee the users the integrity and confidentiality of their data. Companies in the industry that do not take protective measures can run considerable risks to their products and clients.

Individuals are also prone to being victims of the misuse of connected or smart objects in our society. They are unaware of the level of security that is needed and trust the vendors to take into consideration their needs on all aspects. These same users use those devices in fields that are closely related to their well-being and daily tasks. Thus, they intensify the risk they run.

One of the many points of view interesting in the security assessment of IoT devices has to do with the concept of time and how it is implemented. This mainly concerns the devices that depend on correct and accurate time synchronization in order to accomplish the tasks they were made for. These tasks can concern either scheduling features offered by the device or even monitoring.

A few examples of situations where correct time synchronization is considered necessary in IoT networks can be discussed. The first one being in the health care field where we have smart infusion pumps connected. These devices are now included in the process of treating patients as they are programmed with schedule to supply the patients with correct medicine dozes on correct times. The pumps are therefore managed by doctors for those elements. If these devices were to be attacked through tampering with their time and changing it to a less favorable time for the patient to either give them more medicine or less medicine, it can lead to dangerous situations.

Another threatening situation where the use of correct time dependent connected devices can be seen as life threatening is in smart cities. If we set the stage as two consecutive connected traffic light where the first one needs to be red before the second one is green or at least at the exact same time. If the first light is set through an attack on the time protocol to be just a few seconds delayed compared to the second and it goes off late due to that delay, there could be a serious accident as consequence of that.

There are many more situations where today's connected devices that depend on time can be used to impact the real world negatively. This fact leads us to think of how the concept of time is taken for granted in the process of securing smart IoT devices. It is also a situation that motivates us to look into helping vendors realize the severity of ignoring time correctness

in the making of their devices.

That is why it is vital for the IoT fields to take into consideration certain steps in response to one of the main problems concerning connected devices security: Time protocols must be looked into and studied in IoT in as much detail as they have been in traditional networks due to their growing importance.

On another side of the issue of time correctness, we know how diverse Iot devices and networks are especially when it comes to their software and protocol. The time synchronization protocols developed for them are no different. There exists a big variety of protocols that are described as time accurate and smart devices oriented.

The downside to this diversity in time protocols is that it is unclear which ones are the most used by vendors and which ones are secure or how they are implemented. It is also clear that one of the main time synchronization protocols used in the field is NTP.

NTP is a protocol that has been around for a long time but weaknesses are finally being shown through certain security assessment studies. These studies only show the implementations in regular networks. The latest of which gave a more secure authenticated version of this protocol in the year 2019 [4].

The implementation of NTP in these types of networks has yet to be studied. This protocol has shown to be vulnerable to a number of attacks throughout the 35 years of its use in regular networks. How different it is and how its RFC tests are done in IoT networks is not certain.

For this research work, we are focusing on the importance of studying and evaluating the security of time use in connected devices. We are also studying the most popular time protocol in regular networks, in IoT networks. To do so we ask the following questions:

- How do IoT devices rely on time in their use and which services can this use affect?
- To which degree can time be the weak link in the IoT network connected architecture?
- How can NTP be used to tamper with/abuse the services that are made available by these devices?

We intend for this work to attract the attention to the security issue that can be time in internet of things networks. We also work on opening the door for more future research in this area by suggesting possible security measures to be taken in this specific area.

1.3 Research objectives

In response to the questions asked, the main objective of this thesis is to conduct a study of the empirical use of the network time protocol in internet of things networks. This will help show the role time plays in the IoT field and to which degree it can be trusted and secured.

Moreover, keeping in mind how immense this field has become in the last two decades, we intend on proceeding with a risk analysis from a security point of view of the main tasks time is used for in these devices. This analysis will be achieved by accomplishing the following objectives:

1. Defining the range of importance of time in services provided by IoT devices and their importance in human daily use;
2. Drilling down through IoT time synchronization communications and their vulnerabilities;
3. Elaborating a possible protection / security architecture that takes into consideration the time component as well as the mobility service of IoT devices.

1.4 Thesis outline

This thesis contains 5 chapters, the first being the introduction. The second chapter is a literature review discussing IoT time synchronization and NTP security as well as the advances made in linking these two fields. Then, the third chapter is an analysis of a data set of 50 diverse devices, mainly IoT smart devices, and their use on NTP implementation and security components.

Chapter 4 capitalizes on the conclusions from chapter 3 to demonstrate NTP vulnerability in IoT networks through different attacks. We analyze their results and possible repercussions on the use of connected devices in both IoT networks and the physical world through human lives. In this chapter we also , based on the precious vulnerabilities discussed, push forward 2 security architectures to help have a better use of NTP in this field.The final chapter provides an overview of our results and opens our work to limitations and challenges that can be further studied in future works.

CHAPTER 2 LITERATURE REVIEW

This chapter reviews relevant works in the NTP and NTP security literature in IoT networks. We include papers that discuss time synchronization measures and security in connected devices. We note that while the field of network-based time synchronization dates back several decades, we concentrate on recent works as they address the challenges in emerging IoT devices and networks as well as the use of advanced security methods in the NTP protocol.

2.1 General concepts in IoT security

This section provides a context for this research work as well as an overview of the state of the art of IoT security with a focus on network-based time synchronization.

To start, for a clearer and more focused analysis, a spectrum of the type of devices the research includes is defined. We then discuss some of the reasons stopping IoT companies from including complex security measures in their connected devices. And finally, we provide an introduction to how accurate time dependency plays a major role in connected devices functioning.

2.1.1 Spectrum of studied IoT devices

A single, unified definition of IoT devices has been the subject of deliberation in the research field ever since they have become such an imminent part of our daily lives. That is mainly due to the variety of forms and services offered by those devices as well as their features and capabilities. They go from simple sensors with just enough processing power to send a specific number to complex systems that run lightweight software to process intelligent data through complex protocols while still being lightweight.

According to Patel and al. [5], one general definition of an IoT network is a "type of network capable of connecting anything with the Internet based on stipulated protocols through information sensing equipment to conduct information exchange and communications in order to achieve smart recognition, positioning, tracking, monitoring, and administration.".

Although such definition does not specify an exact science by which we can choose the specific "smart objects" to call IoTs, Barrera and al. [6] have stated during their study that there are different scopes for different definitions of IoT devices that can go beyond consumer IoT – devices that an end-user could purchase and connect to their home network to include industrial control systems, supervisory control and data acquisition systems (SCADA), smart

grid devices like smart meters, and vehicular systems.

Many research studies have tried to achieve a detailed categorization of IoT devices. The latest of which introduces machine learning by Desai and al. [7] to develop a framework of fundamental statistical tests on various IoT devices features to rank these features and coming up with an algorithm for device classification based on network traffic data collected from a real testbed.

A second method by Gigli [8] depends on the types of services provided by IoT devices to differentiate between 4 types of IoT devices:

- Identity-Related Services
- Information Aggregation Services
- Collaborative-Aware Services
- Ubiquitous service

These types of services, although Data and flow-oriented, do not include the complexity of the devices and its processing power, which are in most cases the decisive factor when it comes to the use of certain security methods.

A third method suggested by Patel and al. [5] looks at the IoT as a network and applies a layer model to it with 4 layers (Application, Service support, Network layer, and Smart object/sensor). The devices are categorized as three different types of "Technologies" going from the least powerful and smallest devices to devices that support network sharing and take into consideration capacity and latency, to finally a group of devices that is management-capable and supports IoT applications. Although such categorization is the closest to the level of detailed categorization needed for the field it is still lacking the thresholds from which we can make the decision as well as it is very closely related to the models used for traditional networks (networks without smart embedded devices).

A fourth way of Categorizing IoT devices is introduced by Zuener and al. [9] and is a model prototype "aimed at maintaining commercial, operational, but also social, legal and regulatory transparency". It consists of a method of labeling devices depending on a description of data flow direction, storage options, economic and technological life cycle as well as social acceptance issues. It is a fine-grained method using certain subjective criteria that do not necessarily depend on machine measurements.

The final method on this list is the one described on the IETF RFC7228 by Keranen and al. [10]. It contains a detailed classification of IoT nodes according to their power usage

strategies (normally off, low power and always on) as well as their energy limitations (Event energy-limited, Period energy-limited, Lifetime energy-limited, and No direct quantitative limitations to available energy). This standard uses the nodes classes described as follows :

- **Class 0 devices**: very constrained sensor-like nodes. They are limited in memory and processing capabilities. With not enough resources for secure internet-capable communications. They depend on larger devices for monitoring and "management".
- **Class 1 devices**: quite constrained in code space and processing capabilities, using lightweight internet protocols. They have limited existing security capabilities for use in larger networks.
- **Class 2 devices**: less constrained and able to supporting most protocol stacks used on notebooks or servers. They might have lightweight protocols that are energy-efficient for the purpose of consuming less bandwidth. Furthermore, using fewer resources for networking leaves more resources available to applications. Thus, using the protocol stacks defined for more constrained devices on Class 2 devices might reduce development costs and increase interoperability.

Our research will be focusing more on classes 1 and 2 since they are more likely to be using time protocols that deal with synchronization and latency issues. These classes include devices such as connected light bulbs, connected refrigerators, smart traffic lights, or even connected medical infusion pumps. They cover a wide range of life aspects and are close to human daily use. These devices also depend on time for accuracy and completion of their services. This can be shown in the smart traffic lights example which need to all be synchronized and have the same time to avoid accidents or driver confusion. This shows the possible vital importance of time accuracy in such IoT devices.

2.1.2 IoT security vulnerabilities

These connected devices defined above, according to Alarwi and al. [11], are based on traditional mobile network capable computing systems which themselves have proved to be vulnerable to certain attack vectors in addition to the ones that are induced by the IoT devices architecture. These vulnerabilities are mostly due to limitations that can be described in three categories defined by Hossain and al. [12]:

- **Limitations based on hardware**: most connected devices have computational, energy and memory constrain because they function using batteries, low power CPUs

and limited RAM and Flash memory. Therefore, it is rare to use protocols based on algorithms depending on high computational power. This leads to complicated security methods such as those based on cryptography;

- **Limitations based on software**: security modules designed for IoT devices' protocol stack are thin and fault tolerant to allow tasks to be done efficiently;
- **Limitations based on network**: high mobility features on IoT devices come with security threats as the device favors connection over protection when looking for a new network. In addition, the heterogeneity of the networking protocols connected devices use to either communicate with IP or non-IP protocols-based networks makes the use of traditional security policies and rules difficult.

It is now clear that the biggest drawback in the studied IoT devices and the origin of most of the security threats is, as described in what Geneiatakis and al. [13] concluded, their low processing power especially when their attack surface area varies according to several vectors described in Gai and al.'s work [14] which include the ecosystem access control, device memory, web interface, firmware, third party Application Programming Interfaces (API) and cloud interfaces and as many security analyses the more vectors you include in the functioning of a system the greater the attack surface area is.

According to Baker and al. [15] an example of how the low power constrain plays against the efficiency of these devices is apparent in the operability of photoplethysmographic (PPG) wrist-wearable sensors, which are wearable smart devices with sensors capable of measuring arterial pulse in the wrist but "As motion affects the accuracy of pulse readings from PPG sensors, an accelerometer is used to check for movement. When motion is high, the device goes into a low power state and does not record pulse." The PPG wristband stops sending information that could include the person seizing or having cardiac issues during a high motion state such as running. This device that was supposed to accurately monitor a patient's conditioning could be sending false data that could put the patient's life in danger.

This specific CPU issue has been described as well as part of the Challenges for a Secure IoT in the RFC8576 [16] along with fragmentation [17] and bandwidth constraints. The first being the under-usage of the resources on the gateway since the offloading levels on the IoT edge nodes are discrete and coarse-grained. That is due to the inability of IoT devices to change their offloading level with not enough power, which would result in unused resources on the gateway. The reverse end of this issue is the bandwidth constraint leading to a need for smaller packet-size limits which give leeway for more attack vectors.

2.1.3 IoT dependence on accurate time synchronization

All connected devices, whether on regular traditional networks or IoT networks, depend on accurate time precision to complete correct tasks when communicating with each other and with cloud services. That can be explained by the fact that each device on either end is set to wait for a request from the other that usually has a timestamp. There are instances where this can be seen such as timeouts, scheduling services, keeping logs and queuing mechanisms in IoT devices that depend on time to keep working.

The dependence of sensor networks on correct time synchronization has been presented in [18] [19] [20] in their efforts to conduct an in-depth security analysis of sender-receiver synchronization protocols. It is considered a major middle-ware service in these networks especially for cases such timekeeping of logs of events conducted by sensors or the measurements of events that depend on the time referential such as time-of-flight of sound (time taken by sound to travel a specific distance) or the coordination of future events. In case an attack was to tamper with these time values, nodes will not be capable of estimating the correct time for their services which could lead to loss of important packets or a denial of service.

Such examples show the importance of time synchronization correctness in the security of the devices, the service as well as the information communicated.

On the other hand, when it comes to the IoT connected devices that are studied for this thesis (classes 1 and 2 IoTs), their always-on, always-connected nature according to OConnor and al. [21] adds more complications to their security and privacy from a time synchronization point of view. These devices, usually directly connected to sensors on one end and edge devices on the other, are responsible for communicating sensor, state and heartbeat data (low bandwidth message exchange to monitor connectivity health of a device). Data that depends on time synchronization for logging and forensic evidence as well as for periodically keeping tabs on the connectivity health.

2.2 NTP security in traditional networks

This section discusses time synchronization protocols in traditional networks by stating what they are, how they work, the reason behind their use as well as giving a brief security evaluation of these protocols from their vulnerabilities to their security measures. This section also emphasizes the characteristics of NTP and its security in traditional networks.

2.2.1 Time synchronization protocols over the internet

In traditional networks, the notion of time synchronization has been introduced with the start of communication possibilities over the networks between computers and network devices. One of the first means to standardize the measurement of time during these communications was according to RFC868 [22] describing the Time Protocol back in 1983 built on both TCP and UDP. It is a simple poll of several independent sites on the network to determine the system's idea of the correct time.

Aside from the Global Positioning System (GPS), the main protocols used for time synchronization over the internet are NTP and precise time protocol (PTP). They are also the basis for the development of many other lighter or more complex versions of them.

Precise Time Protocol

Precise Time Protocol (PTP) [23] [24] was developed for networks needing highly precise time synchronization without having access to satellite navigation signals such as mobile phone towers. PTP (see figure 2.1) is based on BEST Primary Clock Algorithm [25].It is an algorithm to help choose which clock to use as the source of timing on your network in case we have more than one in the same network. This is helpful as it pushes to avoid the cases of no clock redundancy where loss of the GPS means no clock synchronization. A redundant primary clock takes over in this case. The choice of which clock is primary is defined through the process of both clocks sending a message to the network to detect other clocks. A data set comparison is then performed based on information such as time source accuracy, offset and variance.

PTP is considered a relatively new protocol and is more used for SCADA systems rather than the IoT devices we are studying.

A security analysis conducted on this protocol by Tsang and al. [26] shows that the non application of cryptographic integrity protection on all PTP messages leads to being vulnerable to modification attacks allowing Denial of Service (DoS), incorrect re-synchronization and altering hierarchy of the primary and secondary clocks.

Another vulnerability is present in the lack of a centralized authentication process which makes it easy for an attacker to take on the role of the primary clock and launch an attack on all the network.

The absence of a backup plan in case the time messages are not received allows for attackers to use delay threats as well as replay attacks because of the non-encryption of the network path (spoofing and packet injection).

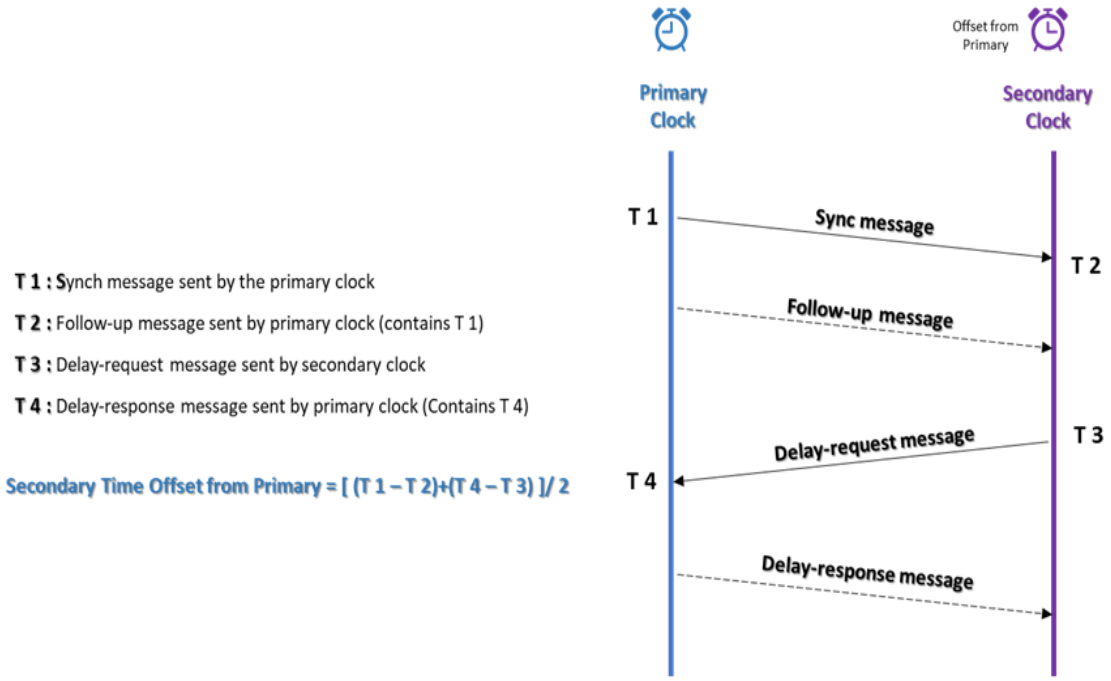


Figure 2.1 PTP Primary clock-Secondary clock communication

Network Time Protocol

The Network Time Protocol (NTP) (see figure 2.2) introduced in 1985 [27] [28] [29] is a UDP only protocol that introduced the notion of *reference clocks* that could be connected to "gateways", connected themselves to each other and to other nodes on the networks. Depending on certain values measured, these gateways verify the time and communicate it to the rest of the network.

NTP introduced the notion of a stratum (see figure 2.3) which is a number indicating how far the gateway is from the reference clock. Reference clocks are considered stratum 0 and the number keeps increasing with every previous gateway the current device is getting its time from. Between 1 and 15, the device can still be considered and NTP server for the next device.

An NTP packet as shown in figure 2.4 also consists of a number of important fields that come into play in the application of its algorithm:

- **Version**: NTP versions go from 1 to 4 but the first two are known to be obsolete and only versions 3 and 4 are currently used with 4 being predominant.
- **Mode**: an NTP device can be in 3 different states (symmetric, client/server or broadcast). These states then vary into the 6 different modes of NTP (symmetric passive,

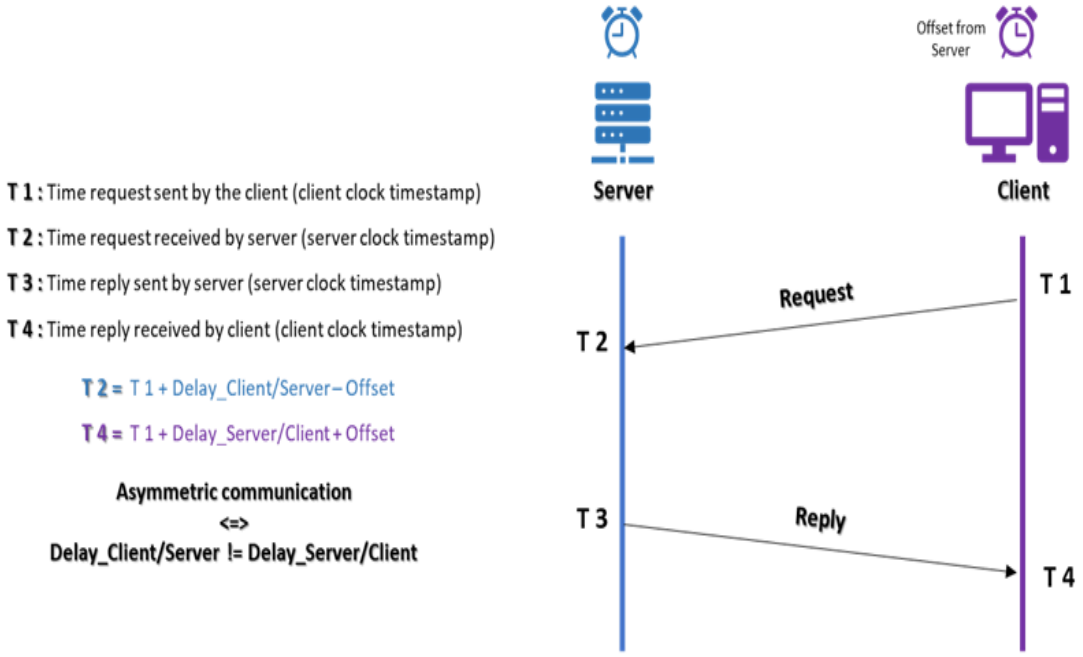


Figure 2.2 NTP Client-Server communication

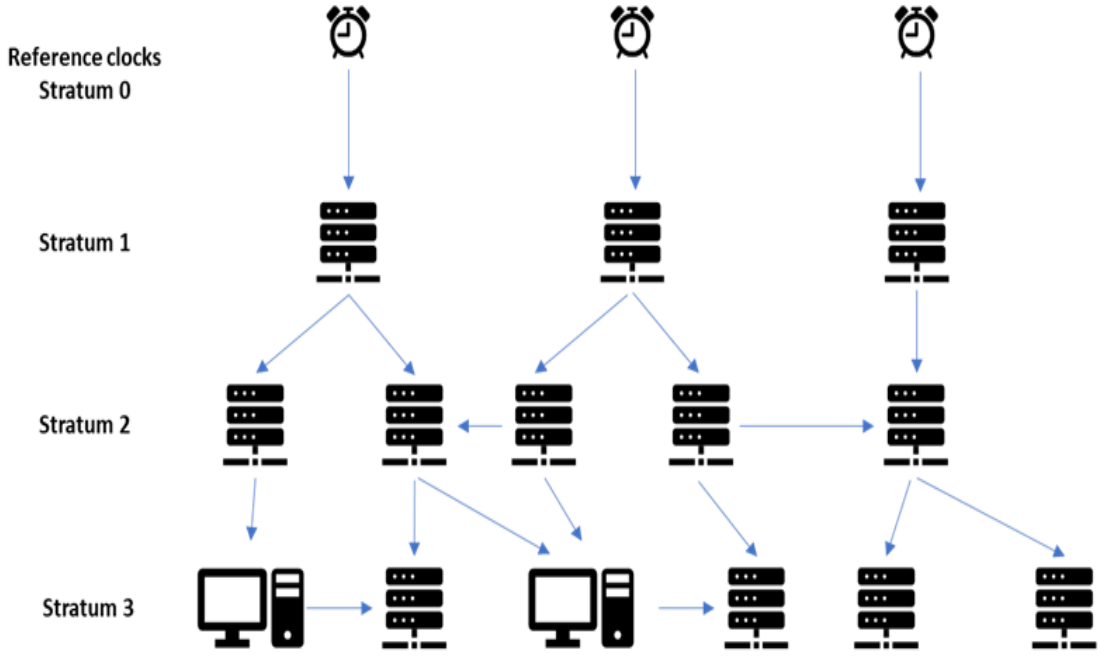


Figure 2.3 NTP Stratums

symmetric active, client, server, broadcast client, broadcast server). The most frequently used modes are client and server (modes 3 and 4).

- **Reference ID**: the meaning of this field varies from IPv4 to IPv6. For IPv4, the reference ID is the IP address of the server the NTP server is getting its time from. For IPv6, it is a 32-bit code identifying the server. Usually the first 4 octets of the MD5 hash of the NTP server's server's IPv6 address.

- **Timestamps**: there are 4 types of timestamps in an NTP packet, Reference, origin, receive and transmit, and are shown on the figure below as T1, T2, T3 and T4.

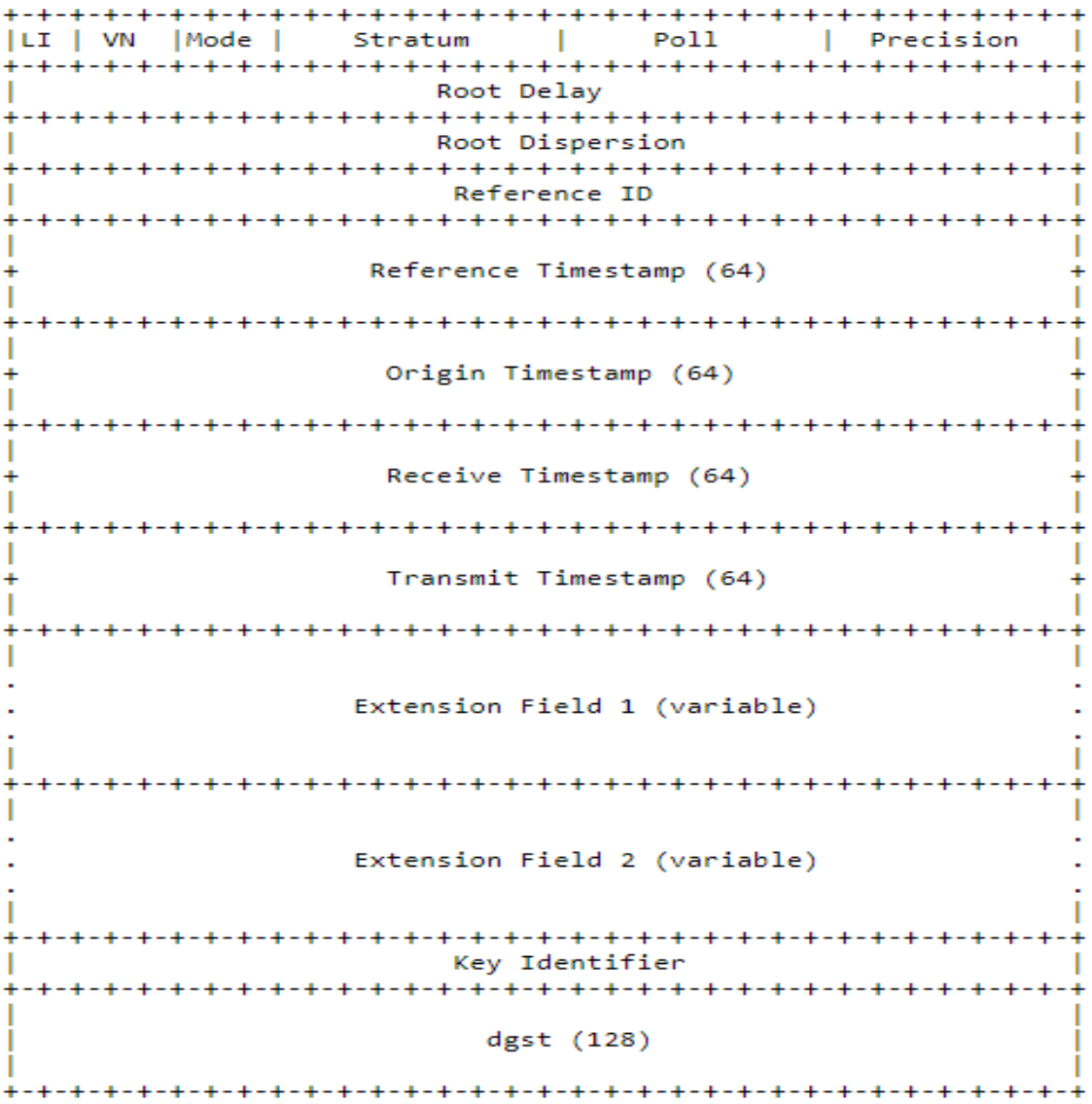


Figure 2.4 Typical NTP packet as presented in RFC5905 [30]

The polling interval of an NTP device is the time it takes for the device to request a synchronization cycle after the previous one is over. Most devices take between 64 seconds and 15 minutes to request time another time from their NTP server. According to the NTP RFC [30], this period can go to a maximum of 36 hours.

In order for an NTP client to accept time from an NTP server, 7 tests have to be done on the packet. The main tests are tests 2 and 3:

- **Test 1**: the packet is at best an old duplicate or at worst a replay by a hacker
- **Test 2**: the client checks the origin timestamp field on the packet received from the server with the transmit times tamp it sent the server to verify the validity of the packet. That is what makes the origin timestamp the NONCE element for NTP.
- **Test 3**: the packet is dropped if one or more timestamps are invalid
- **Test 4**: the access controls have the packet source "blacklisted"
- **Test 5**: the cryptographic message digest does not match the MAC
- **Test 6**: the server's source is not valid
- **Test 7**: one or more header fields are invalid

These two tests (among others) have been at the origin of Distributed denial of service (DDoS) attacks on NTP because of how vague their interpretation can be depending on the implementation of NTP and the OS distribution.

It is also interesting to make mention of the fact that NTP public servers as well as the protocol progression are managed by volunteers and researchers. This community provides public pools which are a gathering of NTP servers that are made available for private devices to synchronize to.

While the algorithm of NTP kept being developed and improved for better precision, another protocol was developed based on it: Simple Network Time Protocol (SNTP) in 1992 [31]. It consisted at first of a clarification of certain design features of NTP to allow operations in a simple and stateless mode. SNTP moved on to become one of the many implementations of NTP with fewer checks and a less complex synchronization algorithm. As the NTP became more complex depending on a much more complex algorithm [30] and getting to a precision degree of milliseconds to microseconds [32], SNTP kept its simplicity to become today known as the lighter version of NTP.

2.2.2 NTP vulnerabilities and attacks

The protocol that is used the most with IoT devices for reasons of ease of use and implementation is NTP and the sub-protocols based on it such as SNTP.

Deemed one of the Internet's oldest still in use protocol, NTP has been the subject of many security analyses since 1985 [33] [34]. This protocol has been improved into different versions through time, but it still proves to be vulnerable to certain attacks aiming to use the unauthenticated NTP such as the tool called Delorean developed by Selvi [35] that makes use of the unauthenticated NTP packets to attack certificate validity period in HTTPS.

Malhotra and al. [33] conduct an assessment of NTP security by applying a total of 6 attacks divided into two main categories:

- on-path attacks, where the attacker occupies a privileged position on the path between NTP client and one of its servers or hijacks traffic to the server;
- off-path attacks, where the attacker can be anywhere on the network and does not observe the traffic between the client and any of its servers.

NTP is also a protocol based on UDP using reserved port number 123. It is, therefore, connection-less and inherits vulnerabilities allowing attacks such as:

- **DoS / Distributed Denial of Service attacks (DDoS) / Flooding / Amplification attacks**: the most commonly used for NTP protocol. The attacker floods the victim device by sending a great number of packets consecutively or simultaneously, rendering the victim incapable of functioning correctly.
- **IP spoofing**: they are attacks that replace the source IP or destination IP addresses by another chosen by the attacker to disturb the normal use of NTP.
- **Replay attacks**: This attack usually relies on test 1 of the NTP algorithm. Because NTP can only keep up to the one previous synchronization cycle transmit timestamp, the device does not test for the timestamps before that. This can be used to replay the previous packets.
- **Person in The Middle (PiTM)**[1] **attacks**: An attacker that can have an idea of what is happening on the network and can sniff traffic on it is capable of using techniques such Address Resolution Protocol (ARP) spoofing/poisoning to tamper with NTP packets.

[1] Also known as Man in The Middle (MiTM).

- **Delay attacks**: attacks in which packets between the server and client are delayed for some time (can be either constant or can change) without tampering with the packet.

A detailed explanation of certain of these of these attacks will follow in chapter 4 as they are at the root of proving NTP weakness in IoT networks. The goals of these attacks are to either step time, hijack servers to create back doors, or to cause a DoS which can have a direct costly effect on services and companies especially with the rise of this malicious behavior since 2013 [36].

2.2.3 NTP security measures

The fact that NTP has been developed and deployed in traditional networks for over 30 years gave researchers time to study it more and introduce security measures for it that counter the current threats. That being said, there aren't many secure authenticated implementations of NTP that are in use today over the public NTP servers' pools.

RFC7384 [37] and RFC 8633 [38] discuss those measures in detail for traditional networks by adding encryption and authentication and authorization techniques to NTP for both masters and slaves. In addition, they refer to integrity protection to avoid packet delay and spoofing.

Certain research studies worked on developing new secure protocols based on the NTP with emphasis on the vulnerabilities such Authenticated Network Time Protocol (ANTP) discussed by Dowling and al. [39] based on two different authentication methods and part of today's NTP and Network Time Security (NTS) [4] which uses a form of Transport Layer Security (TLS) tunneling for regular NTP communications.

Authenticated NTP

ANTP or Authenticated NTP is today part of the NTP RFC authentication mode for version 4. it offers two types of authentication, Symmetric key and Autokey. Both of these have been proved vulnerable to attacks that can change the time or stop the time service.

- **Symmetric key**: This method, although secure, does not offer easy scalability. The NTP use of symmetric key must have the client and the server both mapped to the same key. Therefore, the addition of one client introduces changes on the server-side and thus, changes to all the other clients as well. Another downside of the use of this method is how organisms with working ANTP servers using it exchange symmetric keys with their clients. We have an example of that in NIST's instructions for using their

ANTP servers. Suffice to say that it requires fax or/and postal offices to exchange key information [40].

- **Autokey**: Also based on an implementation of symmetric keys. Each key is only valid for one packet. The server sends the key in an encrypted message to the client and it is from then on used as a cookie with the server signing the extension fields of the NTP packets. This approach has been proven to be vulnerable to DoS attacks as well as time tampering [41].

NTS - Network Time Security

Still in draft phase [4], NTS is a secure implementation of NTP. In its current mode, it depends on the unicast use of the protocol and focuses on working with the extension fields of NTP to protect the time information in the packets. It is also not as common yet and the first implementations of it are just coming out.

NTS is based on a combination of TLS and NTP and starts by a TLS handshake, followed by NTS client/server key establishment and then the NTS-secured NTPv4 communication happens. Although NTP itself is a connection-less protocol, the NTS key establishment stays on until the NTPv4 communication is no longer happening due to packets dropping or the connection is terminated.

The main idea behind these phases is to have a TLS tunneling with a first handshake for an NTP type of handshake while also the communications with the NTP packets are signed by the server. All these steps of security are put in place to avoid any IP fragmentation attacks during the transmission of large certificates.

2.3 Time synchronization in IoT devices

This final section discusses the existing achievements in the establishment of time synchronization protocols and methods for connected smart devices. It outlines how these methods came to be, their background in relation to the existing protocols used in traditional networks described above as well as the security level they are aiming for.

2.3.1 Time synchronization methods and protocols in IoT

Similar to how the majority of IoT protocols came to be, time synchronization protocols are inherited from traditional networks and used as or developed in lightweight versions to

be more accessible for devices with lower CPU power. Most of these methods are clock synchronization methods.

These protocols can be summarized in what follows [42] [43] [44]:

- Network Time Protocol (NTP)
- Precision Time Synchronization Protocol (PTP)
- Global Position System (GPS)
- Reference Broadcast Synchronization (RBS): a method in which the receiver uses the physical layer broadcasts for comparing the clocks.
- Timesync Protocol for Sensor Networks (TPSN): aims at providing network-wide time synchronization in a sensor network. The algorithm works in two steps. In the first step, a hierarchical structure is established in the network and then a pair wise synchronization is performed along the edges of this structure to establish a global timescale throughout the network.
- Tiny-Sync and Mini-Sync: two lightweight Synchronization algorithms
- Lightweight Treebased Synch (LTS)
- Flooding Time Synchronization Protocol (FTSP)
- Energy-Efficient Time Synchronization Protocol (ETSP)
- Maximum consensus-based time synchronization (MTS)

2.3.2 NTP and time synchronization protocols security solution in IoT

A critical issue regarding the time synchronization methods described above is their heterogeneity and dependence on specific vendors and developers. This makes them hard to adapt to all architectures and networks.

To our knowledge, there has been no adaptation of any of the current secure versions of NTP (eg. NTS, ANTP) to IoT platforms.

The way the IoT world is evolving makes standardization very hard to achieve although it is the closest way possible to having secure protocols and stacks. Most work being done today mostly concerns coming up with new methods and ways of security instead of assessing the existing protocols and adding components to them to have a more stable secure environment.

The following includes a list of variable protocols and architectures developed for IoT devices for a more secure time synchronization:

- Gore and al. [44] suggested software-based Kalman filter-based synchronization scheme on an industrial platform to avoid existing protocols containing security issues;
- He and al. [19] Defined a new secure protocol based on MTS called secured maximum consensus-based time synchronization (SMTS) protocol which solves the issue found in MTS of not detecting message manipulation attacks;
- Liu and al. [20] proposed a protocol where a broadcast synchronization packet makes all sensor nodes in the network synchronize with the trusted source while taking into consideration the delay due to these sensors not being capable of doing all synchronization in time, similar to traditional networks.

The common factor between these protocols is the recognition of issues in existing protocols and the creation of new ones instead of working on developing the preexisting time synchronization protocols.

IoTs in general and time synchronization specifically suffers from a standardization issue, as discussed by Barrera and al. [45]. This leads to the existence of methods and protocols that are not necessarily compatible with each other and are in most times specific to products created by specific companies. This variety goes on to create even more security issues the industry does not invest in solving as it efficiency and portability overtake security in IoT networks.

2.4 Conclusion

In this chapter we have reviewed literature discussing the dependence of IoT networks and devices on the correct use of time synchronization methods, especially NTP protocol. We have also defined concepts related to NTP and IoT security. It is fairly clear that the research done in this field can be pushed further to have a more inclusive security of correct time synchronization depending on NTP in IoT networks.

CHAPTER 3 EMPIRICAL ANALYSIS OF NTP USE IN IoT NETWORKS

In this chapter, we discuss the way IoT devices use NTP protocol and the repercussions of such use from a security point of view. We base our analysis on network traces collected from a data set of 50 devices. We study the possible patterns these devices follow in choosing their NTP servers and network ports. We also analyze some NTP packets' components as well and their relationship with TLS and DNS packets.

3.1 Dataset presentation

The main dataset used for the analysis of the empirical use of NTP by IoT devices is the one provided by Alarwi and al. [11]. It consists of 50 connected devices 46 of which are either class 1 or 2 [10] which we focus on for the analysis. They are devices with lightweight use of network stack at different levels as well as a possibility for security measures implementation to limited extents as explained in the RFC7228.

The authors also provide PCAP captures, which are network traces stored in PCAP format, over a period of 13 days (March 20, 21, 25, 2018 and April 10-19 of 2018). It is a total of 3744 files and 303 Gigabytes of data. Based on these PCAP files we can filter the NTP communications happening between the IoT devices and the servers and analyze data such us NTP versions, protocol ports and servers used. This information will provide a baseline NTP behavior of medium sized IoT network. Using this behavior, we can then infer communication patterns which will help inform the proposed solutions in chapter 5.

The dataset is also characterized by the variety of types of IoT devices used (such as light-bulbs, hubs, voice controllers and TVs) as well as the companies behind them which also helps in providing a more generic analysis.

For a more advanced application of the vulnerabilities and to showcase the patterns that we have noticed we will work on, in addition to the dataset provided, virtual machines that are Linux-based and similar to most IoT devices as well as a few real IoT devices for tests. These environments will be introduced in the following chapter.

3.1.1 Identifying devices that use NTP in the dataset

We collect all the 288 PCAP files of each day in one file and apply a protocol filter in Wireshark, a packet analyzer/sniffer, named *ntp* to show only NTP communications between the devices in the network. We collect the result in the form of CSV files that we then filter

to only contain iterations of devices with the IP address starting with *192.168.0.* as it is the subnetwork we are studying.

The only devices left from this analysis are the devices using the protocol NTP in their communications on that day. We collect them on a list and compare them to the devices identified in the network by Alarwi and al. Table 3.1 provides an overview of the dataset as well the use of NTP by its devices.

We notice that out of the 50 devices present in the table, only 7 devices do not use NTP to synchronize their time. it is an average of 86% of the total devices that use this protocol and it can be explained by the previously stated popularity of it in networks. Thus another reason that shows how important to analyze its security in IoT networks as well.

It is fair to say that there are still a few missing pieces in the PCAP files such us certain DNS queries as well the constant reappearance of a specific device (192.168.0.200) that was later explained to be a tablet containing the applications to control the IoT devices. Certain IoT objects connect and sync to it as if it were an NTP server.

3.1.2 NTP version use in the dataset

following the same analysis used in identifying which devices use NTP, we identify which version of NTP every device uses in its synchronization requests. Wireshark gives an overview of this field as well because it is part of the NTP packet.

We represented the results in table 3.1.

Most devices use either version 3 or version 4 as they are the most common versions and very similar. The only additions in version 4 are the compatibility with IPv6 environments and the corrections of certain bugs.

We also notice that there are two devices, Bose SoundTouch 10 and Koogeek light bulb, that are using NTP version 1. It is an obsolete version of the protocol along with version 2. Being the 1st iteration of a protocol that dates more than 3 decades ago, it is prone to many security issues and it is not capable of either asymmetric or symmetric authentication methods.

We tested a new Koogeek light bulb for more information about the use of this version. This new bulb does not use the protocol NTP anymore.

Table 3.1 Dataset analysis

IP ADDRESS	DEVICE NAME	IoT CLASS	NTP	NTP version
192.168.0.2	Google OnHub	2	No	/
192.168.0.4	Samsung SmartThings Hub	2	Yes	3
192.168.0.5	Philips HUE Hub	2	Yes	3 and 4
192.168.0.6	Insteon Hub	2	Yes	3 and 4
192.168.0.7	Sonos	1	Yes	3
192.168.0.8	Securifi Almond	2	Yes	4
192.168.0.10	Nest Camera	1	Yes	4
192.168.0.12	Belkin WeMo Motion Sensor	1	Yes	3
192.168.0.13	LIFX Virtual Bulb	1	Yes	4
192.168.0.14	Belkin WeMo Switch	2	Yes	3
192.168.0.15	Amazon Echo	2	Yes	4
192.168.0.16	Wink Hub	2	No	/
192.168.0.18	Belkin Netcam	1	Yes	3
192.168.0.19	Ring Doorbell	1	Yes	4
192.168.0.21	Roku TV	2	Yes	3
192.168.0.22	Roku 4	2	Yes	3
192.168.0.23	Amazon Fire TV	2	Yes	3
192.168.0.24	nVidia Shield	2	Yes	3
192.168.0.25	Apple TV (4th Gen)	2	Yes	4
192.168.0.26	Belkin WeMo Link	1	Yes	3
192.168.0.27	Netgear Arlo Camera	1	Yes	3
192.168.0.28	D-Link DCS-5009L Camera	1	Yes	3
192.168.0.29	Logitech Logi Circle	1	Yes	3 and 4
192.168.0.30	Canary	2	Yes	3
192.168.0.31	Piper NV	1	Yes	4
192.168.0.32	Withings Home	1	Yes	3
192.168.0.33	WeMo Crockpot	1	Yes	3
192.168.0.34	MiCasaVerde VeraLite	2	Yes	3
192.168.0.35	Chinese Webcam	1	Yes	3
192.168.0.36	August Doorbell Cam	1	Yes	3
192.168.0.37	TP-Link WiFi Plug	1	Yes	4
192.168.0.38	Chamberlain myQ Garage Opener	1	Yes	3
192.168.0.39	Logitech Harmony Hub	2	Yes	3
192.168.0.41	Caseta Wireless Hub	1	Yes	4
192.168.0.42	Google Home Mini	2	Yes	3
192.168.0.43	Google Home	2	Yes	3
192.168.0.44	Bose SoundTouch 10	1	Yes	1
192.168.0.45	Harmon Kardon Invoke	2	Yes	3
192.168.0.47	Apple HomePod	2	Yes	4
192.168.0.48	Roomba	1	No	/
192.168.0.49	Samsung SmartTV	2	No	/
192.168.0.50	Koogeek Lightbulb	1	Yes	1
192.168.0.51	TP-Link Smart WiFi LED Bulb	1	Yes	4
192.168.0.52	Wink 2 Hub	2	Yes	4
192.168.0.53	Nest Cam IQ	1	No	/
192.168.0.54	Nest Guard	2	Yes	4
192.168.0.113	Ubuntu Desktop	Desktop	No	/
192.168.0.138	Android Tablet	2	Yes	4
192.168.0.151	iPhone	2	No	/
192.168.0.159	iPad	2	Yes	4

3.2 NTP importance in IoT networks

Time correctness is essential to several processes for networks and especially IoT networks. Our analysis of the IoT devices in the dataset as well as the general spectrum of connected smart devices led us to identify 3 main uses for Time synchronization and NTP in IoT devices. This analysis was based on identifying and categorizing IoT devices into 11 fields with a total of 53 types of devices. We focused on their dependence on time accuracy, how it is used, and a possible risk scenario if the time is incorrect. The main important tasks depending on time concluded from this study are:

- **Scheduling tasks:** Certain IoT devices offer scheduling services to program them for specific tasks to be done at times. This helps avoid manually activating them for either distance, comfort, or optimization reasons. the importance of time correctness shows in the validity of the schedules and them being applied on the real time. This can be seen in the behavior of certain devices such as connected light bulbs, connected switches, industrial IoTs, connected traffic lights or even connected infusion pumps. Thus, the necessity of applying schedules on correct times varies depending on the purpose of use of the device as well as the field it is used in.

- **Authentication / encryption / networking protocols:** Certain protocols depend on timestamping to function correctly. They use time to verify components' validity such as digital certificates and queues. We can see that in the behavior of protocols like TLS, HTTPS and Message Queuing telemetry transport protocol (MQTT). The fact is, unlike regular networks, IoT devices rarely have an expected behavior when it comes to functioning with digital certificates or encryption protocols for which the time validity window has expired. When a regular network service would stop, according to experiments we have conducted, for IoT devices this could either stop the device's service, continue its use as if nothing has changed or continue working with certain hindrances. All these states were analyzed on the same device in different situations explained later on in our work. This exposes another important role correct NTP synchronization plays in keeping an IoT device functioning correctly as well as the importance of keeping encryption data up to date.

- **Logs / monitoring:** For a number of IoT devices keeping timed logs of tasks, requests and general behavior is part of the OS as well as the function of the device. Hubs for example, keep timed logs of the tasks they accomplish. Another example of this is connected wireless cameras, which are used for monitoring and for which the time information is important in keeps logs of video saved. If we have a camera for which

the time is incorrect or has been shifted purposefully, it could lead to errors in decision making.

3.3 Analysis of NTP communication in IoT devices

3.3.1 Interplay between NTP and DNS

Devices using NTP for time synchronization need to access time servers over the internet or local time servers defined in an internal network in order to obtain an authoritative reference time. The most commonly used public pools of NTP are made available over *pool.ntp.org* or *time.nist.gov*. The first being a public pool of NTP servers managed by volunteers and the second is owned and managed by the National Institute of Standards and Technology (NIST). In addition to these pools, certain vendors make available their own public NTP servers for users to connect to as shown in table 3.2.

Table 3.2 Examples of public NTP vendor pools

Vendors	**Time server**
GOOGLE	time.google.com
MICROSOFT	time.windows.com
APPLE	time.apple.com
CLOUDFLARE	time.cloudflare.com
FACEBOOK	time.facebook.com
HUAWEI CLOUD	ntp.myhuaweicloud.com

In order for a device to reach these servers, NTP implementations usually offer the NTP configuration files where the implementation of the protocol can be instructed to reach specific pools.

For these connected devices to be able to know which NTP servers they can contact as well as their IP address, they send a query to the DNS server for the IP address of the pool and receiving a response with the IP address of one or many NTP servers in the said pool to choose from.

Figure 3.1 shows an example of a connected light bulb requesting the IP address of *pool.ntp.org* and receiving 4 IP addresses of NTP servers available in it.

This light bulb then uses one of the servers IP's received from the DNS query for it's NTP time synchronization as shown in figure 3.2.

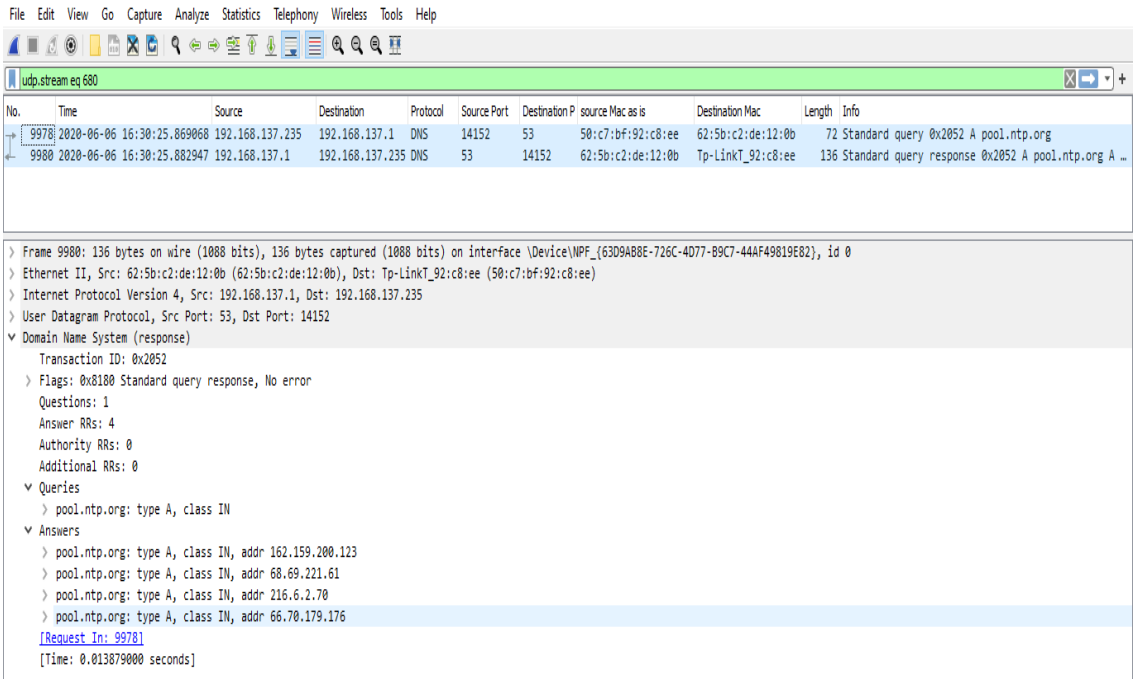


Figure 3.1 IOT DNS query and response for *pool.ntp.org*

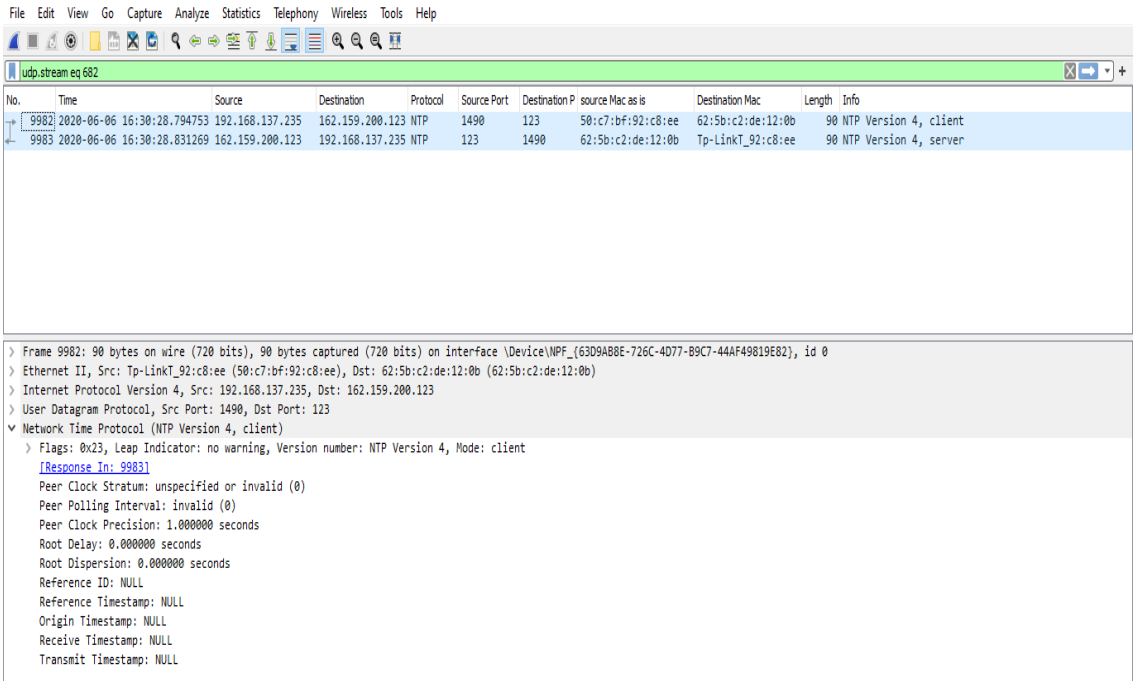


Figure 3.2 IOT NTP query and response for *pool.ntp.org*

Note that devices may use different algorithms for selecting which server IP to use. A popular approach is to use thee first returned IP, but devices may also randomly select a server from the list or change servers after a few synchronization cycles.

In traditional networks, in addition to the use of DNS by NTP, DNS also relies on NTP to keep the cache information up to date. Attacks about time shifting affecting DNS have been studied in works such as the cache expiration attack and cache sticking attacks [46].

3.3.2 Real time clock in IoT devices and its effect on NTP communication

Real time clocks (RTC) are hardware components, as shown in figure 3.3, used to keep track of the current time in devices such as computers. Their size makes it difficult to include them in smaller IoT devices.

The RTC is used as a reference clock in devices to save time after the device is powered off. When the device is powered back on, the time kept by the RTC is used to compare it with the new time the device gets from synchronizing to an NTP server. If the time is coherent, within a few seconds of difference depending on the OS used, the device continues with the time it gets from the NTP server.

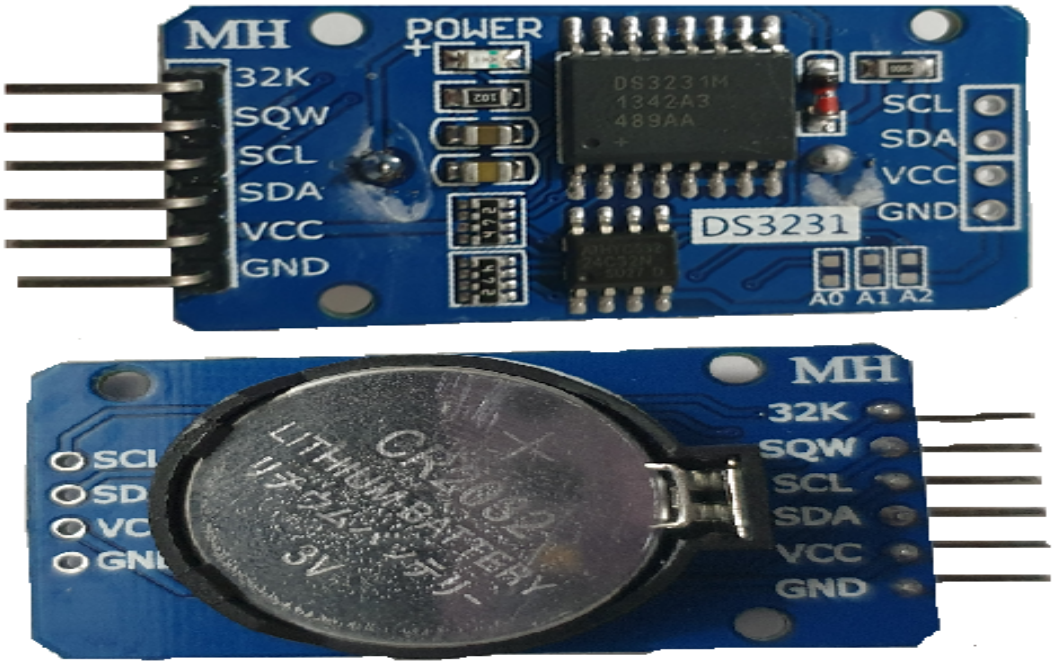


Figure 3.3 Real Time Clock (RTC) hardware

If the time the device's RTC is showing and the time it receives from the NTP server are different by an unacceptable margin, the device keeps sending time synchronization requests

until a sufficient number of responses is showing the same time. Each implementation of NTP uses a different algorithm in synchronizing time and comparing it to the RTC time. they differ in the accepted time difference between NTP and RTC and they also differ in the number of synchronization cycles or servers needed to accept a different time than the one saved in an RTC.

Thus, the more precise an RTC in a device is and the more complex the algorithm of new time acceptance is, the more secure time synchronization on a device is either in regular or IoT networks. This translates in network traces of NTP communications by the RTC populating the *reference timestamp* (the last time at which the local clock was last corrected) and *transmit timestamp* (the time at which the query was sent). the *transmit timestamp* is then used by the server as an origin timestamp and it is also used to authenticate which device the response belongs to.

The absence of this component can be noticed in the form of a **NULL** value in all the timestamps in NTP packets coming from IoT devices. Figures 3.4 and 3.5 show how a device with no RTC, in this case, a smart TP-link bulb, uses NTP timestamps in its communication with the NTP server of its choice.

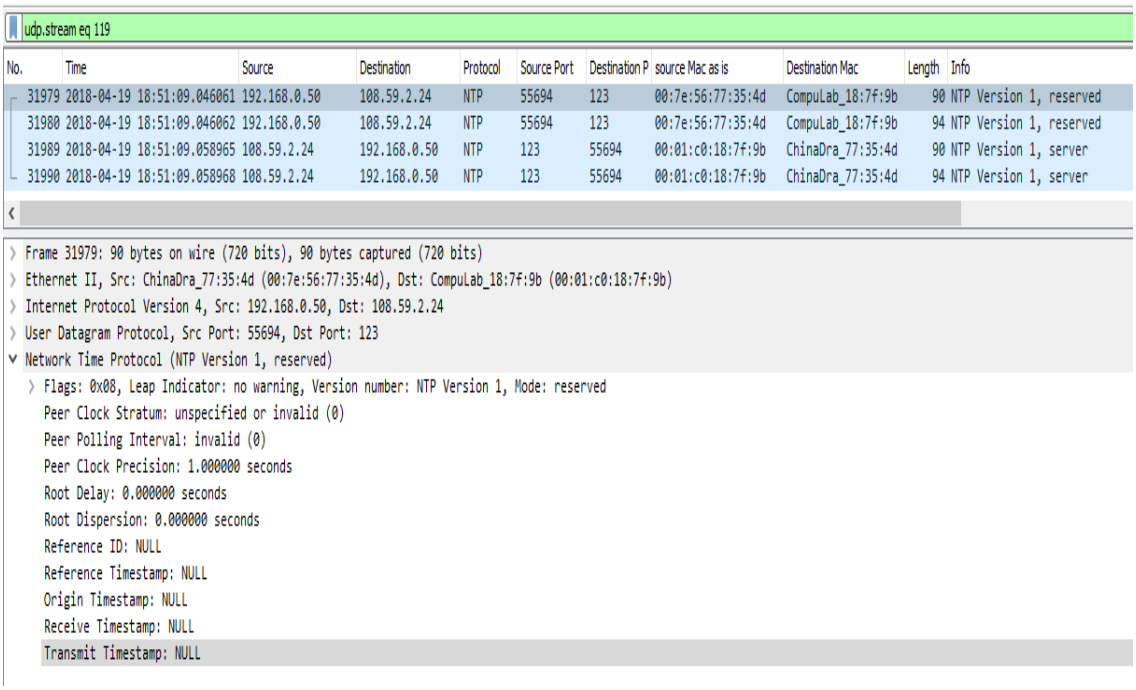


Figure 3.4 NTP request of a TP-Link smart WiFi LED bulb with no RTC showing a NULL value in the transmit timestamp field

The algorithms that are used in NTP communications that compare time kept by RTCs on

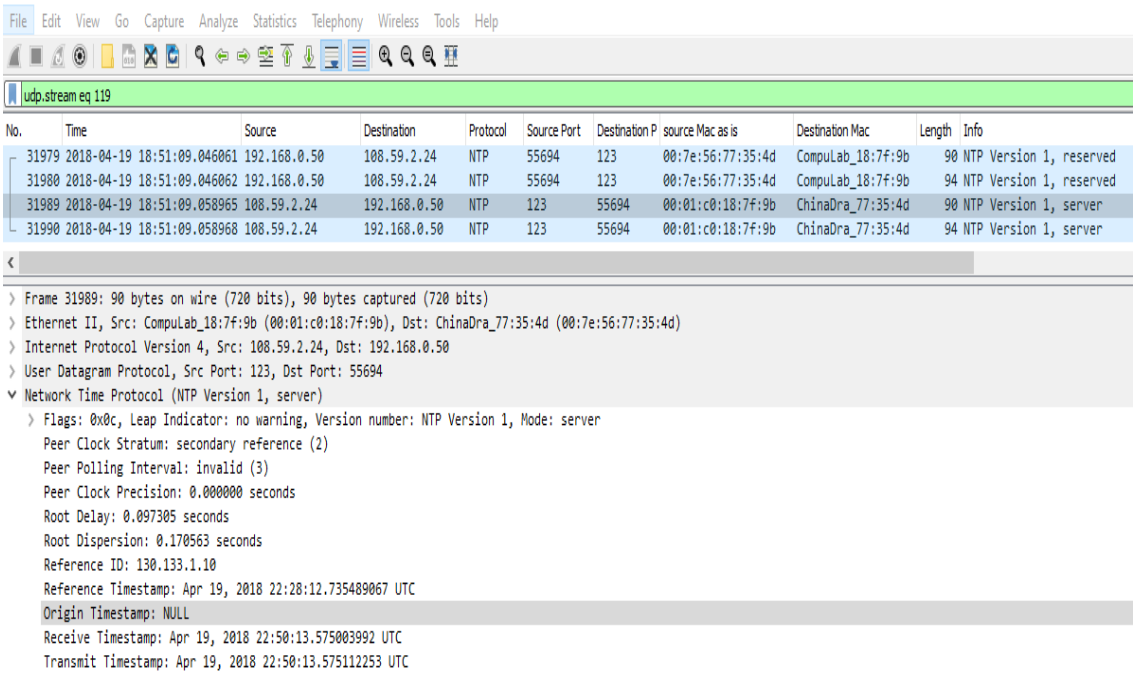


Figure 3.5 NTP response of a TP-Link smart WiFi LED bulb with no RTC showing a NULL value in the origin timestamp field

devices containing this component to compare the time sent by the server with the local time cannot be used in this case. And since there is no transmit timestamp from the client's request, the response does not contain an origin timestamp and therefore no check is done on the server from which time is received by the IoT client. This also means that the client will accept any time provided by the server is it connected to the first time it receives since it has no other time to compare it to.

The nature of this type of synchronization where any time value received by the client IoT is considered correct can lead to making time shifting attacks easier. There are no additional checks that will be done and any incorrect time value sent by an attacker that has succeeded to infiltrate the NTP communications will be accepted from the first try.

3.3.3 NTP port analysis

Seeing as the concept of the five-tuple (network protocol, source and destination IP and ports) is one of the main mechanisms used to identify a network flow, it is a major security interest to assess the behavior of connected devices from the dataset regarding the patterns IoT devices follow in choosing its 5 components.

Because the protocol studied is the connection-less protocol NTP, the source IP is the device,

the destination IP is an NTP server and the destination port is always 123 (the default port for NTP communications). From a security perspective, poor entropy in source port selection can create vulnerabilities. We are thus interested in analyzing the behavior of port selection on NTP use in IoT.

If the NTP source port is predefined on the connected device, added to the UDP nature of NTP, it can be the cause of successful blind attacks seeing as the last variable in the 5-tuple is now constant. A blind attack is one where it is not needed to see the network packets to successfully tamper with the network the way the attacker intends to. If the attack is based on spoofing the IP address and the source port, which is now fixed, it is easy to achieve.

We focused on the dataset from the work of Alarwi and al. [11]. The 50 devices studied through a period of 13 days gave way to 13 (1 per day) folders of 288 PCAP files of 5 minutes each. It is a total of 303 Gigabytes of data. Our purpose is to have a clear graph of a pattern of choosing the NTP network port for each device in the dataset that uses NTP for time synchronization. The methodology we followed to achieve our purpose was the following:

- We used a packet analyze t-shark[1] to extract only the NTP packets in all the PCAP files. this was done in every folder for every day separately by using these bash commands:

```
#mergcap *.pcap -w mergedday1.pcapng
#tshark -r mergedday1.pcapng -w day1.pcap ntp
#tshark -T fields -n -r day1.pcap -E separator=, -e ip.src -e
   udp.srcport > day1.csv
```

  the first line merges all the PCAP files existing in the folder in one folder. The second command applies an NTP filter on the merged PCAP file and prints it into a new one called day1.pcap. the last command prints out the NTP packets in the day1.pcap file in a CSV file showing source IP and the source port only of every packet filtered. We apply this process on each folder of the 13 captured days to end up with 13 CSV files of all the NTP communications between the network's devices.

- We then use Office Excel to merge all the 13 CSV file into one file. On this file we apply a filter to only keep the packets with the IP source addresses included in our list of devices. we then apply another Excel filter to organize the devices so that we can have all the packets from the same device/IP address grouped with under one another. This file, named ports.csv, contained a total of 25780 rows with a total size of 1.7 Megabytes. It had NTP packets from the 39 devices that used NTP out 50 devices in the dataset.

[1]https://www.wireshark.org/docs/man-pages/tshark.html

- In order to have a more meaningful presentation of the ports of choice for each device, we used the BASE function in Excel to convert the port numbers into 16-bits binary values (0-65535). We then split this column into two columns, the first containing the higher 8-bits of the binary 16-bits data and the second column contain the lower 8-bits. We then changed each column's values back to decimal.

- We used bash commands to split our ports.csv file into different files for each device

```
awk -F\, '{print>$1}' ports.csv
mmv 192.168.0.\*\#1.csvls
for i in *.csv
  do
  sed -i !i"ip,port,high,low" $i
done
```

  This created 39 different files for the 39 different devices using NTP.

- In order to graphically show the results of port choices and to decide which patterns we can view, we use a python script that we applied on the 39 files.

```
for filename in glob.glob('*.csv'):
  data = pd.read_csv(filename)
  plt.scatter(data.high, data.low)
  plt.title(filename)
  plt.xlabel("high")
  plt.ylabel("low")
  plt.xlim(0,256)
  plt.ylim(0,256)
  plt.show()
```

  the results from this script gave us a total of 39 graphs showing the ports each device picked for its NTP communications

After manipulating the CSV files, we observed that the source port allocation strategies for each of the devices fell into one of 4 patterns/categories. Figure 3.6 shows these patterns.

The devices' distribution according to their pattern of port choice is represented on the table 3.3.

All the graphs in Figure 3.6 consist of patterns shown only on the right half. That can be explained by the definition given to ephemeral ports by the Internet Assigned Numbers

Table 3.3 Patterns of dataset devices' choice of NTP UDP ports

Patterns of ports choice		*Devices*
Multiple ports during synchronization period	Random port number for every request	192.168.0.159
		192.168.0.54
		192.168.0.52
		192.168.0.47
		192.168.0.43
		192.168.0.42
		192.168.0.34
		192.168.0.31
		192.168.0.15
		192.168.0.10
		192.168.0.5
	Random port every few requests	192.168.0.138
		192.168.0.46
		192.168.0.45
		192.168.0.44
		192.168.0.41
		192.168.0.33
		192.168.0.26
		192.168.0.23
		192.168.0.18
		192.168.0.8
	Random first port choice with a predefined pattern in choosing the next port	192.168.0.50
		192.168.0.13
One same port during the whole synchronization period	All NTP requests using the UDP source port 123 (2+)	192.168.0.29
		192.168.0.25
		192.168.0.14
		192.168.0.12
		192.168.0.6
	All NTP requests using one random port number (2+)	192.168.0.51
		192.168.0.37
		192.168.0.19
	One NTP request for the whole time of use of the device using a random port number	192.168.0.36
		192.168.0.35
		192.168.0.32
		192.168.0.28
		192.168.0.27
		192.168.0.24
		192.168.0.7

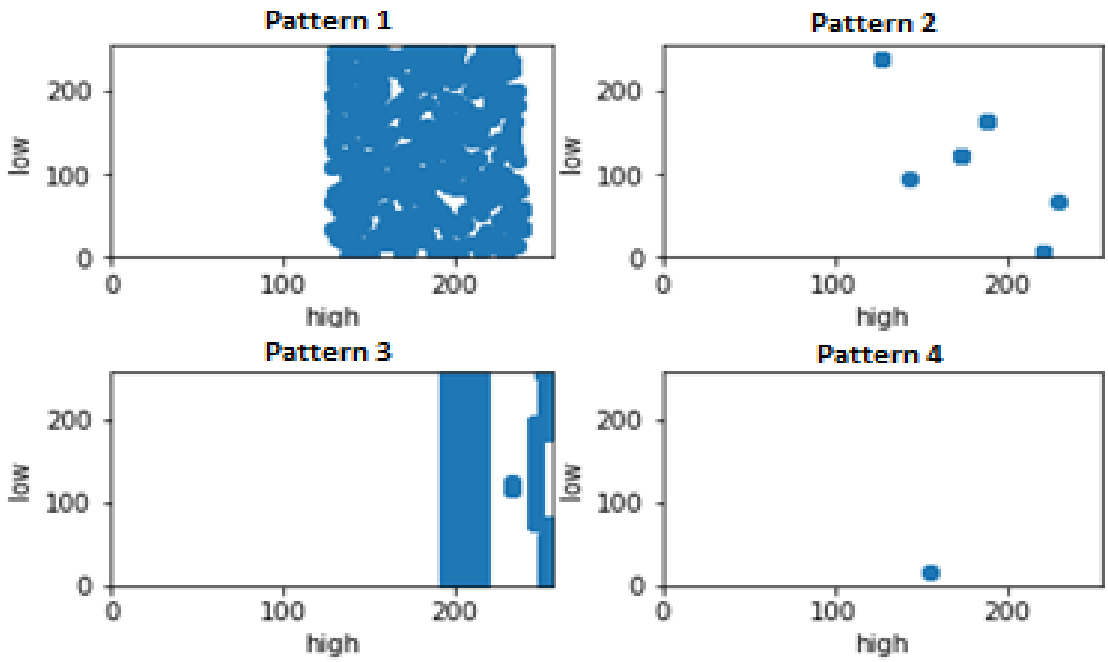


Figure 3.6 Main port patterns in the dataset

Authority (IANA) as well as the ranges chosen for best practices. The dynamic or private ports are from port number 49152 to 65535. The registered ports start from 1024 and include ephemeral ports. Port UDP 123 is part of the well-known ports (1 to 1023)[2].

- **Pattern 1**: Random port number for every request

  The devices using this type of pattern for their choice of the source port number for NTP requests rely on a randomization algorithm, making it almost impossible to predict the ports used in subsequent NTP (or other communication) requests. This results in an added layer of security stopping attackers from having an easy way to spoof replies to the victim's requests if it does not use a common port number such as UDP port 123

- **Pattern 2**: Random port every few requests

  Here we have devices that do depend on a level of randomization for their choice of source port for NTP requests except, for a number of successive requests (in the case of this graph: every 20 to 25 requests), the devices keep using the same port number. An easy way for the attacker to counter the randomization used here is by monitoring the traffic and choosing the right time to attack using the port number already picked by

[2]Network port number 0 is a reserved system port not to be used by TCP or UDP messages

the device, for example right after the first request after switching to another random port number.

- **Pattern 3**: Random first port choice with a predefined pattern in choosing the next port

  This pattern is the least seen in the dataset (only 2 devices out of 39) . It is mainly only hard to guess the device port at the beginning. The security here depends on the randomization of the first port choice and the complexity of the relation between it and the next one. For the example picked here, device 50 chooses port number 63049 and adds 1 to port number after every request. If an on-path attacker is capable of monitoring such traffic, it is easy to predict the port of choice for the next request and thus, the possibility of performing a blind attack using this port number is high.

- **Pattern 4**: Using a single port for all NTP communication

  This type of pattern can be divided to 3 sub-patterns:

  - 1 NTP request for the whole time of use of the device. In which case the attacker can try and use the same port number from the request seeing as the device will not be changing this port later on as it is relying solely on the time fetched from the first and only request and its internal clock. These devices run the risk of time drifting naturally which is a security issue if they have services depending on time accuracy such as logs (Netgear Arlo Camera for example uses this pattern of synchronization through one NTP request using UDP source port number 46757)
  - All NTP requests using the UDP source port 123 (Belkin Wemo Motion Sensor for example). Other than the lack of randomization in this case, an additional threat would be relying on the use of port 123 as it is an easy target for attackers looking into NTP communication. It highly reduces the efforts they need to produce a successful blind attack
  - All NTP requests using one random port number. This is a mix between pattern 2 and pattern 4b as it can be easily used by the attacker by identifying the port number after the first request and the lack of randomization afterwards leaves space for successful but time and effort consuming attacks.

3.3.4 NTP server analysis

In this section we analyze the behavior of our dataset in regard to the devices' choice of NTP servers to synchronize their time. We also discuss the regional distribution of NTP servers

present in the pool *pool.ntp.org*.

Regional distribution of NTP servers

Through analyzing the use of NTP servers in the NTP ecosystem [47], it is clear that the *pool.ntp.org* is used by most open-source operating systems as well as well-known vendors for regular and IoT networks.

We conduct a collection of server IPs on this pool to assess which areas in the world have the most servers as well as how responsive the pool is. to do so we create a file with the domain names (see APPENDIX B) from the pool. Each pool is in the form of *x.region.pool.ntp.org* with x being a number from 0 to 3 and run a continuous dig command on those names until we collect the total servers. It took a total of 2 days to collect all the servers present in the pool at the time of the study.

The script we run to collect server domains and IP addresses is:

```
for j in {1..2}
do
     echo  $j
     for N in $(cat servernames.txt);
     do printf "$N is %s\\n" $(dig +short "$N");
     echo;done >> serverips.ods
     sleep 3
done
```

We collected a total of 4205 server IPs belonging to 4 domains (0, 1, 2 and 3) in each of the 6 regions of the NTP pool project (Africa, Asia, Europe, North America, Oceania and South America). Figure 3.7 shows the distribution of the collected servers in the different regions and how certain regions share a number of servers.

Europe and North America show the biggest concentrations of NTP servers which can be explained by the presence of more vendors and IT companies. We also notice that all regions share some servers except Africa.

We then compared the existence of these servers in the different regions according to their domains. We used the *Show Me* add-on in the tool *Tableau* to draw a network graph presented in figure 3.8 based on the data collected from running our script between NTP pool domains and regions.

This figure confirms the results from Figure 3.7 as it shows the difference of server numbers

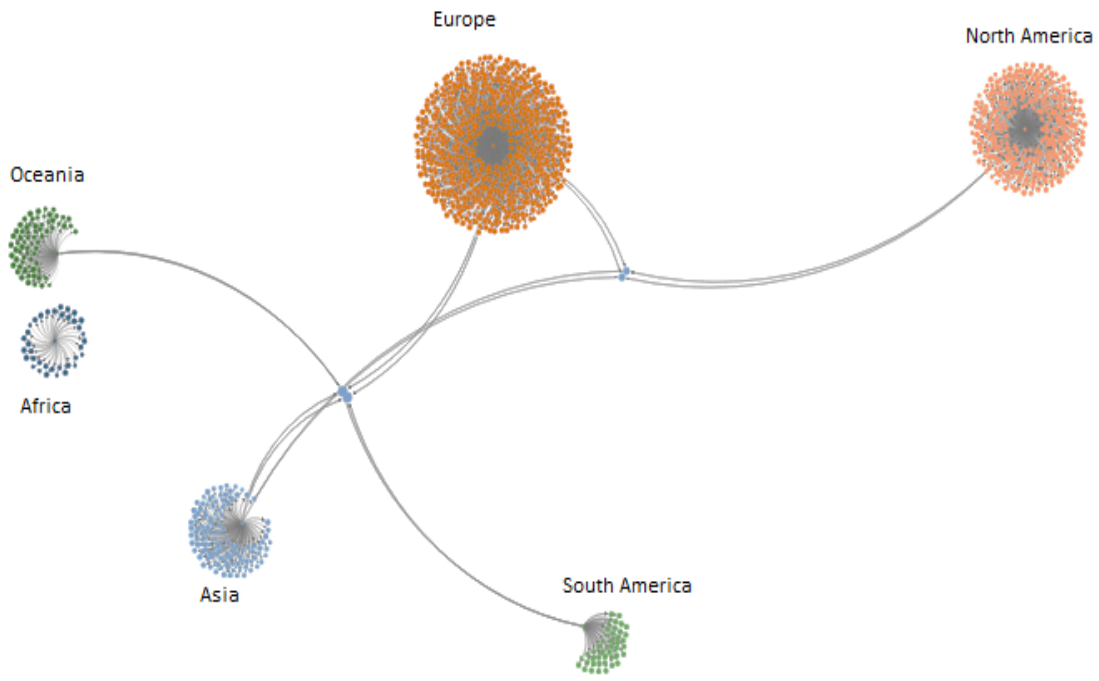


Figure 3.7 Distribution of NTP pool project servers in the 6 regions

between Europe and North America and the rest of the regions. This can lead to an intensive use of the servers existing in the other regions as the number of IoT devices and regular devices is increasing constantly. NTP pool project links the device requiring synchronization with the closest available server in the pool. More research needs to be done in cases of servers not available to give responses to client queries.

Choice of NTP servers by dataset devices

Being part of the 5-tuple, NTP servers are a main component of the security analysis of NTP communication with IoT devices. Based on the dataset, here is the summary of the state of NTP server use on IoT devices:

- All NTP-capable devices are using publicly available NTP. They are either being made available by private companies, NIST or *pool.ntp.org*. Making them reachable by the public can cause these servers to be the target of attacks such as DoS attacks that render a specific server unreachable by other devices. In case this server is the only one hard coded in a device, it can push the device to drift in time until the NTP server is back on.

Figure 3.8 NTP pool regional distribution of servers over domains

- 15 out of the 50 devices use NTP servers from the public NTP pool *pool.ntp.org*. Which can lead to a smaller number of NTP servers to target as this pool contains around 4000 servers (servers are constantly added and removed as they are managed by volunteers). They can be tampered with to either deny all time synchronization services they provide or push them to deny service for specific devices by using the example of an attack explained by Malhotra and al. [33]. This attack uses the Kiss of Death (KoD) packet to force the client to stop sending time requests to its server. The kiss of death is packet made to help reduce the rate at which certain clients query one server. Once a client is known to do so too frequently, the server sends a KoD packet stopping the client for a duration of 2^{Poll} seconds, where $Poll$ is the poll value defined on the received NTP KoD packet.

- 10 devices use the 192.168.0.200 device, their gateway, as an NTP server. This is a Tablet that is contains some of the applications needed to control certain IoT devices in the dataset. Using this device to synchronize time adds another stratum for NTP communications. An additional stratum induces additional network delays as well as a bigger offset compared to the precise correct time. Although this can range from a few milliseconds to seconds, it is a delay that can be relatively important depending on the situation. In case of a delayed traffic light, taking into consideration the human reaction time of the order of 1.5 seconds seconds [48], the few seconds of delay can be the cause for a life-threatening accident.

- 8 devices used servers that are no longer reachable. 2 devices used a previous NIST NTP server (not available since 2012). If these servers were the only ones coded in the IoT device, not being able to reach them can lead to the device depending on its local time. The device will then have a time precision depending on the precision of its reference clock and will be left to drift in time until the user realizes that.

- None of the devices used a server that requires authentication. The packets exchanged between the server and client in this can easily be spoofed and the many attacks become possible. We investigate the impact of this security issue further in chapter 4 through applying attacks that are only feasible if there is no authentication in NTP communication.

- Certain devices synchronize to servers that are present on more than one domain even though they are closer to one domain that contains many servers. There are two possibilities in this case, either the server belongs to both domains or the server is hard coded in the device. If the server is hard-coded into them it could lead to issues

explained in previous remarks. An example of these devices is the MiCasaVerde device in figure 3.9. On the opposite side, we have devices that only call for servers belonging to one domain only such as the nest camera in figure 3.10. In this case, we can think of a possible downside to this being if an attacker has access to the network traffic from this device and realizes its server preference, it will make recognizing a possible NTP server for it easier and thus spoofing a response packet for it with the wrong time more doable. The number of servers from the pool to pick goes down from over 4000 servers to 1053 server (all 4 domains of North America region). A possible solution for picking a domain in this case is to have all domains available for more randomness and to not hard code any specific servers.

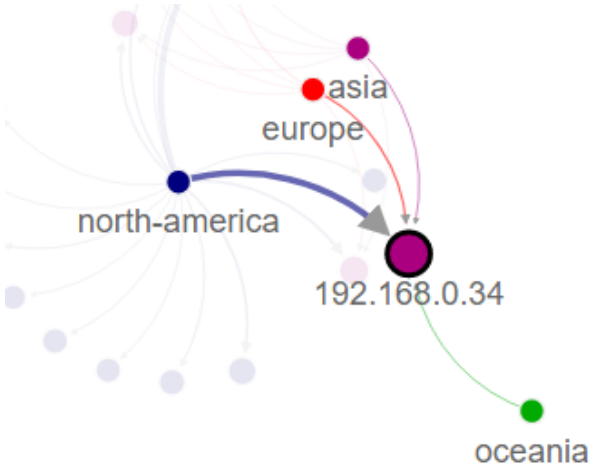


Figure 3.9 Example of a device with more than one domain/region (MiCasaVerde VeraLite)

The information about servers varies from one device to the other in a way that makes it difficult to apply the same analysis used for port randomization in to assess vulnerabilities in this particular aspect. It is mainly due to the constantly rising number of existing public NTP servers as well as the variation of stratum possible. According to Malhotra and al. [33] there were 11 728 656 IPs that potentially run NTP servers back in 2015.

Although more public servers could mean a better chance of randomization it could also mean a higher chance of requesting time from a bad timekeeper for the sole reason that it was randomly picked from the pool of servers defined in the NTP configuration on the device.

Another vulnerability caused by the choice of servers is hard-coding one specific server in the device instead of a pool or using an implementation of NTP that configures the device to keep synchronizing to the first NTP server it receives from DNS.

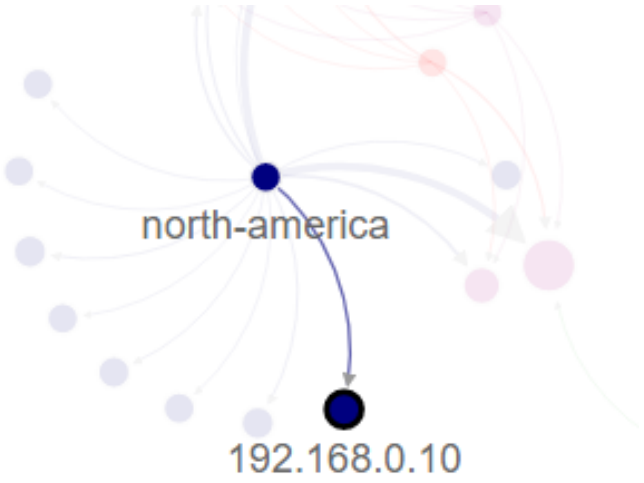


Figure 3.10 Example of a device with only one domain/region (Nest Camera)

3.4 Conclusion

In this chapter, we analyzed the empirical use of NTP in IoT devices by studying a number of its components and how IoT devices deal with them. We presented our dataset of 50 smart devices from various fields and with different vendors. We assessed the number of devices that use NTP version 4. We noticed how certain devices were using obsolete versions of the protocol which is considered a security vulnerability. We also emphasized on the fact that none of the NTP queries were authenticated or used available authentication methods in both NTP versions 3 and 4.

We also discussed how these devices use NTP packets to communicate with their NTP servers that are either hard-coded into them or for which they have pool names. In the second case, we discussed how DNS informs NTP clients which server IP addresses they can use for time synchronization. This depends on which servers are geographically closer to the device and how they free they are to be able to handle requests from it. We also presented a categorization of the use of NTP and time synchronization in general in IoT networks and how important it can be to their operations.

While tackling NTP packets fields from a security point of view, we discussed the way IoT devices choose their network source ports and how random they can be. This can make it harder for an attacker to tamper with time information destined for an IoT device. In this context, we have analyzed the patterns used by our dataset and concluded 4 main patterns of which we discussed impact on security and possible blind attacks.

These packets also showed how an RTC can help an IoT device do more checks on time information sent by the server it is synchronizing to. The check can be done through reference and transmit timestamps. the first one is used to compare time information and accept the closest ones and the second is used to authenticate the server's response. Some IoT devices do not dispose of an RTC and thus they can accept any response from an NTP server. This can lead to easier time shifting attacks.

Through this analysis, we conclude that for an IoT device to have a more secure NTP time synchronization they require an RTC or a reference clock on the route to the device to check the time correctness and server authenticity. It is also important to use a higher randomness pattern in choosing the network source ports these devices call for in their NTP requests.

CHAPTER 4 ATTACKING NTP IN IoT

This chapter investigates weaknesses in NTP protocol when used in IoT networks. Our findings showcase the importance of keeping correct time in IoT devices for tasks depending on it. To do so we created our own IoT test environments. We also implemented an attack tool that was used to demonstrate NTP vulnerabilities in regular networks in these environments. These vulnerabilities were discussed in different attack situations (on-path and off-path attackers). We discussed their feasibility outside the test environments and their repercussions as well. At the end of this chapter we suggest two architectures based on authenticating NTP for securing time synchronization in IoT networks.

4.1 Threat model

NTP vulnerabilities are prevalent in IoT, as shown in Chapter 3. The absence of port and server randomization or secure authenticated implementations of NTP opens vulnerable devices to attacks.

The main purpose behind this research work is to shed light upon the faulty/insecure behavior of these devices while using NTP to synchronize time. In order to investigate the impact and scope of these weaknesses, we will be applying attacks used for regular networks on sets simulating the general behavior of IoT networks. This process will be coupled with our devices' vulnerabilities such as previously discussed nonrandomized NTP servers or NTP UDP ports.

We define two types of possible scenarios: On-path and off-path attacks. Each scenario has its feasibility and requirements.

4.1.1 Attacker goal

The malicious entity in our threat model aims to shift time in a way that does not attract the attention of the IoT device user. If the user can see the time change on their device, they will actively change it or not take it into consideration in the tasks done by the device. In the IoT devices used on an everyday basis such as smart light bulbs, switches or locks, this is not an issue for the attacker because of the fact that many do not have a screen at all or their screens are only big enough to show the state at which they are (example: a smart light bulb with no display, display only big enough to show the temperature in a connected air conditioner and basic menu in a WiFi printer).

There are many existing possibilities for the attacker to shift time:

- The same day in the future or past
- Second, minutes, hours, days or years in the past or the future
- A specific day chosen by the attacker

In the threat model we are implementing, each one of these possibilities is used depending on the purpose of the malicious entity.

The purpose of picking a time shifting attack is to emphasize the possibility of causing greater harm than a DoS attack. This is achieved through controlling schedule times on a device that could be used for safety-critical tasks such as home security or health. Therefore, the attack aims to change the time on an IoT device with a secondary objective of making a change in the physical world. For example, affecting a patient's health by changing the frequency at which a connected infusion pump is used.

4.1.2 Attack assumptions

To accomplish this attack, the malicious user relies on two key assumptions that we have gathered through studying the 50-devices dataset:

- The devices do not use authenticated NTP synchronization methods. Indeed, none of the devices in our dataset were seen using these secure variants of NTP.
- The devices do not have an RTC. Many of the devices did not have a reference timestamp or a transmit timestamp which was previously explained as a direct consequence of not having an RTC as a local reference clock.

4.1.3 Attacker capabilities

Depending on the type of attack the malicious user is using, the have different capabilities:

- **On-path**: The attacker in an on-path attack is inside the local network. Thus, they have access to the network and are capable of sniffing, and selectively dropping, replaying or manipulating any packets sent between a legitimate server/client in the network. They are positioned between the victim and the gateway to apply a PiTM attack with these two as the targets through ARP poisoning.

Once the attacker obtains the NTP request sent by the victim IoT to the NTP server outside the network, the attacker creates a fake response to the request using an NTP attack tool called Delorean created by Selvi [35] (we explain more how Delorean works in section 4.3.1) with the server's IP address and the victim's IP address address as well as the port number used in the request.

The origin timestamp is not important since most IoT NTP request timestamps are NULL. This reduces the security of the device because the origin time-stamp (NTP nonce) check is not done.

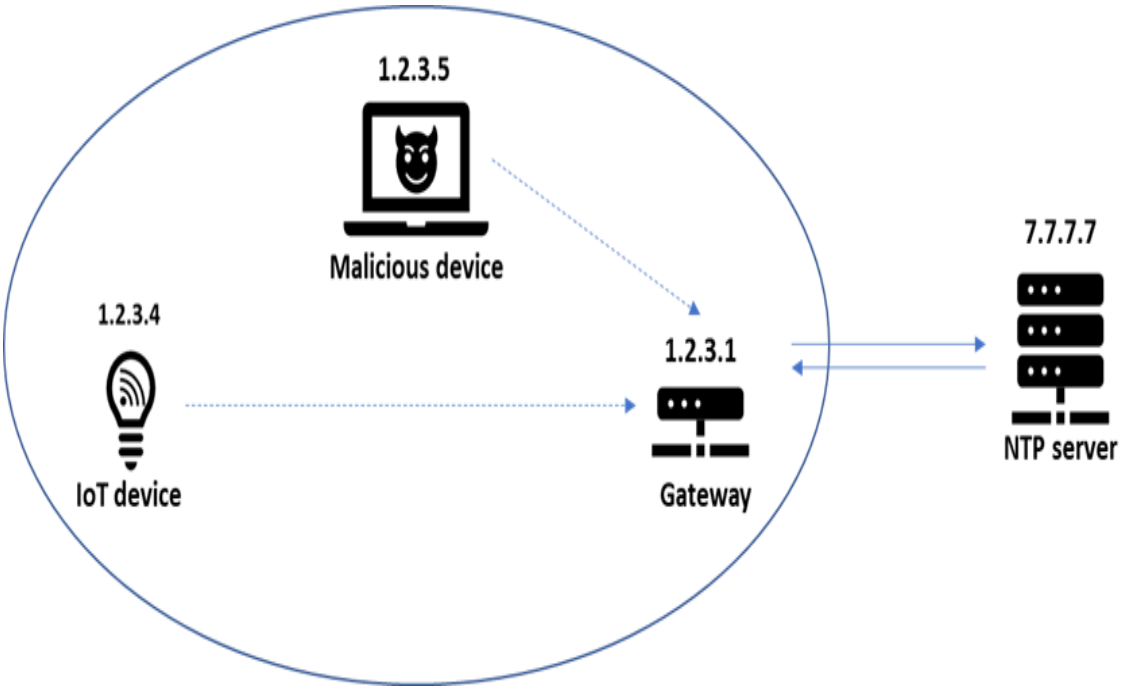


Figure 4.1 On-path threat model

- **Off-path**: The attacker in an off-path attack is outside the local network. Because they don't have access to the network, they cannot see the NTP traffic between the IoT device and the NTP server. This leads to the attacker having to go through a phase of reconnaissance to acquire the 5-tuple information needed for the NTP time shifting attack to succeed. There are a few ways they can do that such as Border Gateway Protocol (BGP)-hijacking or IP spoofing from external machines.

  Following the reception of the information needed, the attacker can fake a big number of NTP responses with the time shift needed and keep sending them as many times as possible until one of them is accepted by the victim.

The nonce check is not important here as well since the request timestamps are all NULL. For this attack to succeed quickly, the device needs to be using the same NTP servers for all requests or a specific pattern for servers' choice which is the case for a number of the devices we have studied in our dataset.

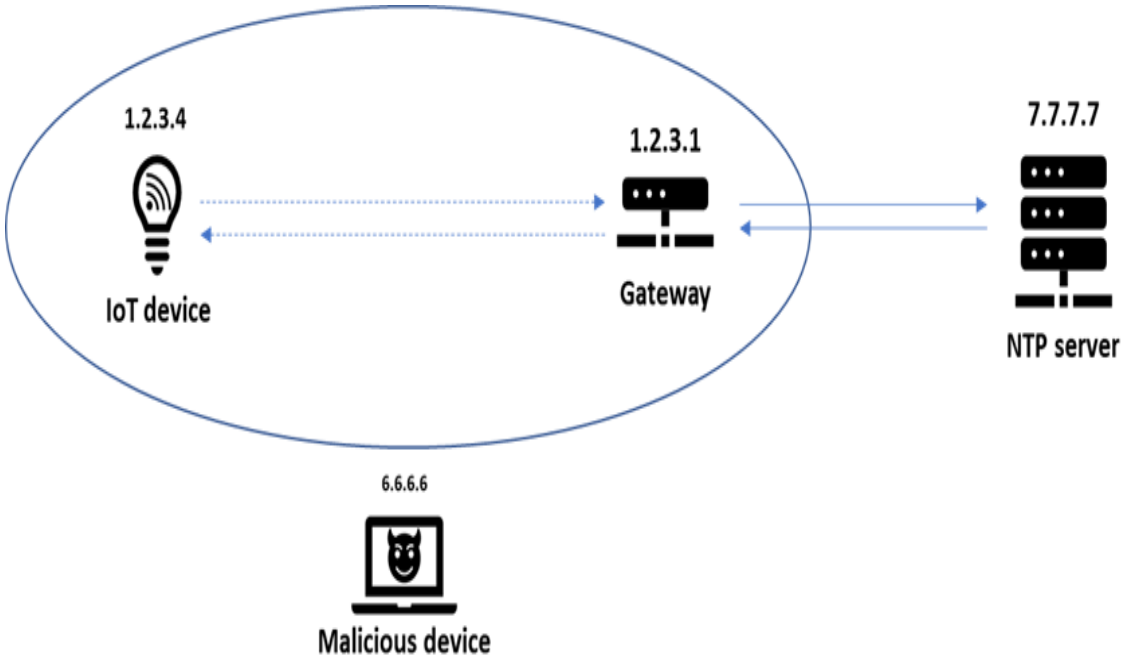


Figure 4.2 Off-path threat model

In this thesis, we implemented the on-path attack as it is the closest to what can be easily done by an attacker targeting connected objects as well as for how accessible IoT networks are. The purpose of this attack is to apply it in different situations in order to assess the behavior of a typical IoT device that does not use port randomization and uses server randomization.

4.2 Testbed presentation and normal NTP usage

To shed light on different components such as: NTP traffic, time changes and multiple vendors, we have put together 3 different test environments in our testbed that we identify as sets. On each set we will be applying NTP attacks in order to demonstrate IoT NTP communication vulnerability. These sets are:

- **Set 1**: A network of 3 virtual machines. This set is for a regular network reference in using the NTP protocol.

- **Set 2**: A network of 1 Arduino based IoT clock, a gateway and a Kali machine to apply the attack. In this set, we have an IoT environment but a non-commercial device that only works using NTP to show time and no other purpose. This environment is used to push for tests on an environment where NTP protocol is as isolated as possible from other network stack protocols.
- **Set 3**: A network of 1 IoT device (connected light bulb), a gateway, an android phone to control the IoT and a Kali machine to apply the attack. This set represents everyday life IoT use and the different limitations we have when trying to study or secure NTP protocol use in commercial smart devices.

4.2.1 Set 1 presentation and use of NTP

Set 1 presentation

In this set we used VMware Workstation 15.5.2, a hypervisor that enables users to set up virtual machines, on a Kali Linux 2020.1 release with a Linux 5.4.0-Kali4-amd64 kernel. Kali is a Linux distribution based on Debian and it is used for the purpose of advance penetration testing and auditing networks and machines. It comes with installed tools and software. We will discuss how we used some of them later in this chapter.

We set up 3 virtual machines on VMware, that we show on figure 4.3, for different purposes:

- **Ubuntu virtual machine**: 18.04 release and a Linux 5.3.0-51-generic kernel. This machine is used as the victim machine. It uses NTP to request time synchronization and is configured to only ask for a specific server (Kali server virtual machine). This machine uses the IP address 192.168.170.137.
- **Kali server virtual machine**: 2020.2 release with a Linux 5.4.0-Kali4-amd64 kernel. This machine is used as the NTP server we synchronize the Ubuntu machine's time to. Kali machines already have NTP installed on them. This server uses the IP address 192.168.170.128.
- **Kali attacker virtual machine**: 2020.2 release with a Linux 5.4.0-Kali4-amd64 kernel. This is considered the malicious entity in our setup and we are using different tools installed on it to position ourselves, as the attacker, in the traffic between the two previous machines. Our Kali attacker machine uses the IP address 192.168.170.129.

The different arrows on figure 4.3 are there to allow us to see the different types of traffic we will be looking into in this set (gateway, NTP and attack traffic). It also demonstrates

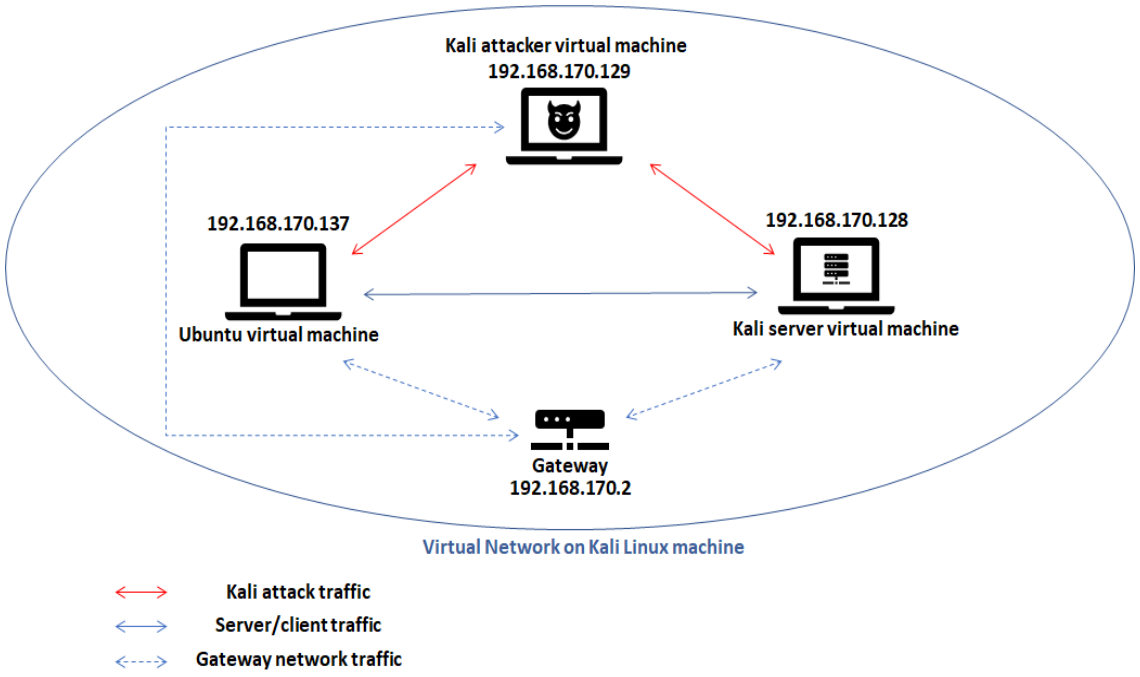


Figure 4.3 Set 1 presentation

how all 3 machines are on the same network because we will be implementing an on-path time shifting attack on this set. They all use the same gateway 192.168.170.2 provided by the hypervisor sub-network.

How NTP protocol is used in set 1

To set up this environment we have to first turn the Kali virtual machine into an NTP server. As NTP is part of the Kali Linux suite, we don't need to install it. We make sure we are synchronizing to lower stratum levels to ensure that our target can synchronize to this machine later on. For this set we chose to synchronize to the closest *pool.ntp.org* servers by using the following pools in the *ntp.conf* file:

```
server 0.ca.pool.ntp.org
server 1.ca.pool.ntp.org
server 2.ca.pool.ntp.org
server 3.ca.pool.ntp.org
```

We then restart the NTP service in the server machine using the shell command:

```
$ sudo service ntp status
```

To control the server to which our Ubuntu machine synchronizes its time, we configure the Ubuntu machine NTP implementation to do so. We first specify the IP and hostname of our NTP kali server machine in the hostname file using the shell command:

```
$ sudo nano /etc/hosts
```

After which we add a line naming our server in the hosts file as follows:

```
192.168.170.128  kali
```

We also add the server to the ntp.conf file the same way we added the pool to ntp.conf file in the Kali virtual server.

We then force the Ubuntu machine to synchronize to the NTP kali server with the command:

```
$ sudo ntpdate -u kali
```

To this the Ubuntu machines answers with the offset between its time and the time it is receiving from our Kali server.

Finally, we restart the NTP service on the Ubuntu machine and check again if it is receiving time from the Kali server. The command used here is:

```
$ ntpq -p
```

The response to this command is shown in figure 4.4. We can also see on it the different delays we can have between the server and the client as well as the ID of the Kali server our Ubuntu machine connects to and its stratum among other information.

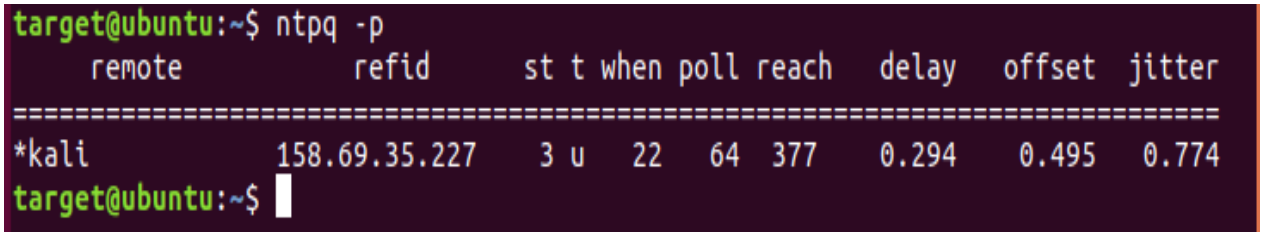


Figure 4.4 Synchronization result on Ubuntu machine in set 1

For the Kali attacker virtual machine, we set up NTP on it the same way we set it up on our server as it is going to try and act as a server during the attach.

While the Ubuntu machine is synchronizing its time to the Kali server, we start Wireshark to see how the traffic and packets are exchanged. We notice how actively the Ubuntu machine

is seeking time from the server. it is at a rate of at least 2 requests per minute but it can go up to 5 requests. These requests use different ports depending on how time is being required:

- UDP source port number 123 is used for all spontaneous requests.
- Random UDP source port number is used for every request we actively make from the command line window using the command ntpdate -u kali.

We also see that every NTP packer (request and response) has all timestamps populated as shown in the figures 4.5 and 4.6.

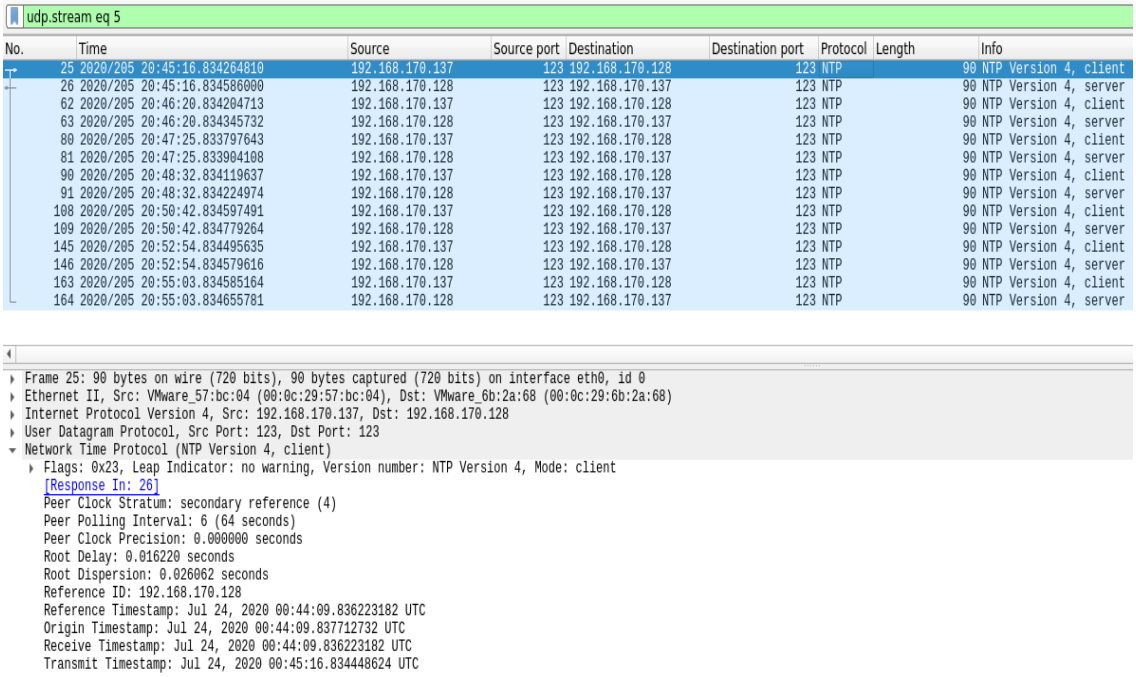


Figure 4.5 NTP spontaneous request from Ubuntu machine to Kali server machine

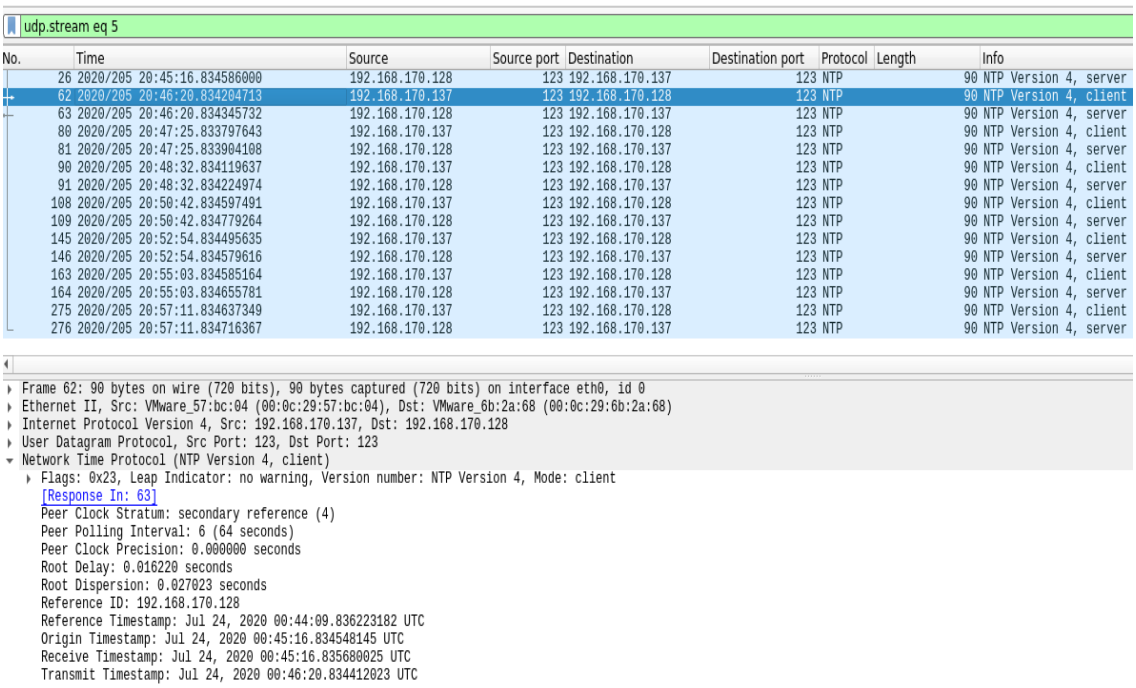


Figure 4.6 NTP spontaneous response from Kali server machine to Ubuntu machine

4.2.2 Set 2 presentation and use of NTP

Set 2 presentation

In this second set we put in place a test environment that is based on an IoT device and a network similar to one that is deployed by users. This network is based on 3 different components that we present in figure 4.8:

- **Arduino IoT LCD clock**: An IoT device that we programmed using Arduino 1.8.12 IDE. It is a clock with a screen that tells the current date and time through an implementation of NTP protocol as shown in figure 4.7. We used a Longruner 20x4 LCD display module IIC/I2C/TWI serial 2004 with screen panel expansion board white on blue. We connected this display to an ESP8266 NodeMCU WiFi shield. We configured the shield to request time from *2.ca.pool.ntp.org* and display specific information (time of the day and date). The code is shown in APPENDIX A.

- **Network gateway**: To be more in control of the network, we use a Lenovo laptop running Windows 10 as a hotspot for the wireless network our devices are connected to.

- **Kali attacker machine**: We use the same Kali Linux machine that was used to host

Figure 4.7 Arduino IoT LCD clock

the virtual machines from set 1 with the spoofing and PiTM tools for an NTP attack that we explain during the attack's presentation.

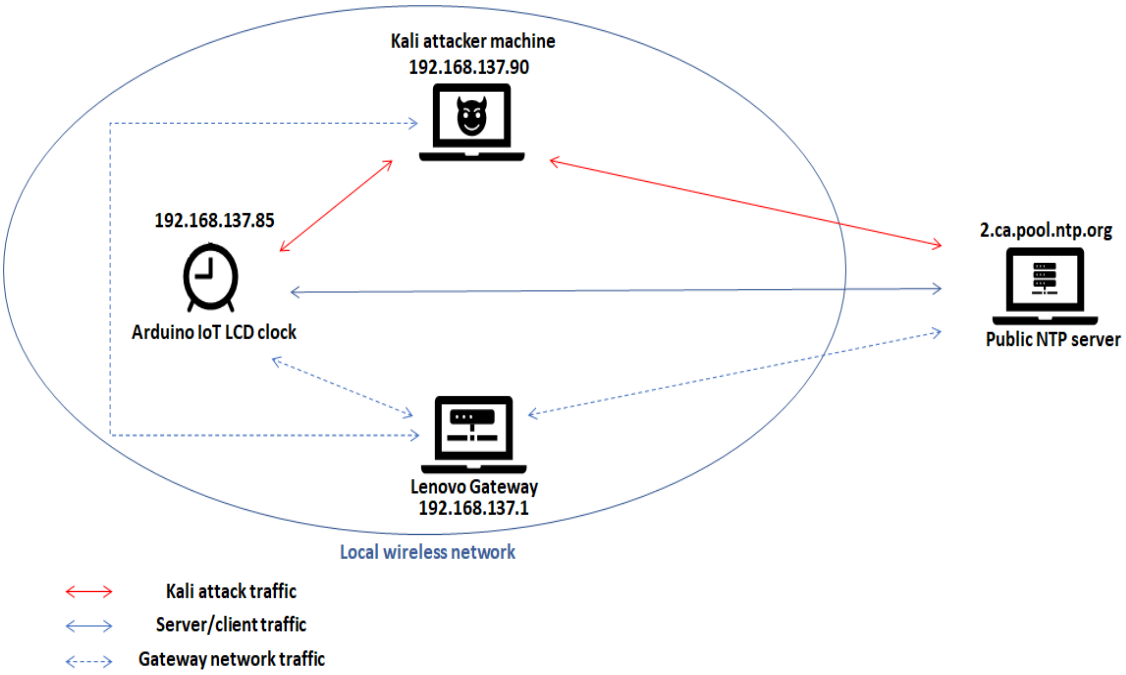


Figure 4.8 Set 2 presentation

How NTP protocol is used in set 2

To build this environment, we turn on the hotspot on our Lenovo gateway, connect the Arduino device to it. This device is set to automatically connect to the NTP server IPs it receives through DNS queries shown in figures 4.9 and 4.10.

We monitor the NTP communication of this device with NTP servers it received from the DNS server. The pattern used is one request for time synchronization every minute with a different server IP from the list received. The device also uses the same UDP source port number 1337 for all NTP requests.

We also recognize the non-existence of an RTC, because we did not use one, in the form of the NULL value of all timestamps of the NTP request sent by the device as shown in figure 4.11. It is also present in the origin timestamp in the NTP response shown in figure 4.11.

The use of this device is important because we can access its code and add configurations accordingly, providing more flexibility to test security measures or attacks.

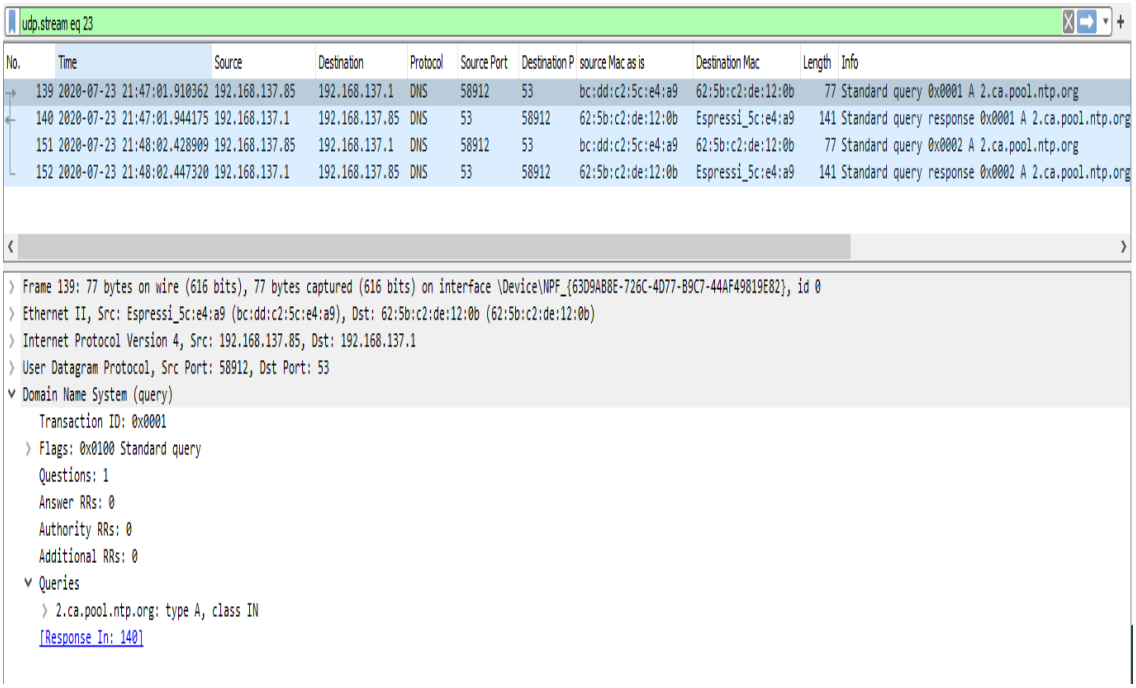


Figure 4.9 DNS request of Arduino device to receive NTP server IP address

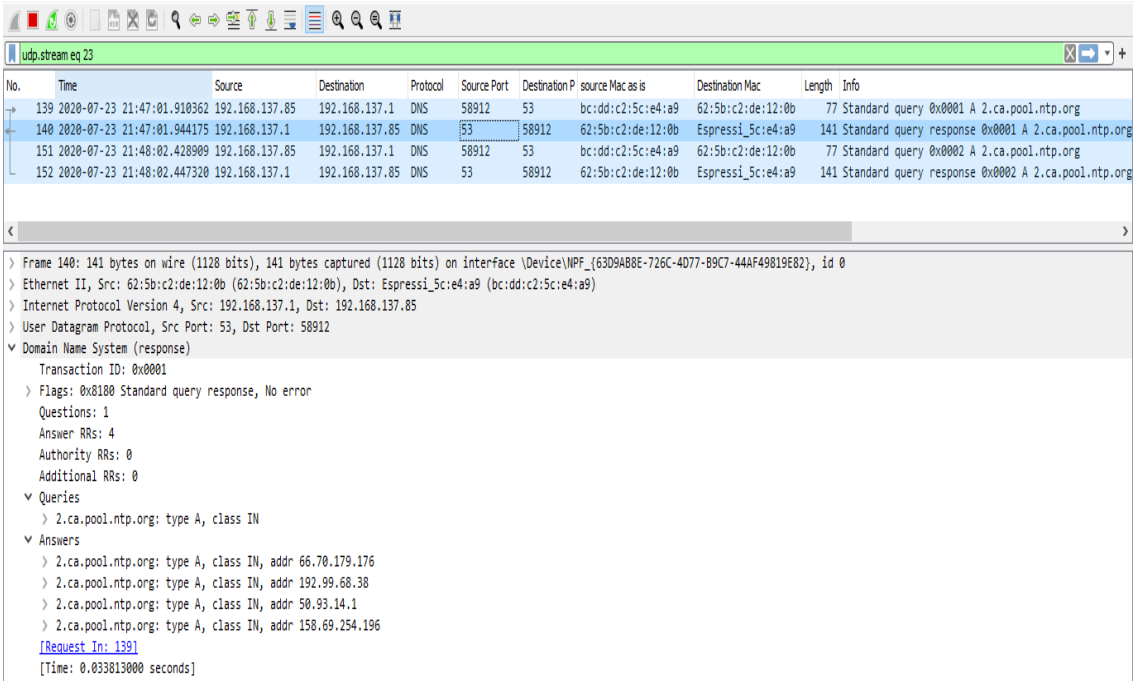


Figure 4.10 DNS response to Arduino device request with NTP server IP addresses from *2.ca.pool.ntp.org*

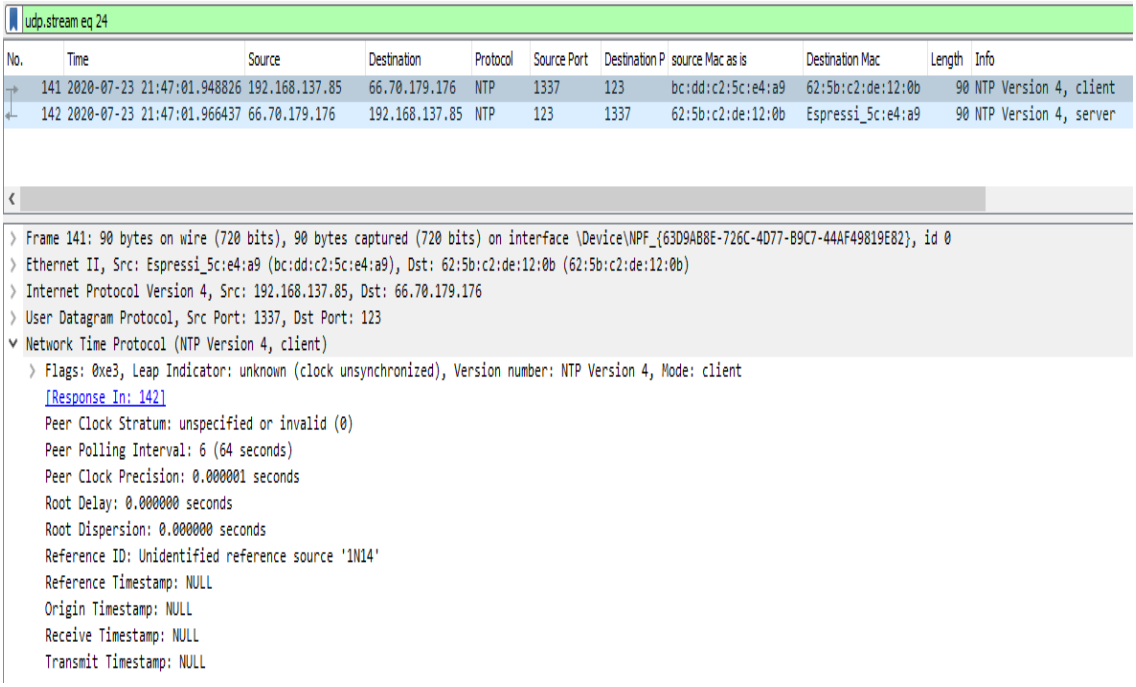


Figure 4.11 NTP request of Arduino device with NULL timestamps

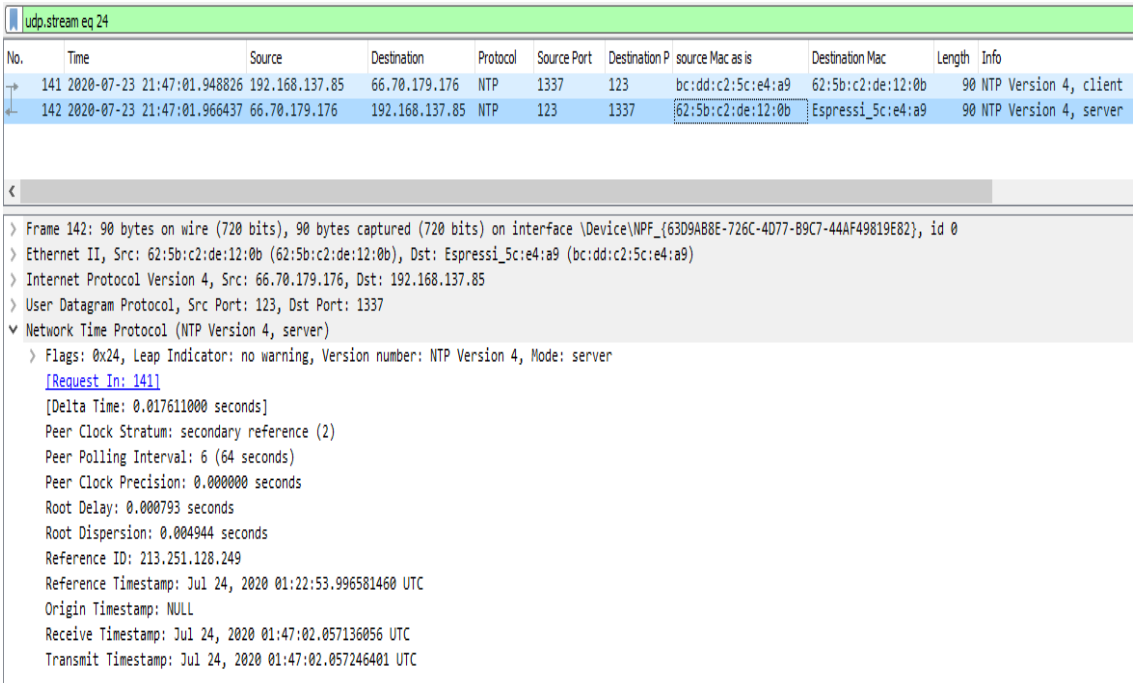


Figure 4.12 NTP response for Arduino device request with NULL origin timestamp

4.2.3 Set 3 presentation and use of NTP

Set 3 presentation

This set is a replication of the second set but with the replacement of the configured Arduino device with a smart light bulb. it is a TP-Link WiFi light bulb series A-19 LB120. it takes the local area network IP address 192.168.137.249. We use the same architecture as the one used in set 2 and we show in figure 4.13.

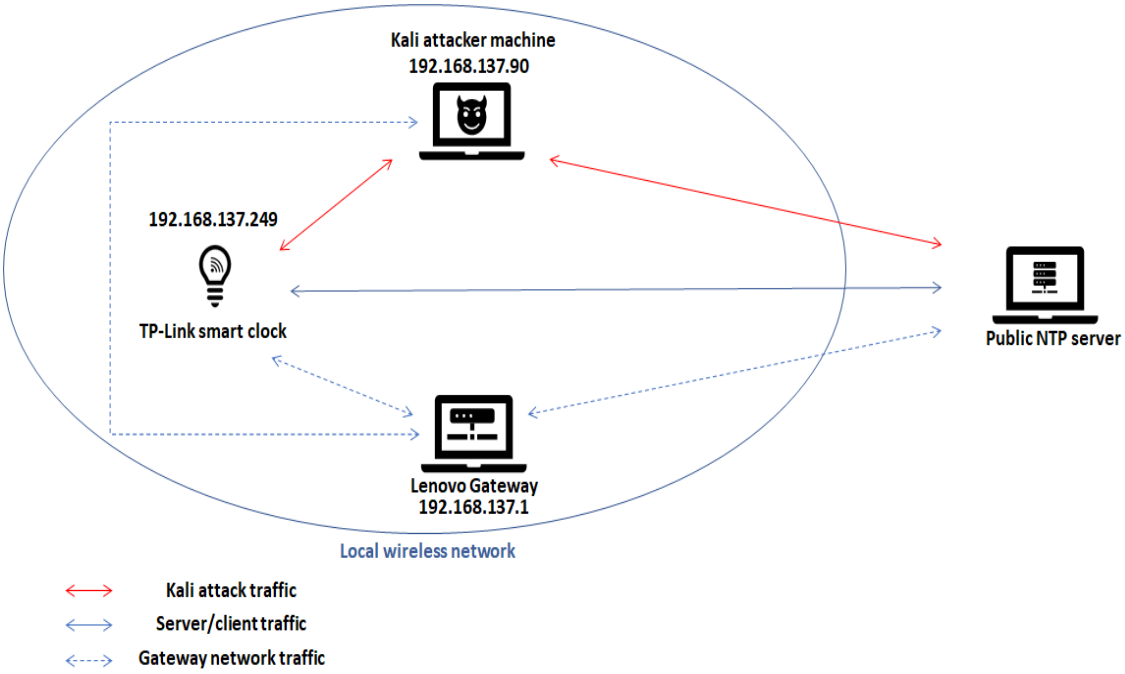


Figure 4.13 Set 3 presentation

How NTP protocol is used in set 3

Like our Arduino device and many IoT devices, the connected light bulb, once connected to the hotspot wireless network through our Lenovo gateway, shows defined patterns in using the NTP protocol. It uses the UDP source port 1490 for time synchronization requests which were being sent every 30 minutes to servers belonging to different pools. The NTP public pools we have recognized from DNS queries are *pool.ntp.org* and *time.nist.gov*.

Another information we have from monitoring NTP traffic by this device is that it doesn't have an RTC as shown on figure 4.14 with NULL timestamps.

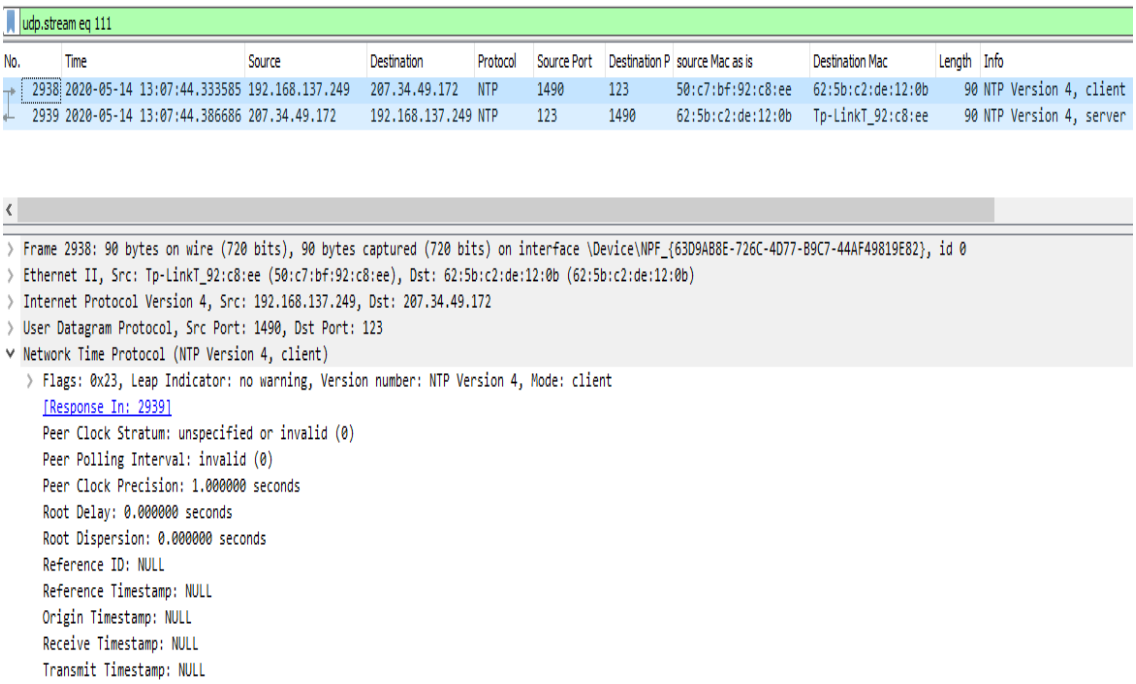


Figure 4.14 NTP request of smart TP-Link light bulb in set 3 with NULL timestamps

4.3 On-path time shifting

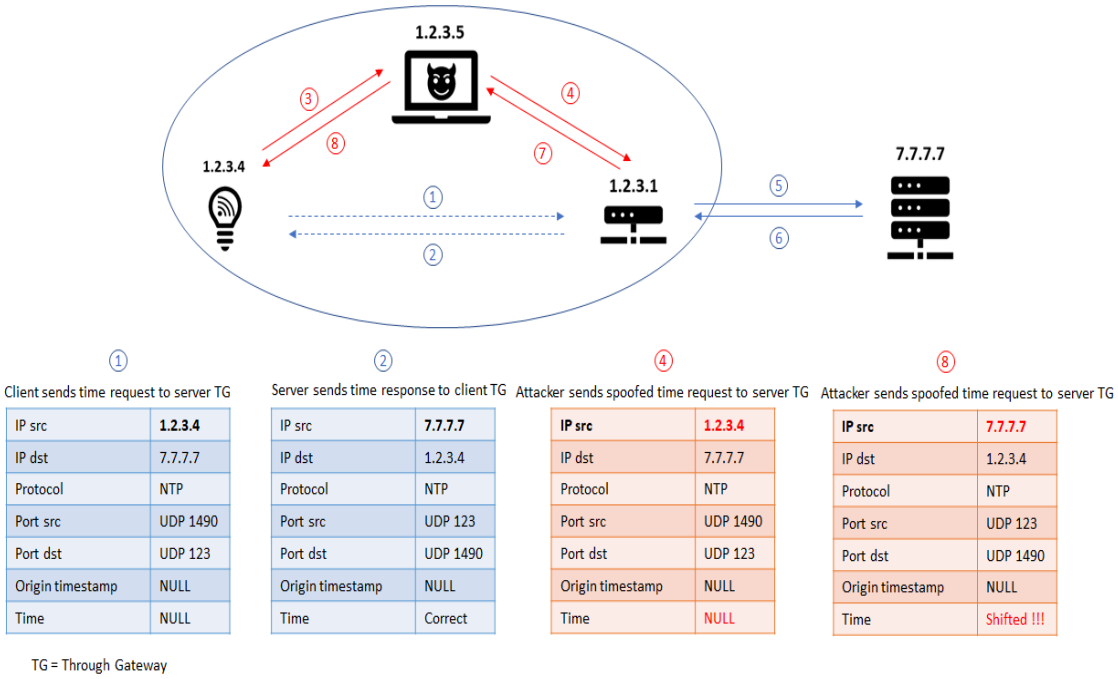
4.3.1 Attack architecture

The architecture of the attack we are using to evaluate how vulnerable the implementation of the NTP protocol is in IoT devices consists of a man in the middle attack. We are using ARP spoofing coupled with the tool Delorean [35].

Delorean is an NTP hacking tool we are running on the Kali attacking machine, both virtual and hardware. It is based on the fact that most NTP communications do not use use the authenticated version. Thus, if a malicious user connected to the same local area network has access to Delorean and can see the NTP traffic, they can also use this tool coupled with a PiTM tool to spoof NTP packets and send them to the IoT device. These packets will have the shifted time either to the future or to the past to accomplish their goal.

In most IoT devices it is rare to have a screen showing the time and date. This is why in set 3 we chose a connected light bulb with scheduling properties and our set 2 was connected to an LCD screen that shows the time.

In applying the attack, whether it is a virtual or hardware Kali attacking machine, the process is the same. Once we infiltrate the victim network, the Kali attacker machine is configured



IP src	1.2.3.4
IP dst	7.7.7.7
Protocol	NTP
Port src	UDP 1490
Port dst	UDP 123
Origin timestamp	NULL
Time	NULL

IP src	7.7.7.7
IP dst	1.2.3.4
Protocol	NTP
Port src	UDP 123
Port dst	UDP 1490
Origin timestamp	NULL
Time	Correct

IP src	1.2.3.4
IP dst	7.7.7.7
Protocol	NTP
Port src	UDP 1490
Port dst	UDP 123
Origin timestamp	NULL
Time	NULL

IP src	7.7.7.7
IP dst	1.2.3.4
Protocol	NTP
Port src	UDP 123
Port dst	UDP 1490
Origin timestamp	NULL
Time	Shifted !!!

Figure 4.15 On-path attack architecture

to PiTM all traffic between the victim and the NTP server (or gateway). We additionally configure the Kali attacker machine to intercept outbound NTP queries and redirect them to the Delorean tool listening on UDP port 123. This is done as follows:

```
1 #echo 1 > /proc/sys/net/ipv4/ip_forward
2 #PREROUTING -p udp -dport 123 -j REDIRECT -to-port 123
```

Line 1 in the code enables IP forwarding. Line 2 redirects NTP queries to the Delorean tool.

For our attack we chose to use Ettercap to apply a MITM ARP spoofing attack. The command used for Delorean time shifting attack 5 minutes to the future is :

```
1 #/usr/bin/python delorean.py -s 5m
```

4.3.2 Shifting time effect on scheduling

We executed the time shifting attack on the 3 sets in two ways. We configure the Delorean tool through the command shell to respond to victim NTP queries with packets containing time shifts into the future or the past. We applied these versions of the attack and noted the results on the different sets as follows:

- When applying the attack on set 1, although Ubuntu sends time synchronization requests every minute, the time sent by Delorean is taken into consideration every time a network interface goes up. After that it takes the Ubuntu victim machine in our environment an average of 2 synchronization cycles with the attack applied for it to accept the shifted time as it is comparing it to the reference clock and sending more requests to apply NTP checks.

  Although this machine is TLS capable, it does not make use of it or any encryption in the NTP traffic. This is explained by the implementation of unauthenticated NTP protocol. The victim computer reacts in the same manner to both future and past time shifts. Time changes can be seen on the machine's clock as well as on Delorean's command line window in the Kali attacking virtual machine. Figure 4.16 shows the Ettercap and Delorean tools together while applying a time shifting attack on the Ubuntu virtual machine from Kali attacker virtual machine to send it 10 minutes into the future.

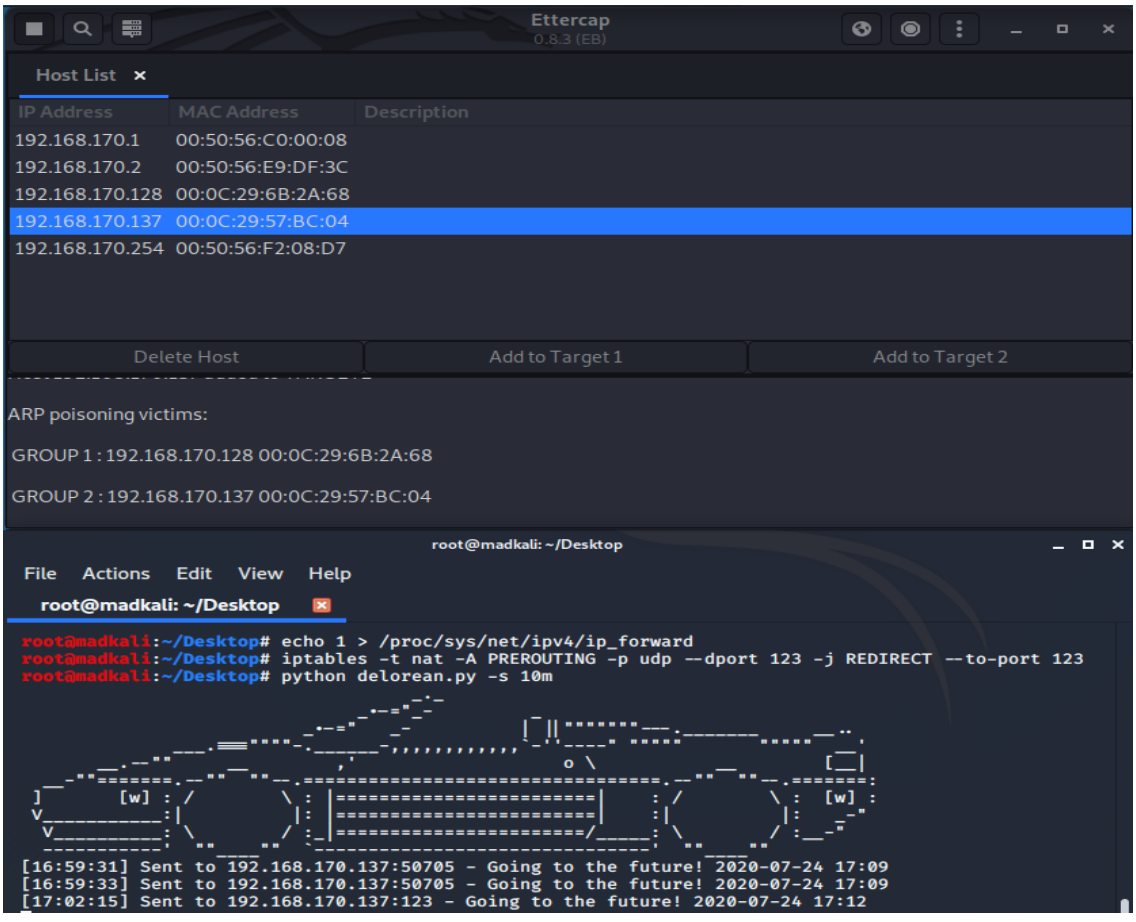


Figure 4.16 Attack on NTP on the Kali machine: Ettercap and Delorean

- We configured set 2 in a way that makes the Arduino IoT device request time every minute using NTP version 4. This device is a simple clock and does not use TLS. Therefore, it does not use authentication for its NTP queries either. This makes the device vulnerable to PiTM attacks.

  In addition to that, The Arduino IoT clock is modeled on most IoTs in our dataset and does not have an RTC. Because of the absence of checks due to the NULL value caused by the absence of RTC, once we apply our on-path time shifting attack on this set, the first synchronization cycle sends the device into the shifted time without any checks. The shifted time can be seen on the screen of the Arduino device, the Delorean attack screen and the packets captured using Wireshark as for set 1.

- We test the same attack on our final set (set 3) to showcase the effects it has on real manufactured connected devices. Because our device is a connected light bulb that does not have a screen to show time, we are testing the effects of the attack on the scheduling service provided by the app monitoring the light bulb. The IoT device reacts to the attack in the same manner our Arduino device does because they both don't have an RTC to check the shifted time against.

  Once we set the attack to send the time 5 minutes into to the future on the light bulb, we wait for the first synchronization cycle. It sets the time on the light bulb 5 minutes ahead. We set a schedule for the light bulb to turn on at 15:55 and it lights up at 15:50. That demonstrated that the time on it adopted the attack's time as the correct time. Once we disconnect the light bulb from the internet with an already set schedule with the shifted time and while keeping it with connected to a power source, it continues working according to the shifted time.

4.3.3 Shifting time effect on TLS certificates

In Set 3, the connected TP-link light bulb uses TLS handshake to make regular contact with the app as well as for every "lights on" and "lights off" command. Figure 4.17 shows the different packets exchanged between the light bulb and its servers during a "lights off" command. In these packets there is a TLSv1.2 handshake that is made. During the handshake, the client and server exchange keys and the server also sends 4 certificates to the client. They range in time from the year 2004 to the year 2034 and are shown in figure 4.18.

These same certificates are exchanged during every handshake. That is why we thought of testing how far the use of the light bulb can be affected by tampering with certificates validity start and end dates.Thus, in this part of the on-path time shifting attack, we aim to disturb

tcp.stream eq 55

	Time	Source	Destination	Protocol	Sourc	Destir	source Mac as is	Destination Mac	Length	Info
1986	2020-07-24 13:54:55.461825	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	58	63220 → 443 [SYN] Seq=0 Win=1452 Len=0 MSS=1452
1987	2020-07-24 13:54:55.495136	52.45.56.20	192.168.137.187	TCP			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	58	443 → 63220 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460
1988	2020-07-24 13:54:55.497927	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	63220 → 443 [ACK] Seq=1 Ack=1 Win=4356 Len=0
1989	2020-07-24 13:54:55.618882	192.168.137.187	52.45.56.20	TLSv1.2			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	196	Client Hello
1990	2020-07-24 13:54:55.651620	52.45.56.20	192.168.137.187	TCP			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	54	443 → 63220 [ACK] Seq=1 Ack=143 Win=27872 Len=0
1991	2020-07-24 13:54:55.654792	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	[TCP Dup ACK 1988#1] 63220 → 443 [ACK] Seq=143 Ack=1 Win=4356 Le
1992	2020-07-24 13:54:55.660648	52.45.56.20	192.168.137.187	TLSv1.2			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	1506	Server Hello
1993	2020-07-24 13:54:55.660736	52.45.56.20	192.168.137.187	TCP			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	1506	443 → 63220 [PSH, ACK] Seq=1453 Ack=143 Win=27872 Len=1452 [TCP s
1994	2020-07-24 13:54:55.660787	52.45.56.20	192.168.137.187	TCP			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	1506	[TCP Window Full] 443 → 63220 [ACK] Seq=2905 Ack=143 Win=27872 Le
1995	2020-07-24 13:54:55.665852	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	63220 → 443 [ACK] Seq=143 Ack=1453 Win=3630 Len=0
1996	2020-07-24 13:54:55.666502	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	63220 → 443 [ACK] Seq=143 Ack=2905 Win=3630 Len=0
1997	2020-07-24 13:54:55.666504	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	63220 → 443 [ACK] Seq=143 Ack=4357 Win=3630 Len=0
1998	2020-07-24 13:54:55.702701	52.45.56.20	192.168.137.187	TLSv1.2			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	1294	Certificate, Server Key Exchange, Server Hello Done
1999	2020-07-24 13:54:55.949164	52.45.56.20	192.168.137.187	TCP			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	1294	[TCP Retransmission] 443 → 63220 [PSH, ACK] Seq=4357 Ack=143 Win=
2000	2020-07-24 13:54:56.060861	192.168.137.187	52.45.56.20	TLSv1.2			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Mess
2001	2020-07-24 13:54:56.061370	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	[TCP Dup ACK 2000#1] 63220 → 443 [ACK] Seq=269 Ack=5597 Win=3630
2002	2020-07-24 13:54:56.061372	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	[TCP Window Update] 63220 → 443 [ACK] Seq=269 Ack=5597 Win=4356 L
2003	2020-07-24 13:54:56.102623	52.45.56.20	192.168.137.187	TLSv1.2			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	105	Change Cipher Spec, Encrypted Handshake Message
2004	2020-07-24 13:54:56.105419	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	63220 → 443 [ACK] Seq=269 Ack=5648 Win=3630 Len=0
2005	2020-07-24 13:54:56.210925	192.168.137.187	52.45.56.20	TLSv1.2			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	639	Application Data
2006	2020-07-24 13:54:56.255572	52.45.56.20	192.168.137.187	TLSv1.2			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	1271	Application Data
2007	2020-07-24 13:54:56.264946	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	63220 → 443 [ACK] Seq=854 Ack=6865 Win=3630 Len=0
2008	2020-07-24 13:54:56.266631	52.45.56.20	192.168.137.187	TLSv1.2			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	88	Application Data
2009	2020-07-24 13:54:56.272258	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	63220 → 443 [ACK] Seq=854 Ack=6899 Win=3630 Len=0
2010	2020-07-24 13:54:56.315636	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	63220 → 443 [FIN, ACK] Seq=854 Ack=6899 Win=3630 Len=0
2011	2020-07-24 13:54:56.367357	52.45.56.20	192.168.137.187	TCP			62:5b:c2:de:12:0b	Tp-LinkT_92:c8:ee	54	443 → 63220 [FIN, ACK] Seq=6899 Ack=855 Win=28665 Len=0
2012	2020-07-24 13:54:56.371190	192.168.137.187	52.45.56.20	TCP			50:c7:bf:92:c8:ee	62:5b:c2:de:12:0b	54	63220 → 443 [ACK] Seq=855 Ack=6900 Win=3629 Len=0

Figure 4.17 Packets exchanged between Light bulb and TLS and TCP servers during the "lights off" command sent from app on phone

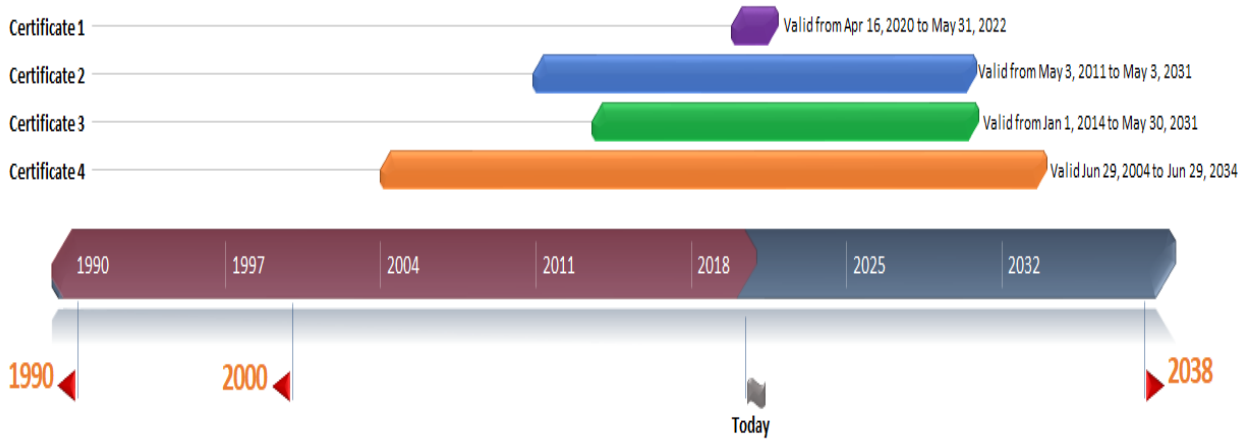


Figure 4.18 Validity duration of the 4 TLS certificates on TP-Link light bulb

the use of TLS by sending the device beyond the validity time of its TLS certificates. We chose the years 2000 and 2038 for the attack as they are 4 years before or after the widest validity range of the certificates that we mentioned above. Both attacks went through and we had the following results:

- **Time shifting to 2000**: both scheduling and commands services are functioning the same as with normal correct time. The TLS exchange we notice during the handshake continues with the same certificates although they not valid yet. On the side of the attacking machine, we can see the time on the response is the year 2000 as well as intercepted NTP packets.
- **Time shifting to 2038**: Similar to the previous time shifting to the year 2000 attack, both scheduling and commands services still function as they should with the correct time and the TLS certificates exchanged are the no longer valid. The difference between the two attacks is there are no intercepted NTP packets on the side of the attacker machine. There are also no responses for the 3-time synchronization requests sent by the light bulb for every synchronization cycle.

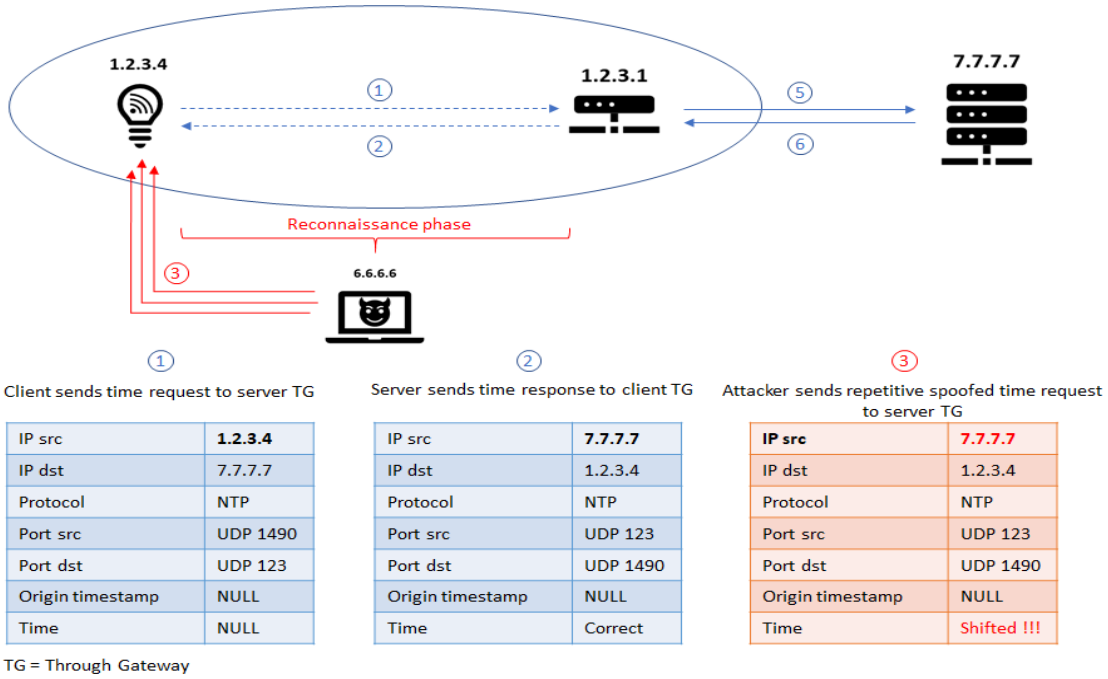
As a result of these attacks, it appears that the device does not check the validity of the certificates. Although tampering with NTP will not render the device useless, it is a security issue when it comes to TLS itself. A few questions about the efficiency of the TLS implementation used by this IoT device and whether or not the certificates are doing their security task.

4.4 Off-path time shifting

4.4.1 Attack architecture

Although off-path attacks are harder to deploy because of the number of unknown variables in the 5-tuple there are, they are still part of the attack spectrum possible on IoT networks. This can be explained by the fact that IoT networks lack security measures and are usually apart from the traditional networks which are secured by firewalls.

The off-path attack we are discussing relies on an extensive sending of NTP response packets to a specific device until this device accepts said packets and synchronizes its time to the attack time.



IP src	1.2.3.4
IP dst	7.7.7.7
Protocol	NTP
Port src	UDP 1490
Port dst	UDP 123
Origin timestamp	NULL
Time	NULL

IP src	7.7.7.7
IP dst	1.2.3.4
Protocol	NTP
Port src	UDP 123
Port dst	UDP 1490
Origin timestamp	NULL
Time	Correct

IP src	7.7.7.7
IP dst	1.2.3.4
Protocol	NTP
Port src	UDP 123
Port dst	UDP 1490
Origin timestamp	NULL
Time	Shifted !!!

Figure 4.19 Off-path attack architecture

4.4.2 Requirements

As discussed in the threat model of the off-path attack, acquiring the 5-tuple is one of the main requirements for the success of this attack. The requirements for this attack are:

- **Target IoT device's internal IP address**: the attacker needs to go through a phase of reconnaissance to acquire it. This can be done through several methods such as BGP-hijacking.
- **Server IP address**: the attacker can use public servers' IP addresses. If they know the specific server then it is easier. If the server is unknown, the attacker can either choose one server and keep sending packets with its spoofed IP (if the IoT randomizes server choice, it is possible) or go through all the servers one after the other until the attack succeeds.
- **Target IoT device's NTP port**: if the attacker recognizes the IoT device in question, they can figure out the port it uses for NTP communication if it does not randomize port choice. If port choices are randomized, the current attack can be coupled with a fragmentation attack where the missing field is the IoT device's NTP port as explained by Malhotra and al. [33] (see APPENDIX C). NTP communications make it so that

fragmented packets are assembled when received and certain missing fields are added with the correct components.

- **Server NTP port**: the NTP port used by servers is the default NTP port number 123.

4.4.3 Feasibility

The feasibility of this attack depends mainly on the reconnaissance phase. if the attacker can't recognize a specific target then their choice will be blind attacking the IoT network.

Blind attacks require time and patience especially because of the no-RTC feature of IoT devices. If one packet succeeds in going through and changing the time, then the attacker needs to continue sending that same packet because the device will switch back to the true correct time as soon as it goes through another synchronization cycle. This means the attacker needs to have the means of sending continuous similar packets and knowing when the device accepts them.

Such off-path attacks are mostly used for DoS attacks because of the repetitive behavior and the lack of information about the destination. A similar concept has been used for recent NTP amplification attacks where the attacker used bots to simulate clients and spoof their IP addresses to render a server out of service.

Although very demanding in computing power, they are still feasible even if it means getting the device to change time for a short period of time.

If we take into consideration an IoT device that is only request time from the public *pool.ntp.org* with its current 4039 servers, the probability of the attacker guessing the correct server that we label S is $2.4x10^{-4}$.

Our light bulb IoT uses the port 1049 and synchronizes its clock every 30 minutes. The Arduino clock synchronizes its time every 1 minute and uses port 1337. If we consider an IoT device synchronizing its clock every 5 minutes using only one random port of the 64513 possible ports (from 1024 to 35535 adding the possibility of using default NTP UDP port number 123). This leads to a probability of guessing the right source port number P of $1.55x10^{-5}$.

The probability of then having the correct server and correct port number is $B = SxP = 3.72x10^{-9}$. This demonstrates how randomization of both values can be a highly appreciated feature of security.

To have an idea about how long this attack could take to succeed we proceed as follows: If

we consider the attacker capable of providing 100 spoofed packets every second, coupled with the fact that it takes 300 seconds for the IoT device to send a request, the attacker can send $3x10^4$ spoofed packets during that time therefore the possibility of having the right spoofed packet during the 300 seconds is $C = Bx3x10^4 = 1,116 * 10^{-4}$. For this to become true it will therefore take up to 32 days of continuous spoofing attack ($300/C$).

Although clearly time and effort consuming, these attacks are still feasible especially with the use of bots that can focus on different UDP source ports and servers and thus reduce the time needed.

4.5 Defenses

We have, so far through our literature review as well as the experiments done on different types of devices, demonstrated how the IoT field can be vulnerable to attacks and misuse of a protocol such as NTP. It is difficult to trust these devices to do their job as should be when they rely on insecure time synchronization.

It is also important to take into consideration the challenges we face in securing such a ubiquitous especially when developers don't appear to use best practices. That is why we discuss the possibility of two classes of solutions: backward compatible ones, and backward incompatible one.

4.5.1 Backward compatible NTP IoT security measures

Because of the great numbers of IoT companies and devices already in the wild, there is a big diversity in components and implementations as well. These devices use different OSs, microcontrollers, embedded systems and different versions of NTP protocol with different patterns of choice of UDP source ports as well as NTP servers.

One way of securing the NTP implementation in these devices would be to include the manufacturers in this process. They can recall and upgrade their products. But this is highly expensive and not always inclusive, especially for end-of-service devices that are still used by customers or companies that have gone out of business during that time. It makes this option unthinkable from an industry point of view alone.

A second way of dealing with the issue of securing NTP could be through reverse engineering the most popular devices and including secure implementations of NTP in them. While this is possible for the IoT developers' community, this option is not user friendly as it requires bringing sophisticated changes to the software through methods the average user of a smart

device does not need to have. It is also not inclusive towards the other existing devices that are being used by a smaller number of people.

Therefore, the logical way to go about protecting IoT devices from time-shifting attacks is by working on adding security features to their gateways. These gateways have an already existing more powerful resources that can accept advanced updating. Gateways are also a cost effective (1 gateway for all devices), they scale well (any new device only needs to connect to it) and do not require an advance user knowledge of the device. They have been used to solve security issues before in this field as well [49] [50].

We propose an architecture for securing NTP traffic in IoT devices that relies on the advances already made in traditional networks security without affecting the connected devices. This can be done by applying security measures on the router-side or adding a gateway box with adequate security option to do so.

The gateway box needs to be capable of monitoring and redirecting NTP traffic to either internal NTP servers or trusted public servers. This box should also be TLS capable to force authentication of time synchronization traffic and communication with servers that only accept authenticated NTP requests.

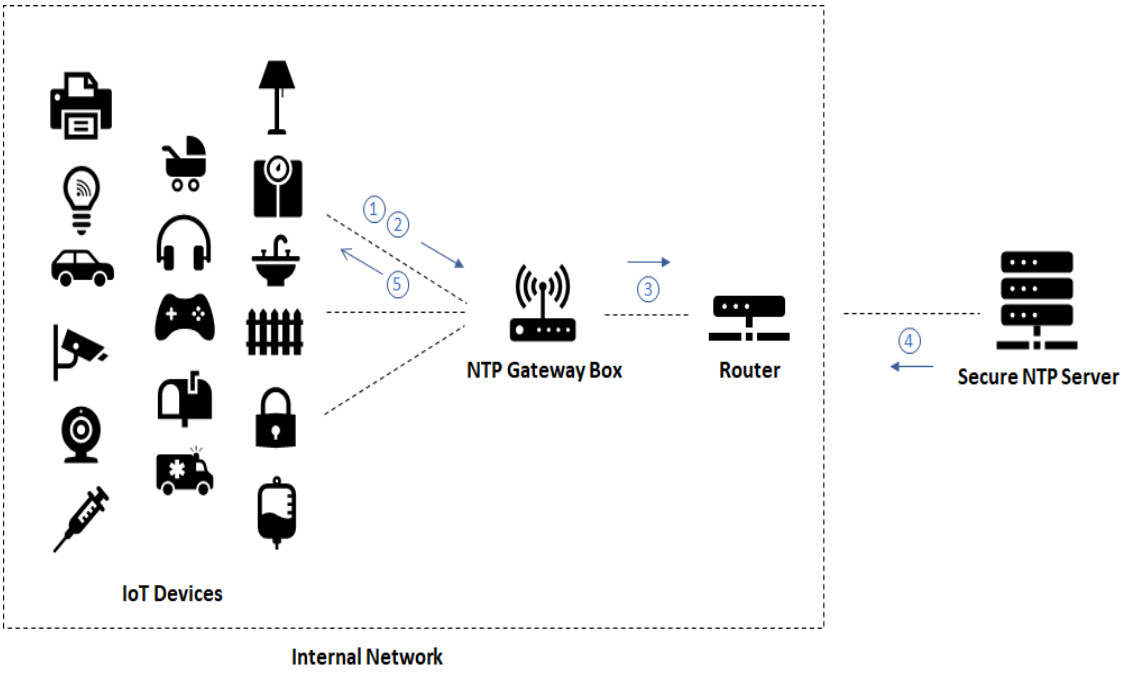


Figure 4.20 Proposed NTP security gateway for IoT devices that cannot be modified

The way this architecture works is by following these steps:

1. The IoT device is connected to the NTP gateway which is connected to the router.
2. The IoT device sends its NTP request to the NTP server through the NTP gateway box.
3. The NTP gateway box intercepts the request and redirects it to the correct secure NTP server with authentication information.
4. The server responds to the request.
5. The IoT device receives the NTP response.

to prevent ARP spoofing between the NTP gateway box and the IoT device, we include ARP spoofing detection and alert tools in the gateway. These attacks can be detected by Wireshark packet sniffing or by listing the ARP table (if one MAC address is shared by two IP addresses, it is more than likely ARP poisoning). An existing tool that does this is `arpwatch`[1]. After the attack packets are detected, they can be dropped using filters and firewalls.

This solution achieves the goals of securing NTP time synchronization in IoT networks and protecting these devices. It allows incremental deployment, and in general appears easy to use even for non-experts as it should only require as much interaction as connecting an IoT device to the home router.

4.5.2 Backward incompatible NTP IoT security measures

One secure NTP infrastructure for IoT devices is to include NTP security at device manufacturing time. This can be done by configuring IoTs to connect to secure NTP servers using secure methods. The latest of NTP security in traditional networks is the use of NTS described in chapter 2. We can reproduce NTS for IoT networks. To do so we can depend on the same public NTS server that exists or use the NTS implementation made public because IoT devices depend on the same pools used by traditional networks.

TLS is complex protocol, but it is making its way into even very embedded IoT devices. Our connected TP-Link light bulb is also using as described in chapter 4. There are also more and more libraries describing different implementations of this protocol in Arduino IoT devices.

NTS depends on the use of NTP and TLS. IoT devices can be TLS-capable which we notice in the connected light bulb's communication we have in our set 3. The light bulb uses TLS v1.3 as well as certificates to communicate with the server. The total size of the certificates

[1]https://linux.die.net/man/8/arpwatch

transferred from the server and used by the light bulb is 6053 bytes. This can be seen in the screen shot from the TLS handshake between the light bulb and the server in figure 4.21.

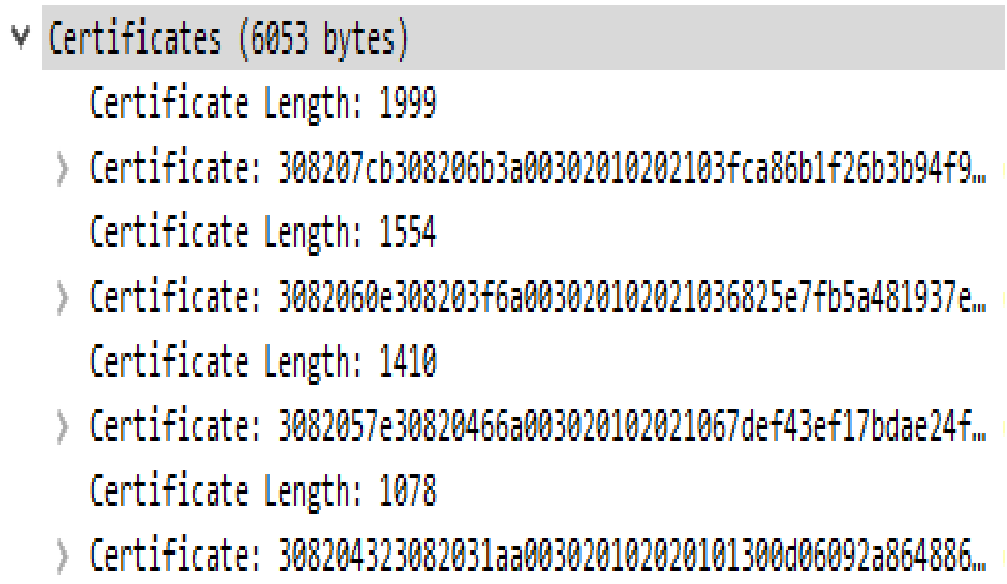


Figure 4.21 TLS certificates total size in a connected light bulb

Similar certificates can be used in our Arduino set up with the ESP8266 WiFi shield. We ran a code on the shield that shows the free memory size in it. the results of this code are shown in figure 4.22 The total Memory size in this device is 4194304 bytes. It is enough for the use of a certificate with a minimum of code for the essential functioning of the device as an NTP clock.

In addition of the code and the certificates (average of 5.12 Kbytes per certificate), using NTS relies on the creation of a library that is responsible for creating the TLS tunneling, using the certificates for signing and authenticating the communication as well as introducing NTP packets inside the TLS tunnel with signed fields. One of the libraries allowing the using of TLS is BearSSL[2]. A minimal server implementation using this library can run about 20 kilobytes of compiled code and 25 kilobytes of RAM leaving more space for tunneling and NTP to create an NTS library based on it.

This rundown of memory size available in an Arduino device only showcases how it is possible to use an implementation of NTS secure NTP protocol in more advanced resources we have in everyday IoT devices.

[2]https://bearssl.org/

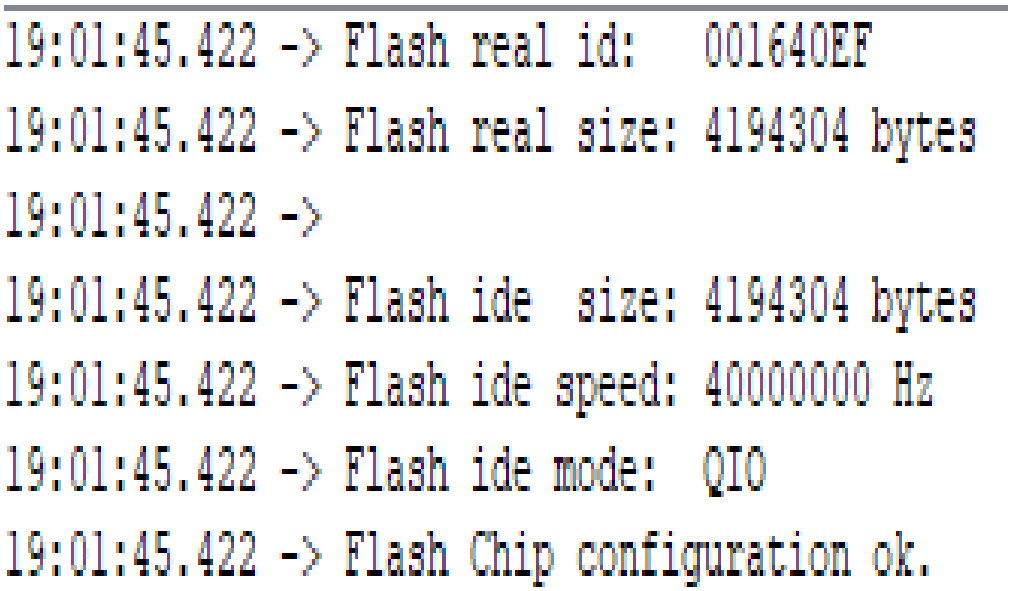


Figure 4.22 ESP8266 memory size

4.6 Conclusion

We have been able to tamper with the time service in different types of IoT device sets by shifting time according to attack schedules. We were able to confuse connected devices and push them towards either being delayed or early in doing their tasks from the first synchronization request. This highlights how insecure and easy to access IoT NTP communications are, thus motivating the need for including more correct time synchronization security features.

For this purpose, we proposed two different ways of securing NTP time synchronization in IoT devices. The first was inclusive to even existing devices that are today in customer homes. The second architecture deals with including a secure implementation of NTP such as NTS inside IoT devices.

CHAPTER 5 CONCLUSION

In this chapter, we discuss the results of our experiments as well as their possible effect on the day to day use of IoT devices. We also review the accomplishment of our research objectives and discuss the possibilities our work offers for future advances in the area of IoT time synchronization security. Finally, we list the limitations and challenges we faced in the process of working on this thesis project.

5.1 Overview of results

To study NTP security in IoT devices and to demonstrate its importance, we have studied an existing dataset of 50 IoT devices [11] as well as built 3 different testing environments. Based on the analysis of the dataset information we concluded that NTP is broadly used in IoT devices and we observed how they use it. We also did an extensive investigation into the reasons why this protocol is important from a security point of view and concluded that it is critical. Our investigation not only included the dataset of 50 devices but it also discussed several devices used in different fields in human daily life. This study gave way to a categorization of NTP use in IoT:

- Scheduling tasks
- Authentication / encryption / networking protocols
- Logs / monitoring

This categorization is based on a breakdown of reasons why the devices make NTP queries and where the information received from their servers is used later on in either task that the user can notice such as schedules or tasks that are essential to the functioning of different network stack protocols or finally information that is stored inside the device for a later purpose such as log files.

Out of the 50 devices in the first dataset, we had 39 devices that used NTP for time synchronizations which are 78% of the variety of devices we had. This goes to showcase even more the important use of this protocol in IoT networks.

Based on the 50-device dataset we have also concluded the existence of different patterns of NTP port choice in IoT devices depending on the design and security priorities chosen. these patterns are:

- Multiple ports during the synchronization period
  - Random port number for every request
  - Random port number for every few (2 to 8) requests
  - Random first port number choice with a predefined pattern in choosing the next ports
- One port during the synchronization period
  - Port 123 for all requests
  - One random port depending on the vendor's choice different than port 123
  - One NTP request during the whole time of use, thus one port choice

This led us to think about the non-randomization of NTP port choice effect on IoT security as it is an important part of the 5-tuple concept and could be used for spoofing attacks. These attacks are made especially easier because NTP is a UDP protocol, therefore connection-less, is coupled with the attacker knowing the rest of the 5-tuple components.

From analyzing these patterns, we notice that, to a certain extent, these devices show more variety and randomization in port choice than regular network devices which are set to use the default UDP port number 123. IoT devices have different algorithms in picking their NTP source port for NTP queries. The risk in these choices goes up as the randomization value goes down. For devices using the same UDP port number 123 or a UDP port that is predefined by the vendors and never changing, it is higher. Certain devices are more secure because for every NTP request they use a different, randomly chosen UDP port.

It is important to note that these different algorithms and patterns in choosing NTP UDP source ports can only add more variety to the pool of IoT devices. As the NTP community is used to port 123 being used for NTP communications, certain firewalls are specifically configured to only allow NTP requests or responses from this port. This practice can lead to certain IoT NTP queries being dropped and so these devices either drift in time depending on their reference / local clocks or they are out of service if their network stack depends on time information reception.

Another component of the 5-tuple that we have analyzed in this work is the choice of the server IP address in NTP IoT communications. Out of the 39 devices using NTP, at least 26 (66.66%) were using servers from public vendor (GOOGLE, MICROSOFT) or volunteer-managed pools and at least 15 devices (38.6%) were using servers from the *pool.ntp.org* pool.

This goes to say that the presence of public servers in the NTP communications is predominant. It is also important to note that these were unauthenticated servers/communications, including the ones with NIST servers.

As for the patterns of choosing NTP servers, it depends on many factors such as the pool hard-coded into the device, the pools' rules, the DNS servers' responses including NTP servers' IP addresses as well as possible restrictions added by IoT vendors. The latter ones can force devices to connect to specific servers, sometimes open to the public. This process of hard coding an NTP server or more in devices made by specific companies can lead to spoofing attacks such the use of the KoD (Kiss of Death) NTP packet with spoofed known NTP server to stop the device from requesting time or to render the server itself out of service (DoS attack).

We can compare 3 different situations regarding hard coding servers IPs in an IoT device:

1. Hard-coding one open public NTP server IP address in an IoT device.
2. Hard-coding an NTP pool in an IoT device.
3. Hard-coding an IP address of a server that only accepts authenticated requests (authenticated NTPv4 or NTS servers).

The first option leads to easier DoS attacks as the server component of the 5-tuple is known. the second option can also lead to a DoS attack, but it is harder to achieve. This is explained by the number of servers in the pool and the randomization factor of the algorithm used to pick the NTP server by the device can make the possibility of DoS attack even less likely. It takes longer and more spoofed queries from the attacker to make send a packet with the right server IP from the pool to achieve a DoS attack.

The third option used a secure NTP connection. This type of connection requires encryption for an authenticated query to be accepted and a response to be issued and accepted by the device as well. It is even less likely, although possible, for an attacker to successfully target such communication. None of the devices in the dataset and test environments used authenticated NTP queries. This is a security issue in both regular and IoT networks.

When analyzing the NTP packets of the communications in the different test sets, we concluded the effect of not having an RTC in most IoT devices or any reference clock was the reason behind the NULL value of all timestamps in NTP communication. These timestamps are used by NTP to verify basic checks to authenticate servers and clients. The NULL value makes it impossible to properly complete these checks and The IoT device accepts the first

time response it gets from the server's IP. Thus, it is easier to shift time in IoT devices than it is in regular networks.

We then set up 3 different sets to test IoT devices with a time shifting attack using the tool Delorean. This tool targets devices using unauthenticated NTP for time synchronization. The first set is that of 3 virtual machines (NTP server, Ubuntu client, Kali attacking machine) to test the reaction of a regular network to the attack. The second set is an Arduino IoT with a kali attacking machine and a gateway. This device uses a public NTP server similar to the third set which replaces the Arduino device with a smart TP-Link light bulb.

We studied the feasibility of two types of attacks regarding these 3 sets: On-path and Off-path time shifting attack. As we conducted the on-path attack, we tested different time shifts as shown on figure 5.1:

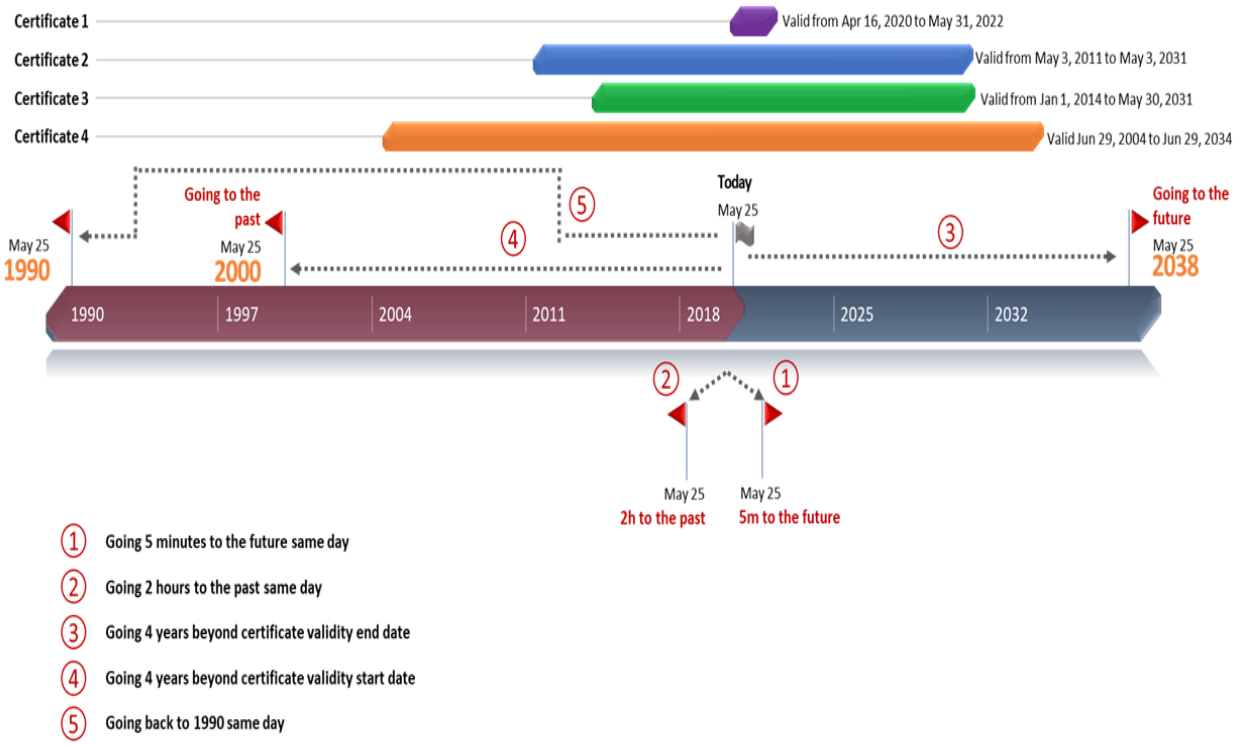


Figure 5.1 On-path attack time shifts

All these attacks succeeded and demonstrated the ease of time shifting in IoT devices as well as how it can affect these devices compared to regular networks.

After finishing the attacks, we discussed two types of possible solution architectures that can protect IoT devices from NTP time shifting attacks. The first architecture is backwards

compatible and focuses on securing the elements surrounding IoT devices to be more user-friendly and to include the devices already in use. This architecture uses an NTP gateway box that can be included in the router or separated but to which the IoT devices are connected first. This box needs to support NTS for NTP communication authentication and IP spoofing prevention. All the NTP requests coming from the IoT devices connected to it will be redirected to a trusted NTP server (accepting authenticated requests only) either internal or public.

The second architecture for NTP communication security in IoT devices requires adding changes to the way IoT devices use this protocol and including a new implementation of NTS. We have checked TLS and certificates requirements for use in IoT devices and one way of including the is to use certificate fingerprints rather than a full certificate to not take as much memory space. It is also important to note that the use of a reference clock in IoT devices is important to avoid security issues caused by the NULL timestamps value.

5.2 Overview of thesis objectives

IoT devices are considered an important part of our lives in all fields. They are used in households as well as outside in hospitals, schools, military, sports, and the streets. They are also becoming more trusted in the heart of tasks on which human lives can depend such as connected pacemakers or connected traffic lights. If either one of these two example devices has an issue or is tampered with, it could harm us greatly.

A great part of the correct functioning of these devices is the correctness of the time they have. This also depends on their correct and secure time synchronization. This feature has been studied and proved to be vulnerable in traditional networks but, as far as our research is concerned, it has not been tested in IoT networks.

Under these circumstances, we started this thesis project with three main objectives :

1. Defining the range of importance of time in services provided by IoT devices and their importance in human daily use;

2. Drilling down through IoT time synchronization communications and their vulnerabilities;

3. Elaborating a possible protection/security architecture that takes into consideration the time component as well as the mobility service of IoT devices.

At the end of this work we have achieved each of the objectives respectively in the following

way :

1. We have studied existing IoT communication data sets as well as our environments to analyze the importance and the roles time synchronization plays in IoT usage. We have especially focused on the nature of protocols and tasks that require accurate time from NTP. We have also worked on showing the possible impact tampering with time in IoT networks could have.

2. In attempting to showcase how vulnerable NTP is in IoT networks, we have analyzed its use of the 5-tuple components and how predictable the patterns of choosing them can be. More than 60% of the devices studied use known public servers and predictable rules of choosing NTP port if not going for the default port 123. In addition, we have shown how the absence of an RTC inside these devices leads to additional insecurity lying in the NULL value of timestamps and voiding necessary NTP checks. After these packets checks, we have set a testing environment to apply attacks on IoTs with the tool Delorean (making use of the unauthenticated nature of NTP communications) that are used on regular networks. We were thus able to demonstrate and show NTP vulnerabilities through time shifting and changing the correct functioning of a preset schedule to obey the schedule set by the attacker.

3. Based on the vulnerabilities noticed in the studied devices, we have analyzed and suggested two possible architectures for secure IoT NTP communication. The first is backwards compatible and includes already in use connected devices through the addition of an NTP gateway for authenticated and secure NTP queries. The second architecture requires adding NTS authentication features into IoT devices.

The possible future goals of this work include but are not limited to working on creating an implementation of NTS for IoT networks' use and testing it with the attacks we have tried so far. This also includes working on elaborating a more precise implementation of the protocol TLS/SSL.

5.3 Open questions/remaining challenges

After showing the importance of NTP and its security in IoT networks and suggesting possible solution architectures as well as their feasibility, a few challenges are left to tackle. These challenges require more time to put into work. We can mention:

- Finding a way to force existing IoT devices to randomize their choice of NTP port numbers.

- Making a more inclusive list of all networking protocols that depend on time accuracy and acquire it from NTP.
- Including NTP authentication measure in existing IoT devices.
- Helping IoT industry professionals as well as the vendors to include NTP security checks in their devices.

Most of the challenges for Future IoT advancements is related to including security measures in the already in production IoT devices. To do so, it is important to put into action and verify the backwards compatible security solution architectures.

5.4 Limitations of thesis

While working on this research project we were able to come to several conclusions important in NTP security in IoT networks as shown in the previous sections. We also had a few limitations due to time constraints as well as the current world environment. These limitations can be summarized in what follows:

- **More 5-tuple patterns**: Although the patterns noticed for NTP port choice and server choice were analyzed on 50 devices, it is not possible to say that this is the most inclusive list of patterns possible. A bigger dataset would be needed to do such analysis as well as support from vendors to have access to their devices' software as well as their hardware
- **Application of off-path attack**: Because the off-path attacks were already shown to be possible through an example of amplification attacks known for NTP, we have analyzed of the feasibility of these attacks without being able to try them in the 3 sets we have built.
- **Application of security measures**: We have conducted a feasibility analysis for both security measures as well by studying their different components but were not able to test them as they require extensive coding in the form of the development of an implementation of NTS in Arduino.

The IoT field is diverse. We demonstrated how vulnerable its implementation of NTP protocol can be. As we have experimented with attacks on both scheduling services and TLS implementation, we concluded that certain elements can be used in IoT devices to help solve security issues. We can introduce an RTC as well as an implementation of NTS to at least

have the device do more checks before accepting time during synchronization cycles. Developers and manufacturers should include NTP synchronization in their security priorities list.

Connected Smart devices will continue to be used, especially with the ever-evolving needs humanity has, to conform to our environment. The latest of these devices include smart ventilators being worked on to help assist medical staff in curing COVID-19 patients. If these ventilators were to depend on time synchronization as the connected devices we have studied in this research work do, the patients can run an additional risk of being at the mercy of cyber attackers. Taking the security of time synchronization into consideration is a step that needs to be done to assure a safe usage and dependency on IoT devices.

REFERENCES

[1] U. Nations, Ed., *Long-Range World Population Projections: Two Centuries of Population Growth, 1950-2150*. New York: Dept. of International Economic and Social Affairs, United Nations, 1992.

[2] B. Safaei, A. M. Monazzah, M. Bafroei, and A. Ejlali, "Reliability Side-Effects in Internet of Things Application Layer Protocols," Dec. 2017.

[3] D. Reinsel, J. Gantz, and J. Rydning, "The Digitization of the World from Edge to Core," p. 28, 2018.

[4] D. Sibold, D. Franke, R. Sundblad, M. Dansarie, and K. Teichel, "Network Time Security for the Network Time Protocol," https://tools.ietf.org/html/draft-ietf-ntp-using-nts-for-ntp-28.

[5] K. K. Patel, S. M. Patel, and P. Scholar, "Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges," p. 11, 2016.

[6] D. Barrera, I. Molloy, and H. Huang, "IDIoT: Securing the Internet of Things like it's 1994," *arXiv:1712.03623 [cs]*, Dec. 2017.

[7] B. A. Desai, D. M. Divakaran, and I. Nevat, "A feature-ranking framework for IoT device classification," p. 8.

[8] M. Gigli and S. Koo, "Internet of Things: Services and Applications Categorization," *Advances in Internet of Things*, vol. 01, no. 02, pp. 27–31, 2011.

[9] H. Zuerner, "The Internet of Things as greenfield model: A categorization attempt for labeling smart devices," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*. Seoul, Korea (South): IEEE, Mar. 2014, pp. 5–9.

[10] A. Keranen, M. Ersue, and C. Bormann, "Terminology for Constrained-Node Networks," https://tools.ietf.org/html/rfc7228.

[11] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "SoK: Security Evaluation of Home-Based IoT Deployments," p. 19.

[12] M. M. Hossain, M. Fotouhi, and R. Hasan, "Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things," in *2015 IEEE World Congress on Services*. New York City, NY, USA: IEEE, Jun. 2015, pp. 21–28.

[13] D. Geneiatakis, I. Kounelis, R. Neisse, I. Nai-Fovino, G. Steri, and G. Baldini, "Security and privacy issues for an IoT based smart home," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Opatija, Croatia: IEEE, May 2017, pp. 1292–1297.

[14] A. Gai, S. Azam, B. Shanmugam, M. Jonkman, and F. De Boer, "Categorisation of security threats for smart home appliances," in *2018 International Conference on Computer Communication and Informatics (ICCCI)*. Coimbatore: IEEE, Jan. 2018, pp. 1–5.

[15] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of Things for Smart Healthcare: Technologies, Challenges, and Opportunities," *IEEE Access*, vol. 5, pp. 26 521–26 544, 2017.

[16] O. Garcia-Morchon, S. S. Kumar, and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges," https://tools.ietf.org/html/rfc8576.

[17] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power IoT edge devices," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. Reston, VA, USA: IEEE, Dec. 2016, pp. 7–12.

[18] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync Protocol for Sensor Networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 138–149.

[19] J. He, J. Chen, P. Cheng, and X. Cao, "Secure Time Synchronization in WirelessSensor Networks: A MaximumConsensus-Based Approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 1055–1065, Apr. 2014.

[20] Y. Liu, J. Li, and M. Guizani, "Lightweight secure global time synchronization for Wireless Sensor Networks," in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2012, pp. 2312–2317.

[21] T. OConnor, W. Enck, and B. Reaves, "Blinded and confused: Uncovering systemic flaws in device telemetry for smart-home internet of things," in *Proceedings of the 12th*

Conference on Security and Privacy in Wireless and Mobile Networks - WiSec '19. Miami, Florida: ACM Press, 2019, pp. 140–150.

[22] J. Postel and K. Harrenstien, "Time Protocol," https://tools.ietf.org/html/rfc868.

[23] "IEEE 1588-2002 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," https://standards.ieee.org/standard/1588-2002.html.

[24] L. Montini, T. Frost, G. Dowd, and V. Shankarkumar, "Precision Time Protocol Version 2 (PTPv2) Management Information Base," https://tools.ietf.org/html/rfc8173.

[25] S. T. Watt, S. Achanta, H. Abubakari, E. Sagen, Z. Korkmaz, and H. Ahmed, "Understanding and applying precision time protocol," in *2015 Saudi Arabia Smart Grid (SASG)*. Jeddah, Saudi Arabia: IEEE, Dec. 2015, pp. 1–7.

[26] J. Tsang and K. Beznosov, "A Security Analysis of the Precise Time Protocol (Short Paper)," in *Information and Communications Security*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, P. Ning, S. Qing, and N. Li, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 4307, pp. 50–59.

[27] D. Mills, "Internet time synchronization: The network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, Oct./1991.

[28] "Ntp.org: Home of the Network Time Protocol," http://www.ntp.org/.

[29] D. L. Mills, "Network Time Protocol (NTP)," https://tools.ietf.org/html/rfc958.

[30] J. Burbank, D. M. <>, and W. K. <>, "Network Time Protocol Version 4: Protocol and Algorithms Specification," https://tools.ietf.org/html/rfc5905.

[31] Mills, "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI," https://tools.ietf.org/html/rfc4330#page-6.

[32] D. Mills, "Measured performance of the Network Time Protocol in the Internet system," RFC Editor, Tech. Rep. RFC1128, Oct. 1989.

[33] A. Malhotra, I. E. Cohen, E. Brakke, and S. Goldberg, "Attacking the Network Time Protocol," in *Proceedings 2016 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2016.

[34] H. Stenn, "Securing network time protocol," *Communications of the ACM*, vol. 58, no. 2, pp. 48–51, Jan. 2015.

[35] J. Selvi, "Bypassing HTTP Strict Transport Security," p. 4.

[36] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir, "Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks," in *Proceedings of the 2014 Conference on Internet Measurement Conference - IMC '14*. Vancouver, BC, Canada: ACM Press, 2014, pp. 435–448.

[37] T. Mizrahi <talmi@marvell.com>, "Security Requirements of Time Protocols in Packet Switched Networks," https://tools.ietf.org/html/rfc7384.

[38] H. Stenn, D. Sibold, and D. Reilly, "Network Time Protocol Best Current Practices," https://tools.ietf.org/html/rfc8633#section-5.

[39] B. Dowling, D. Stebila, and G. Zaverucha, "Authenticated Network Time Synchronization," p. 19.

[40] "Instructions for using the NIST authenticated Network Time Protocol (NTP) server," p. 4.

[41] S. Röttger, "Analysis of the NTP Autokey Procedures," p. 38.

[42] H. M. Ramos, P. M. Ramos, and P. Paces, "Development of a IEEE 1451 Standard Compliant Smart Transducer Network with Time Synchronization Protocol," in *2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007*. Warsaw, Poland: IEEE, May 2007, pp. 1–6.

[43] K. Shahzad, A. Ali, and N. Gohar, "ETSP: An Energy-Efficient Time Synchronization Protocol for Wireless Sensor Networks," in *22nd International Conference on Advanced Information Networking and Applications - Workshops (Aina Workshops 2008)*. Ginowan, Okinawa, Japan: IEEE, 2008, pp. 971–976.

[44] R. N. Gore, N. Elizabeth, D. Dzung, and S. Ashok, "Towards Robust Synchronization in IoT Networks," in *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*. Bengaluru, India: IEEE, Jan. 2019, pp. 678–683.

[45] D. Barrera, I. Molloy, and H. Huang, "Standardizing IoT Network Security Policy Enforcement," in *Proceedings 2018 Workshop on Decentralized IoT Security and Standards*. San Diego, CA: Internet Society, 2018.

[46] A. Malhotra, W. Toorop, B. Overeinder, R. Dolmans, and S. Goldberg, "The Impact of Time on DNS Security," Tech. Rep. 788, 2019.

[47] T. Rytilahti, D. Tatang, J. Kopper, and T. Holz, "Masters of Time: An Overview of the NTP Ecosystem," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. London: IEEE, Apr. 2018, pp. 122–136.

[48] D. Sam, C. Velanganni, and T. E. Evangelin, "A vehicle control system using a time synchronized Hybrid VANET to reduce road accidents caused by human error," *Vehicular Communications*, vol. 6, pp. 17–28, Oct. 2016.

[49] "Bitdefender BOX - Home Network Security for All Connected Devices," https://www.bitdefender.com/box/.

[50] M. McCormack, A. Vasudevan, G. Liu, S. Echeverría, K. O'Meara, G. Lewis, and V. Sekar, "Towards an Architecture for Trusted Edge IoT Security Gateways," p. 10.

[51] M. W. Condry and C. B. Nelson, "Using Smart Edge IoT Devices for Safer, Rapid Response With Industry IoT Control Operations," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 938–946, May 2016.

[52] D. Mills, "Simple Network Time Protocol (SNTP)," https://tools.ietf.org/html/rfc1361.

[53] E. F. Dierikx, A. E. Wallin, T. Fordell, J. Myyry, P. Koponen, M. Merimaa, T. J. Pinkert, J. C. J. Koelemeij, H. Z. Peek, and R. Smets, "White Rabbit Precision Time Protocol on Long-Distance Fiber Links," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 63, no. 7, pp. 945–952, Jul. 2016.

[54] W. Dong and X. Liu, "Robust and Secure Time-Synchronization Against Sybil Attacks for Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 6, pp. 1482–1491, Dec. 2015.

[55] Z. Fang, H. Fu, T. Gu, Z. Qian, T. Jaeger, and P. Mohapatra, "FORESEE: A Cross-Layer Vulnerability Detection Framework for the Internet of Things," p. 9.

[56] K. Fan, Y. Ren, Z. Yan, S. Wang, H. Li, and Y. Yang, "Secure Time Synchronization Scheme in IoT Based on Blockchain," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Jul. 2018, pp. 1063–1068.

[57] S. Ganeriwal, C. Pöpper, S. Čapkun, and M. B. Srivastava, "Secure Time Synchronization in Sensor Networks," *ACM Transactions on Information and System Security*, vol. 11, no. 4, pp. 1–35, Jul. 2008.

[58] "Hjp: Doc: RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification," http://www.hjp.at/doc/rfc/rfc5905.html.

[59] T. Kawamura, M. Fukushi, Y. Hirano, Y. Fujita, and Y. Hamamoto, "A Network-Based Event Detection Module Using NTP for Cyber Attacks on IoT," in *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*. Takayama: IEEE, Nov. 2018, pp. 86–91.

[60] T. Kida, L. R. Yoshioka, S. Takahashi, and H. Kaneda, "Theory on extended form of interpolatory approximation of multidimensional waves," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 75, no. 4, pp. 26–39, 1992.

[61] M. V. Larsen and F. Gont, "Recommendations for Transport-Protocol Port Randomization," https://tools.ietf.org/html/rfc6056.

[62] M. Lichvar, G. Gont, and F. Gont, "Port Randomization in the Network Time Protocol Version 4," https://tools.ietf.org/html/draft-ietf-ntp-port-randomization-00.

[63] A. Malhotra and S. Goldberg, "Message Authentication Code for the Network Time Protocol," https://tools.ietf.org/html/rfc8573.

[64] A. Malhotra and D. Franke, "NTP Client Data Minimization," https://tools.ietf.org/html/draft-ietf-ntp-data-minimization-04.

[65] D. L. Mills, *COMPUTER NETWORK TIME SYNCHRONIZATION : The Network Time Protocol on Earth and in Space*, second edition ed. CRC Press, 2011.

[66] D. Mills, *Computer Network Time Synchronization: The Network Time Protocol on Earth and in Space, Second Edition*. CRC Press, 2017.

[67] D. L. Mills, "Network Time Protocol (NTP) General Overview," p. 22.

[68] B. Moussa, M. Debbabi, and C. Assi, "Security Assessment of Time Synchronization Mechanisms for the Smart Grid," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1952–1973, 23.

[69] J. Postel, "User Datagram Protocol," https://tools.ietf.org/html/rfc768.

[70] J. A. Sherman and J. Levine, "Usage Analysis of the NIST Internet Time Service," *Journal of Research of the National Institute of Standards and Technology*, vol. 121, p. 33, Mar. 2016.

[71] Shulong Wang, Yibin Hou, Fang Gao, and Songsong Ma, "A novel clock synchronization architecture for IoT access system," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. Chengdu, China: IEEE, Oct. 2016, pp. 1456–1459.

[72] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, Aug. 2019.

[73] "World Population Prospects - Population Division - United Nations," https://population.un.org/wpp/DataQuery/.

[74] J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, and Y. Jin, "Security analysis on consumer and industrial IoT devices," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. Macau: IEEE, Jan. 2016, pp. 519–524.

[75] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A Survey on Security and Privacy Issues in Internet-of-Things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, Oct. 2017.

[76] K. S. Yildirim and A. Kantarci, "Time Synchronization Based on Slow-Flooding in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 244–253, 2014.

[77] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "IoT Security: Ongoing Challenges and Research Opportunities," in *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*. Matsue, Japan: IEEE, Nov. 2014, pp. 230–234.

[78] H. Zhao, L. Li, H. Peng, J. Xiao, Y. Yang, and M. Zheng, "Fixed-time synchronization of multi-links complex network," 2017.

[79] K. Zhao and L. Ge, "A Survey on the Internet of Things Security," in *2013 Ninth International Conference on Computational Intelligence and Security*, Dec. 2013, pp. 663–667.

APPENDIX A LCD-CLOCK IOT ARDUINO CODE

In order to create our LCD clock IoT device based on NTP, we used Arduino to synchronize our NodeMCU V3 ESP8266 to reach 2.canada.pool.ntp.org and request time from it.

```
#include <Time.h>
#include <TimeLib.h>
#include <NTPClient.h>
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,20,4);

const char *ssid     = "my_hotspot";
const char *password = "iotattack2020";

int GMT = 5;
int DAY, MONTH, YEAR, Seconds, Hours;
long int time_now;
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "2.canada.pool.ntp.org");

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  lcd.begin(20,4);
  lcd.clear();
  lcd.init();
  lcd.backlight();
  lcd.setCursor(5, 0);

  while ( WiFi.status() != WL_CONNECTED ) {
    delay ( 500 );
    Serial.print ( "." );
```

```
31    }
32    timeClient.begin();
33  }
34
35  void loop() {
36    timeClient.update();
37    time_now = timeClient.getEpochTime() - 14400;
                                      //get epoch time and adjust it
        according to your GMT
38    DAY = day(time_now);
39    MONTH = month(time_now);
40    YEAR = year(time_now);
41    Hours = hour(time_now);
42    Seconds = second(time_now);
43
44    LCD_DISPLAY(String(DAY), String(MONTH), String(YEAR));
45     delay(1000);
46  }
47
48  void LCD_DISPLAY(String DAY, String MONTH, String YEAR)
49  {
50    lcd.setCursor(5, 0);
51    if (Hours > 9)
52      lcd.println(String(Hours) + ":" + timeClient.getMinutes() + "
          :");
53    if (Hours < 10)
54      lcd.println("0" + String(Hours) + ":" + timeClient.getMinutes
          () + ":");
55
56    lcd.setCursor(11, 0);
57    lcd.println(String(Seconds));
58
59    lcd.setCursor(14, 0);
60    if (isPM(time_now))
61      lcd.println("PM");
62    if (isAM(time_now))
```

```
63      lcd.println("AM");
64
65    lcd.setCursor(6, 1);
66    lcd.println(DAY + ":" + MONTH + ":" + YEAR);
67  }
```

APPENDIX B NTP POOLS NAMES

The text file servername.txt used in section 3.3.4 for regional distribution of NTP servers contains the following NTP pool names:

0.africa.pool.ntp.org

1.africa.pool.ntp.org

2.africa.pool.ntp.org

3.africa.pool.ntp.org

0.asia.pool.ntp.org

1.asia.pool.ntp.org

2.asia.pool.ntp.org

3.asia.pool.ntp.org

0.europe.pool.ntp.org

1.europe.pool.ntp.org

2.europe.pool.ntp.org

3.europe.pool.ntp.org

0.north-america.pool.ntp.org

1.north-america.pool.ntp.org

2.north-america.pool.ntp.org

3.north-america.pool.ntp.org

0.oceania.pool.ntp.org

1.oceania.pool.ntp.org

2.oceania.pool.ntp.org

3.oceania.pool.ntp.org

0.south-america.pool.ntp.org

1.south-america.pool.ntp.org

2.south-america.pool.ntp.org

3.south-america.pool.ntp.org

APPENDIX C NTP ATTACKS IN REGULAR NETWORKS

Tables C.1 and C.2 explain the different attacks on NTP vulnerabilities, how they work and in some cases the possible security measures that can be taken to counter them according to Malhotra and al. [33].

Table C.1 Denial of service and fragmentation attacks on NTP in regular networks

Attack concept	Kiss-Of-Death : Denial of service attack (KOD packet makes the client refrain from requesting time from its server for some time)			Off-path NTP fragmentation attack
Attack	Exploiting KoD packet	Low rate off path DoS attack on ntp clients	Pinning to a bad timekeeper	Off-path NTP fragmentation attack
How to	Spoof KoD packet (Only need IPs of client and server) + setting poll value on spoofed KoD to high value to stop client from getting time from server for a long time. Priming the pump : spoof mode 3 client requests to make them look from server like the client has too many requests and wait for the client to send the last request before the KoD from the server.	Off path attacker can use this to turn off NTP from the client which will be pushed to use local clock or drift during the attack : - attacker sends mode 3 NTP query to victim client which responds with mode 4 NTP response : giving attacker client's server from ref ID -attacker makes the server send KoD packet to client to stop sync for some time. If client gets another server, attacker does the same again. Attacker learns IPs of all client servers and does a KoD attack periodically to stop client from sync	Same process from previous attack to learn about all servers and check them to see if they are good or bad using a mode 3 query : the client at this moment should be sync-ing to the bad timekeeper	Attack works against specific clients that use specific classes of ipv4 fragmentation policies + client is configured with only one server
Notes	If priming the pump is used then attacker can't control polling interval of KoD message			Attack surface small but non negligible
Security	Forcing validation of origin timestamp on client side as well as server side to avoir PiTM attacks + authenticate all communication with the server		Authentication to avoid spoofing	Avoid fragmentation of packets

Table C.2 Stepping time attacks on NTP in regular networks

Attack concept	STEP time with NTP	
Attack	Time skimming	Exploiting reboot
How to	On-path traffic highjacking And Step threshold (S.T) smaller than STEP smaller than panic threshold (P.T)	__g option allows client when initialized and before sync to accept any time shift (even more than panic T). On-path attacker knows exact time of reboot bcs of INIT in ref ID of all NTP packets + shift before sync : feel free to panic. Small step big step: clients reboot + init start + sync server + server sends decent step bigger than S.T and smaller than P.T + client accepts sending "STEP" in ref ID + Server sends big step because allow_panic never set to false (2 syncs with less than S.T turn allow_panic to false). Stealthy time shift: use small step big step to expire an object on the client and get back to regular time.
Notes	Takes too long and is complicated Stepout went from 15 to 5 minutes P.T is 16 minutes S.T is 125 ms	Can make the client shift for a year in one step Attacker has to be on path for it to work
Security	Authentication to avoid spoofing	Remove __g option Set allow_panic to flse after 1st clock update No INIT in ref ID