Institutionen för datavetenskap
Department of Computer and Information Science

Examensarbete

A Security Analysis of Wireless Smart Home Technologies

av

Ludvig Fischerström, Niclas Hansson, Alexander Lantz

LIU-IDA/LITH-EX-G--14/084--SE

2015-01-08

Linköpings universitet

Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings universitet
581 83 Linköping

Linköpings universitet
Institutionen för datavetenskap

Examensarbete

A Security Analysis of Wireless Smart Home Technologies

av

Ludvig Fischerström, Niclas Hansson, Alexander Lantz

LIU-IDA/LITH-EX-G--14/084--SE

2015-01-08

Handledare: Marcus Bendtsen
Examinator: Nahid Shahmehri

A Security Analysis of Wireless Smart Home Technologies

Ludvig Fischerström
ludfi239

Niclas Hansson
nicha094

Alexander Lantz
alela654

January 8, 2015

Students in the 5 year Information Technology program complete a semester-long software development project during their sixth semester (third year). The project is completed in mid-sized groups, and the students implement a mobile application intended to be used in a multi-actor setting, currently a search and rescue scenario. In parallel they study several topics relevant to the technical and ethical considerations in the project. The project culminates by demonstrating a working product and a written report documenting the results of the practical development process including requirements elicitation. During the final stage of the semester, students create small groups and specialise in one topic, resulting in a bachelor thesis. The current report represents the results obtained during this specialization work. Hence, the thesis should be viewed as part of a larger body of work required to pass the semester, including the conditions and requirements for a bachelor thesis.

Abstract

The use of electronics connected to local networks and the Internet is growing all the time. Nowadays you can control your electronics in your house even when away from home, which opens up for potential security threats. The purpose of this report is to point out the potential risks with connecting home electronics to the Internet and to shed light on what security mechanisms that are needed in these kinds of systems. This report contains a theoretical part in which relevant material has been summarized. This material includes the smart home solution Tellstick Net and the wireless technologies ZigBee and Z-Wave, which are commonly used in home automation. The Tellstick Net system was mapped out and a risk analysis with *attack trees* was performed. After the analysis of the system, the implementation of two potential security threats were attempted. The two attempted attacks were *replay attack* and *cross-site request forgery*. The replay attack was unsuccessful due to the way the system communicates and keeps connections alive. However, the cross-site request forgery was discovered to be successful in some cases. It depended on if the browser of the target supported *cross-origin resource sharing*, as that property protects against cross-site request forgery. Finally, the report discusses what impact the found security deficiencies have, what they entail and how they reflect on the need for security in smart technologies for the home.

Contents

List of Figures

List of Tables

1 Introduction

Smart technologies keep expanding and new fields of application are continuously being developed. Remotely controlling your home via the Internet is one such field of application. This is accomplished by utilizing smart devices that relay data to physical objects such as electrical sockets. In turn, this will allow any electrical device that can be connected to the socket to consequently be controlled remotely. Physical devices connected to the Internet is commonly referred to as the *Internet of Things* [1]. The Internet of Things opens up for a lot of possibilities. However, as new technologies becomes available new security threats emerge with it. When communicating through the Internet one becomes vulnerable to attacks such as *man in the middle* or *denial of service*. Extensive research regarding security issues concerning the Internet of Things have already been made, outlining hidden dangers and security risks [2][3].

1.1 Motivation

Smart technologies are starting to make their way into our homes and as Google and other large companies get involved in the business of smart homes, this product category is sure to grow and in the near future become an integrated part of the home of everyday people. It has become evident that surveillance of ordinary people is happening and when more smart products make their way into our homes, security risks including integrity violations are an increasing concern [4]. The growth over the last few years in the area of Internet of Things provide further reasons for extended security, as insufficiencies will result in intruders gaining access to physical objects. This can compromise the security of the physical, intellectual or ethical values of the individuals or corporations utilizing these new products. It is not hard to imagine considerable consequences of an intruder gaining access to e.g. a smart heat pump or security camera.

1.2 Purpose

The primary purpose with this report is to establish the necessity for well thought-out security policies and mechanisms when it comes to design of smart technologies for the home.

1. By analyses we aim to find security deficiencies in a Tellstick Net device and surrounding system to discover how it could be exploited.
2. If deficiencies are found we aim to implement attacks to exploit the system.
3. After implementation of an attack we mean to set the result in context to conclude what could be done to avoid security leaks.

1.3 Limitations

Before reading this report one should be aware that the analyses are limited to issues within network communications, software and human-technology interactions. We will not be looking into opportunities of exploiting hardware.

2 Theory and Related Work

In this section we introduce a few smart home technologies, among them is the *Tellstick Net* system that we later will investigate further by performing a risk and system analysis in Section 3.1 and 3.2. We also present some research that has been made regarding issues concerning Internet security and especially security in the area of Internet of Things.

2.1 Tellstick Net

Tellstick Net is a device that lets you remotely control your connected electronics via the Internet. It is compatible with many different remote socket receivers. This gives you high control of your home even from a remote location. Tellstick Net transmits and receives signals at a rate of 433.92 MHz. Receivers respond to this by turning on and off their electricity infusion, which in turn lets you remotely control certain parts of your home, such as lamps, music and thermostats. In this report a receiver is referred to as an electrical device capable of receiving short range radio waves and being set up with a device such as the Tellstick Net to be part of a smart home system [5][6].

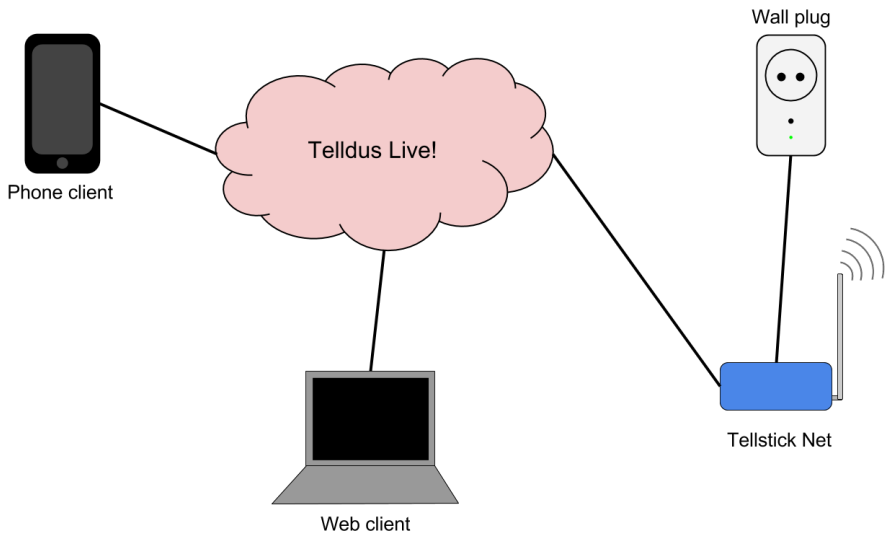

Figure 1: Tellstick Net system

Tellstick Net is accessible via *Telldus Live!*, which is Telldus' online service. This makes it possible to login and control the receivers from a remote location. This is accomplished by connecting your Tellstick Net to your local network and

be able to communicate in your home. This means you can turn your electronics on and off even when you are away from home. The described setup can be seen in Figure 1. In the figure, the wall-plug is a receiver that communicate with the Tellstick Net device. The Tellstick Net receives commands from either the phone or web client. When a client sends a command the communication goes through Telldus Live! that works as an intermediary.

As far as we are aware, no previous work has been published on the subject of risk analysis of the Tellstick technology.

2.2 Z-Wave

Z-Wave is a wireless protocol stack used in home automation. Z-Wave is a proprietary protocol licensed by Sigma Designs, Inc. The Physical and MAC layer of the protocol stack has been defined as an ITU standard called G.9959 [7][8].

Z-Wave can be run in the 868.42 MHz (Europe) and the 908.42 MHz (United States) frequency band with FSK or G-FSK modulation and Manchester or NRZ coding depending on which version and send data at either 9.6 kbps or 40 kbps [9]. Z-Wave supports mesh networks meaning that the different nodes can communicate directly with each other and does not need a central node controlling them. A central node in a Z-Wave network is commonly used to connect it and let the entire network gain access to the Internet [9]. The protocol supports both encrypted and unencrypted communication where the encrypted version uses AES-128. If the unencrypted version is used, the packages sent between nodes can be detected and reproduced since it has all the information in plain text [9].

There are a few cases where the encrypted version has been found vulnerable. One vulnerability was found in the implementation of the firmware of a door lock device. This vulnerability could be exploited to change the network key and gain complete control of the device. The attack is made possible by a lack of state validation in the key exchange protocol handler in the door lock device. After analyzing the key exchange protocol handler, Fouladi and Ghanoun found that the handler fails to validate the shared key. This allows for an attacker to overwrite the key with his own by packet injection. If the key is overwritten, the attacker gains complete control and can perform unauthorized commands such as unlocking the door lock [9].

2.3 ZigBee 2012

ZigBee 2012 (ZigBee) is a wireless technology with a variety of implementation areas such as healthcare, retail services and home automation. ZigBee use the IEEE 802.15.4 standard. However, ZigBee extends the standard by adding a network and security layer as well as an application framework. By utilizing the application framework the ZigBee Alliance, who work to improve interoperability for the ZigBee technology, has created several standards for different fields of application. Furthermore, the application framework allows for independent

users to create their own standard when interoperability with other systems is redundant [10].

The 802.15.4 and ZigBee can be run in the 2.4GHz, 915Mhz and 868MHz frequency band where in the 2.4GHz band there exists 16 channels. The channel access method used is CSMA/CA. For wireless security ZigBee uses the standard AES-128. Moreover, ZigBee is set up as a mesh network and has a single-hop transmission range of up to hundreds of meters which allows for reliable connectivity. The ZigBee as well as the 802.15.4 technology try to be energy-efficient on account of 802.15.4 power management and power saving mechanisms included in ZigBee [10][11].

The ZigBee technology was designed to deliver a high level of security. However, the security is only valid as long as the implementation is valid. This is a widespread security issue that in this case has caused the technology to be vulnerable to an array of attacks. ZigBee devices use a network key that is hard copied to the device and loaded into memory on power up. If an attacker should gain physical access to a device he could extract the key and gain access to the network. Even if the devices themselves are physically secured, a hacker can still acquire encryption keys by mimicking a ZigBee device to capture and analyze traffic. The captured information can then be used to decrypt the communication. Replay attacks are also a threat to devices utilizing ZigBee as the session checking is lacking [12].

2.4 Internet Security

One has to be aware of the multitude of dangers that using an Internet connected device brings with it. The prospect of becoming a victim is increasing according to an Internet security threat report from Symantec. The report indicated, among other things, that web-based attacks were up by 23% from the year before, that 552 million identities were exposed via breaches and that 1 of 392 emails contained a phishing attack in 2013 [13].

The first step in securing a system is to be aware of the dangers. In general, both the public and private sector keep a high level of Internet security awareness. The private sector is often ahead in implementing security since they do not have the same resource constraint that actors within the public sector often experience. On the other hand educational institutions are often more aware of the latest security threats. Meanwhile, both sectors struggle with keeping its software up-to-date, especially that of its users. Fortunately, to handle new Internet threats many companies have installed the position of chief information officer (CIO). Their job is to keep the company up-to-date with the latest information technology security trends [14].

Furthermore, research has been made about different dimensions of information security awareness. The research concluded that security awareness among ordinary Internet users is crucial and that the public sector has an obligation to take the reins in this educational effort. They further state that the individual's need for knowledge about information security has become much greater as Internet usage has grown at a fast pace. There are also arguments that the rapid

expansion of information technology has caused people to want even more daily integration and only see the price-tag as an obstacle. This way, people ignore potential security threats as the security aspect is forgotten [15].

2.4.1 Internet of Things

A lot have been written about the security of the Internet of Things. We will expose ourselves to new risks when connecting everyday things to the Internet. This means that security and risk management will always be an important aspect when dealing with this technology. Pablos Holman states in his talk about wireless technology that when you turn your car, mobile phone, toaster or fridge into a computer it will inherit all the security properties and problems of a computer. These objects that control important things in your life, can then become pretty dangerous [21].

Home automation is a large appliance of the Internet of things. Home automation is susceptible to web-based attacks however, efforts have been made to find practical and secure solutions with hardware constraints taken into consideration. To make initially unsecure products safe, one can add extra security via a dedicated global server. The server will handle communications from remote clients and make sure integrity and authenticity are being upheld to guard against e.g. replay attacks. By adding external security, no upgrade of hardware is necessary as the needed processing power is outsourced [22].

2.4.2 Security concerns

Cross-site request forgery (CSRF) is an attack that is becoming more frequent and that poses a big threat to Internet users. We will later on try to implement a CSRF against the Tellstick Net system. The details of the attack can be read about in Section 3.2.2 and 5.1. CSRF have previously been implemented by others and has even been attempted against a financial institution where the attackers managed to transfer money from the target's account into the attackers' own account [18]. There already exists some standard protection against CSRF, such as cross-origin resource sharing. The existing protection is unfortunately lacking or in some cases not even implemented as it is dependent on which web browser the user has. However, research has been conducted to find new solutions to this problem. One proposition is to implement a proxy-based solution that acts as an external module. The proxy would be located on the server-side between a client and a web server. All outgoing communication from the client to the server would go through the proxy which would be able to determine if the request was a valid one or if a cross-site request forgery was attempted. The proxy would use tokens to determine the validity. This solution protects effectively against cross-site request forgeries and the implementation is independent of what type of application it is applied to [19]. There are others that suggest a client-side solution with a similar token-based approach that works as a local proxy server [20]. Even though these newer protections exists, many applications remain unprotected and thereby vulnerable. A likely

explanation is for developers and users alike to still be unaware of the dangers of CSRF.

Hostile hosts in the form of corrupted, publicly available computers is also a problem. A user can then be locally subjected to harmful attacks when accessing online services. For instance, a keylogger can be in place for an attacker to read or at least store the password of the user. Replay attacks that would replay connection information is also a concern. However, research has been made to come up with solutions to these problems. One suggestion is for all authentication to be handled by a separate web service. The authentication to the separate web service can be handled by an alternative method such as mobile phone authentication [17].

In order to diminish the number of attacks against a system that it connected to the Internet, it is very important to keep the software up-to-date. The Computer Emergency Response Team/Coordination Center has estimated that 95% of security breaches could have been prevented if up-to-date patches had been implemented. The Slammer worm is a standing example that illustrates the importance of software updates. In 2003 the worm was very harmful and caused problems to a lot of critical public services. However, 6 months prior to the appearance of the worm, Microsoft had released a patch that fixed the vulnerability that the worm exploited. If the affected systems had kept its software up-to-date a lot of damage could have been avoided [16].

3 Method: System Analysis and Risk Analysis

In this section we explain the method of the system analysis as well as the risk analysis with attack trees on the Tellstick Net system.

3.1 System Analysis

Here we present the method of the analysis of the Telldus system. All tests were made three times to be able to exclude any irrelevant traffic. To be able to see the data sent on the different communication channels we used a man in the middle consisting of a laptop equipped with a *USB to Ethernet adapter*. The adapter allowed us to bridge two of the computers network interfaces in order to forward traffic. In all the tests we used *Wireshark* to *sniff* and analyze the traffic.

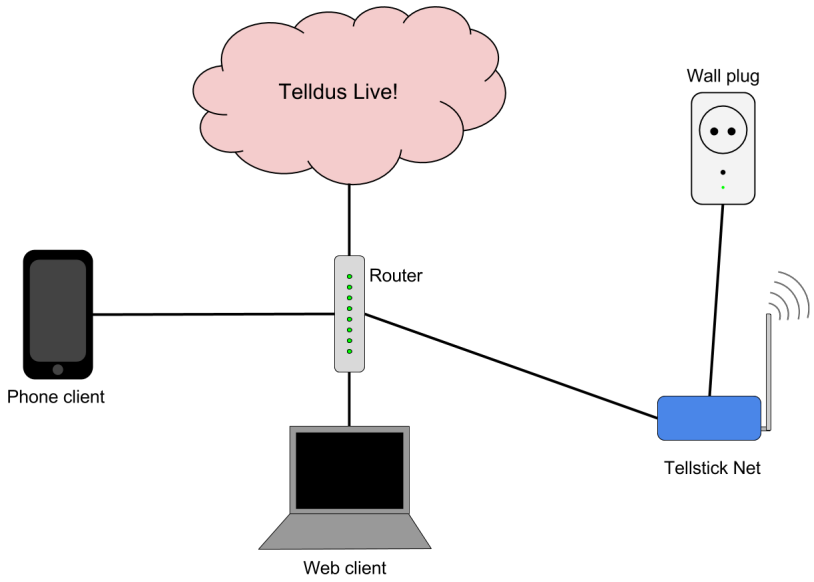

Figure 2: The setup we used when analyzing the system

To be able to sniff the traffic to and from the *web client* , as seen in Figure 2, we began a Wireshark session on the same computer as we were running the web client on. We also retained the IP address of the *Telldus Live!* server and filtered the trace based on this IP address.

When tracing the traffic to and from a phone client we set up a computer to share its ethernet connected Internet connection via *WiFi*. This enabled us to be a man in the middle as seen in Figure 3. We then connected the phone to the computers WiFi connection. With this setup we began a Wireshark session listening to the WiFi interface and filtering on the traffic to and from the Telldus Live! server.

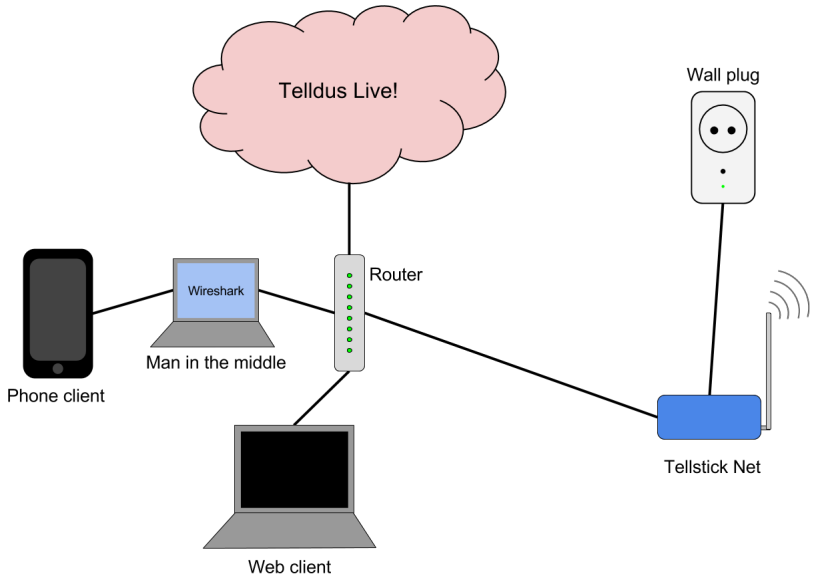

Figure 3: Our man in the middle setup to be able analyze traffic to and from the iPhone

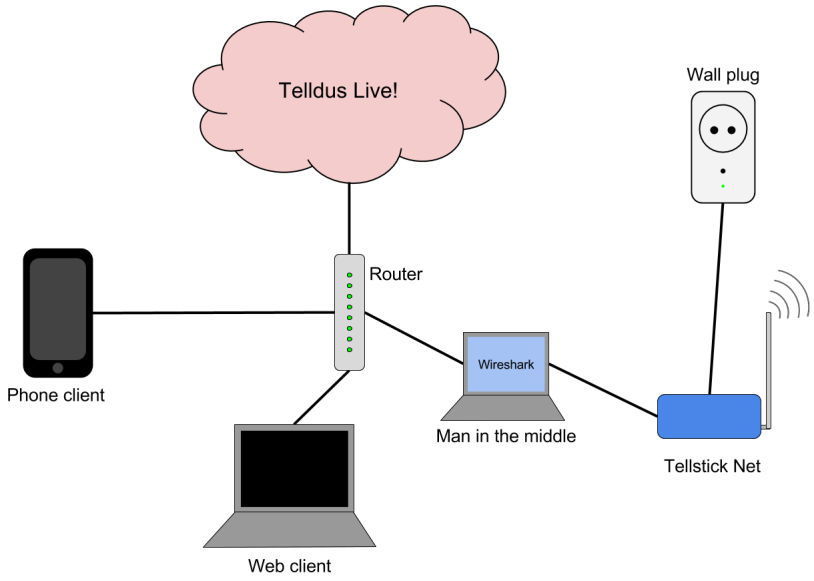

Figure 4: Our man in the middle setup to be able to analyze traffic to and from the Tellstick Net

Lastly, we managed to trace the traffic to and from the Tellstick Net device using the adapter to forward the man in the middle computer's Internet connection via the original ethernet network interface. This setup can be seen in Figure 4. Once again we filtered the traffic to only retain the relevant packets.

3.2 Risk Analysis

The method of the risk analysis made with attack trees is presented in this section. This section also contains information regarding the different attacks that were considered during the analysis.

3.2.1 Attack Trees

The risk analysis was done using a method called attack trees. The attack tree method is a form of risk analysis focusing on different attack angles for a given objective. The end-goal, seen from an attacker's point of view, works as the starting point of the analysis. The method consists of building a tree with the end-goal as the root node. The children of the root node contain different ways of achieving its content. In turn, each child of the root node can have children of its own. In a similar manner as with the children of the root node, the content of each child in the tree describes a way of achieving the content of its parent. Furthermore, a node can have different kinds of nodes as children, AND- and OR-nodes. When a node only has one or more AND-children it implies that the content of the node only can be attained by achieving all of its AND-children. AND-nodes can be seen in Figure 5. On the other hand, when a node has one or more OR-children it can be achieved if any of its OR-children can be achieved, which is also seen in Figure 5 [23].

We performed a brainstorming session in order to come up with root nodes i.e. attack goals for the attack trees. Thereafter each root node was expanded and broken down into branches, representing ways of achieving the goal. This was once again performed by using brainstorming sessions. Each leaf of each tree was further analyzed to potentially be broken down into its own branches. The breakdown was continued until each leaf of each tree was specific enough for it to be evaluated in regards to cost of performing the attack and probability. All this was done in a span of several days as is described as good practice in Schneier's paper on attack trees [23].

When building the tree, one approach is to expand the tree downwards until each leaf can be evaluated sufficiently in regards to probability, i.e. if the node is considered possible or impossible to execute. However, there are a number of labels you can choose to bestow on each node, instead of branding them as possible or impossible, e.g. if an attack requires a low or high skill level or how expensive an attack is to perform. This will allow for complex and highly customizable analyses using a tree that includes different kinds of labels [23].

When each tree was fully developed the leaves were evaluated and values for cost and likelihood of the content of each node was set. We determined the values by utilizing planning poker, a method for estimating values, typically

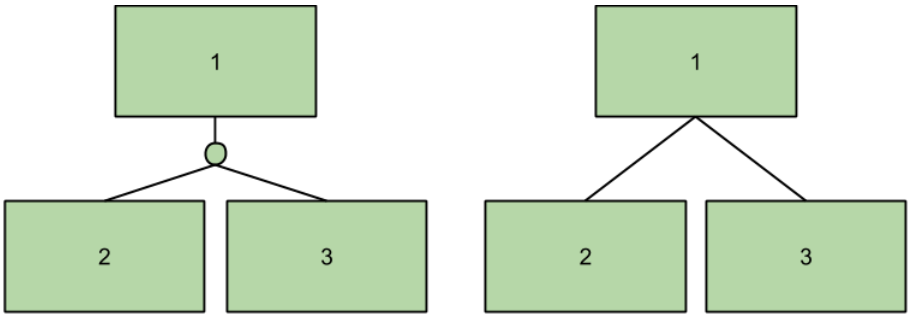

Figure 5: Two types of nodes in an attack tree, to the left AND-nodes and to the right OR-nodes

used in software development. When determining the cost of each node we worked under the following presumptions:

- The attacker has a computer available for development.
- The attacker will have to buy a new computer or server if one computer is not enough or if the device will have to run for the sole purpose of the attack.
- The amount of work and time put into the attack is not included in the cost evaluation.

In some cases the same attacks can be used to achieve several attack goals, e.g. gaining access to login information is included in all four attack trees. To not replicate large branches we used referencing within the attack trees. If a node was evaluated to be either possible or impossible, a "P" or "I" was set respectively. The cost of a implementing a leaf node is represented by a number. The cost scale is presented in Table 1.

Scale	Cost (SEK)
1	<100
2	100-1000
3	1001-5000
4	5001-20000
5	>20000

Table 1: Cost scales used in the attack trees

3.2.2 Attack Vectors

These attacks were considered when we evaluated an attack goal that acted as root node for an attack tree.

Man in The Middle

Man in the middle is performed when a third party intercepts the communication between two others, effectively sniffing every packet being sent in both directions. The sender and receiver is most likely unaware of the eavesdropper unless the attacker decides to modify or in other ways alter the communication between the targets. Even then, the attacker might not be detected. There are many different kinds of man in the middle attacks which differ a lot and each of them have their own strengths and weaknesses [24].

Cafe Crack

Cafe Crack is a type of man in the middle attack which trick people into connecting to a rogue access point when there is supposed to be a trusted one in the area. This is achieved by mimicking and optionally shutting down the original access point and letting people connect to the rogue access point by their own accord. The position as AP can then easily be used to further trick targets into giving away sensitive information or in other ways cause them harm [25].

Users can protect themselves against man in the middle attacks made through these rogue access points and other similar attacks as well. Many of the measures that can be taken will not completely protect the user but will make it much harder for an attacker to succeed. Even if a user wants to connect to an unsecure access point the user can make sure that the AP do not have control of the DNS and instead use a third-party DNS server. This will prohibit the attacker from producing a phishing site with the same name as the original domain. However, the user will have to watch out for phishing sites with similar names as the original. Another way to protect against this kind of man in the middle attack is to make sure that whenever possible, secure protocols such as DNSSEC and HTTPS are being used [25].

ARP Spoofing

ARP spoofing is done by spoofing a local area network by sending fake ARP messages in order to make it believe the MAC address of the attacker correspond to the MAC address of another host inside the network. This will cause all traffic meant for the spoofed host to be sent to the attacker instead and traffic sent from the attacker will be perceived as that of another (trusted) host. It is possible because of the lack of authentication in the ARP protocol. [26].

It is hard to protect against ARP spoofing since the ARP protocol is very naive when storing MAC addresses in caches as all ARP packets are trusted. There are other protocols such as S-ARP that would solve the problem, however they are not practical to implement. One suggested solution is to actively probe for inconsistencies in the network [26]

DNS Spoofing

DNS spoofing is used by attackers to retrieve personal information by redirecting the target to a site controlled by the attacker. When the user types in a domain name in their browser it will check with the local DNS server to see which IP address that is connected to the domain name. DNS spoofing means returning the IP address of another website [27].

DNS spoofing is achieved when a man in the middle act as the name server or when the cache of the name server being used has been compromised. Defending against the attack is therefore done by the same methods as described in the cafe crack and DNS cache poisoning attacks.

DNS Cache Poisoning

DNS servers use caches to supply users with correct IP addresses to ensure good quality of service. However, these caches can be targeted for attacks which cause them to store a false address to one or more websites. Since these caches are used to supply a large number of users, poisoning of the cache can cause a lot of people to be affected. There are a number of ways to execute a *DNS poisoning attack*. One way is to brute force a name server by sending a large number of DNS queries as well as fake DNS replies until all the needed parameters match and the name server accept a fake reply. [28].

As with many other attacks, there is no absolute protection against a DNS cache poisoning. There are however several actions to reduce the risks such as the use of DNSSEC and BIND 9.x [29]

Replay Attack

A *replay attack* is executed by the attacker intercepting a message sent between a sender and a receiver, where the receiver will be the target of the attack. The message is later replicated and in some cases altered and then sent to the destination of the original message. If the service requires the user to be logged in, the attacker can exploit the time-window in which the login session is active or record the actual login conversation and replay that as well. The attacker does not need to decrypt the message to replicate it and thereby the encryption can be completely surpassed [30].

However, there are several counter measures to be taken against replay attacks. One of these involve including a timestamp in each transmission which will allow the receiver to verify that the message was sent within a certain time frame and thereby not intercepted, stored and transmitted at a later date. The timestamp can be included in a hash of the message to ensure no one tampered with the timestamp in an attempt to circumvent it. Further, there are other similar countermeasures involving for instance sequence numbers [30].

Cross-Site Request Forgery

Cross-site request forgery is an attack used to gain access to something secured by a login. The idea is that while the user is logged in to a secure service, make the user visit a web page where you have code that will make requests to the secure service. Since the web browser has the session stored in a cookie it will be included in the requests from the second site as well. This type of attack is also called session hijacking and abuse the trust a web server has in a user [31].

A lot of web browsers have built-in protection against cross-site request forgeries thanks to implementing either a same-origin policy or a mechanism called cross-origin request sharing. The same-origin policy means that a website only accept requests made from itself. I.e. a website will not accept a request made from another website. Most modern web browsers use cross-origin request sharing. This will allow requests to be made cross-origin, however a new field is added in the header that lets a website know where the request came from and thereby make an informed decision if the request should be allowed [32].

Trojan

When a user installs a program on their computer a *trojan* can be included and installed along with the intended program or the trojan can be disguised as the intended program and all together fool the user into installing the malware of their own accord. The nature of a trojan can differ a lot, however, the common denominator is that it will contain malware that in some way cause harm to the user. In some cases the trojan will even supply other malwares with a back door to breach the target computer [33].

Protecting against a trojan is not easily achieved however, there are some precautionary actions that can be taken. To prevent a trojan from finding an entrance into a computer system, one can step up a firewall in order to block access to hosts outside of the network. If a system already has been infected, antivirus programs can be used to detect and delete the trojan. However, most antivirus programs will rely on recognizing known trojans. This means that new or unknown trojans may go undetected. Trojans typically need to be granted access to enter a system which is why many trojans are disguised as other softwares. Consequently, not downloading suspicious or unknown programs or content goes a long way in preventing trojans from entering the system [34].

Keystroke Logger

A keystroke logger, also known as a *keylogger*, is a malware designed to record user interaction on the targeted computer. This can be achieved by logging different mediums such as the keyboard or by simply recording the video output stream. A keylogger is typically used to retrieve sensitive and valuable information such as passwords. Often the keylogger is injected into the target system by a trojan [35].

A firewall will help with suppressing a keylogger as most firewalls will detect any data being sent out to the Internet through an unauthorized software. This

will allow the user to find and remove any suspicious softwares. However, a firewall is not 100% effective and even monitoring outgoing traffic through a software such as Wireshark can be difficult as keyloggers that log keyboard strokes will send out such a small amount of data. A user can however protect sensitive information by utilizing other softwares that save their passwords etc and autofill when required. This way no keystrokes have been made and the keylogger can not detect the password except when it was used for the first time. This will not surpass the risk of a keylogger but will minimize it [36].

Denial of Service

Denial of service is an attack used to hinder clients from reaching a service by bringing down its servers. This is often done by a hostile client overwhelming a server with requests, providing the server with so much data to be processed that it can not keep up. This will cause long response times towards all users, rendering time sensitive services useless, and in some cases the server can be forced to shut down altogether. This attack can be devastating for a service that rely a lot on its servers being up and running at all times [37]. *Distributed denial of service* (DDoS) is a type of DoS which is used to bring down larger services and larger servers. The difference from DoS is that in this case there are many clients attacking, possibly a botnet (corrupt clients controlled by a single attacker e.g. via a trojan). Historically this has been used to bring down government websites, online newspapers and online games [38].

Even though a DOS attack is generally hard to prevent there are methods that can be used to thwart an ongoing attack. To thwart the attack, the administrator can set up a predetermined request per time-unit threshold. If a client exceeds this threshold, actions can be taken to either restrict or block the offender. There are however ways to get around this prevention method, as a blocked client can change its origin and reconnect to the server which also can result in the server having to process a lot of requests [39].

4 Results: System Analysis and Risk Analysis

4.1 System Analysis

In the following sections we explain the different actions that were taken for us to be able to analyze the system and what was discovered during the tests. During each event or action we monitored each network connection described in Section 3.1.

4.1.1 On/Off Request

Using both a phone application and the web client we turned on a connected device, waited a few seconds and then turned it off again. Requests made from the phone were encrypted using HTTPS which meant that we could not read which type of requests that were made. The requests from the client to the server however were not encrypted. The traffic looked very similar for *on* and *off* requests. They both contained an id and a mode *URL parameter*. The id identified which unit the command was meant for and the mode parameter was used to define if the request was for turning on or off the device. The requests were answered by the server with an "HTTP 200 OK" containing "true". The response contained a TCP segment with set PSH and ACK flags. If the Telldus Net was offline the HTTP 200 OK would change flags from PSH and ACK to FIN and ACK.

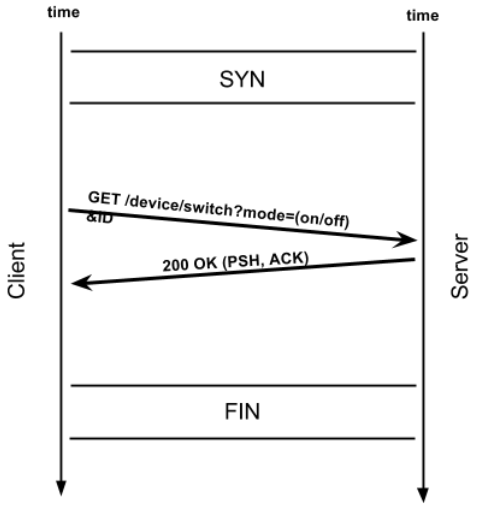

Figure 6: The communication between the web client and the server when turning a device on or off

The communication between the Tellstick Net and the server was analyzed when both a web client and when a phone application was used to trigger a

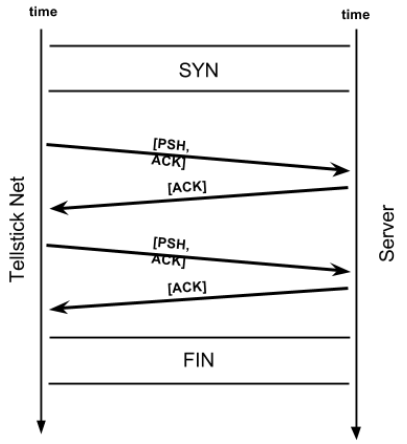

Figure 7: The communication between the Tellstick and the server when turning a device on or off

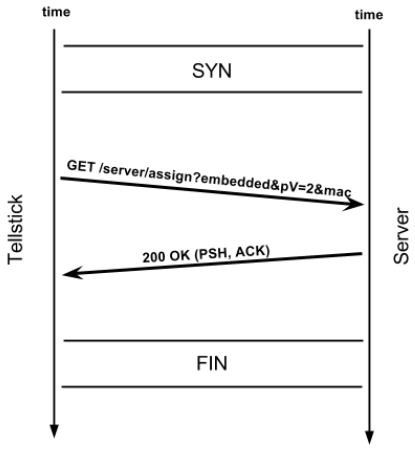

Figure 8: The communication between Tellstick Net and the Telldus Live! server during setup of the Tellstick Net

connected receiver. There was only one difference detected in the Tellstick Net when the client sent a request in contrast to when a phone application sent one. The request from the client included data that was sent in the application layer as application data, while the request from the phone sent its data in the transport layer frame as TCP segment data. Each on or off request produced four TCP packets. It also contained the usual TCP setup and tear down sequence. TCP packets that included a PSH and ACK flag also included application data. This communication can be observed in Figure 7.

4.1.2 Learn Request

Before a receiver could be used it needed to be incorporated into the Tellstick Network. This was accomplished by a learning mechanism. A button on the receiver was pushed which allowed a Tellstick Net device to connect to the receiver. The connection was initialized when a learn request, from either a phone application or web client, was made.

We enabled the *learning mode* on a receiver and, on separate occasions, pressed the learn button in the phone application and web client. The learn request produced a GET request similar to when an on or off request were sent. The mode parameter was however, set to *learn*. The communication between the web client and the server was not encrypted however, the communication between the phone and the server was encrypted with HTTPS. At the Tellstick Net device the communication was similar to that of an on or off request that can be seen in Figure 7.

4.1.3 Login Request

We signed out of the Telldus account and made sure to restart the application as well as the browser before logging in again on the phone application and web client. Before logging in we started our Wireshark session on the computer which was acting as a man in the middle. The communication sent during a login attempt was encrypted using HTTPS and could therefore not be read.

4.1.4 Tellstick Net Setup

We unplugged the Tellstick Net's ethernet cable which made the Tellstick go offline and a light on it turned red. Then we set up our Wireshark session and reconnected the ethernet cable. When the Tellstick Net came online it made a DNS request for api.telldus.com and started communicating with api.telldus.com. Using a GET request the Tellstick device asked the API server which server to use for further communication. The API server then assigned the Tellstick a server <*female name*>*.telldus.com*, this was accomplished with an HTTP 200 OK response containing an address to the assigned server (e.g. ebba.telldus.com). This can be observed in Figure 8. The Tellstick Net then established communication with the assigned server. This communication was held open and waiting for tasks (e.g on/off requests).

4.1.5 Tellstick Net Installation

After connecting the Tellstick Net to the Internet we logged in to Telldus Live! in order to install the device and connect it to our account. Using the auto discover feature on Telldus Live! we could find and connect the Tellstick Net. When the Tellstick Net device was being set up there were a lot of communication going on between the client and the server. The client seemed to request a list of available devices and after choosing a device the client would ask the server to activate and register it. This development can be seen in Figure 9. When the client sent a GET device/index?openid.assoc_handle=<id> request, one can read in plain text that HMAC and SHA-1 were being used. In the POST /register/acceptClient request one can see that an id was sent which corresponded to the id seen in the GET /activate/client?=id=<id> request.

There were less communication taking place at the Tellstick. When the Tellstick had reconnected to the Internet, the device sent out a DNS request to retrieve the address to *api.telldus.com*. After the communication had been established the device sent a "GET /server/assign" request and the server then replied with an HTTP 200 OK which included the address to the server as an added field, which can be seen in Figure 10.

4.1.6 Idle

To isolate the behavior of the Tellstick Net when no requests were made we started a new Wireshark session and recorded the behavior while doing nothing. When the Tellstick did not receive any requests from the server it still kept the

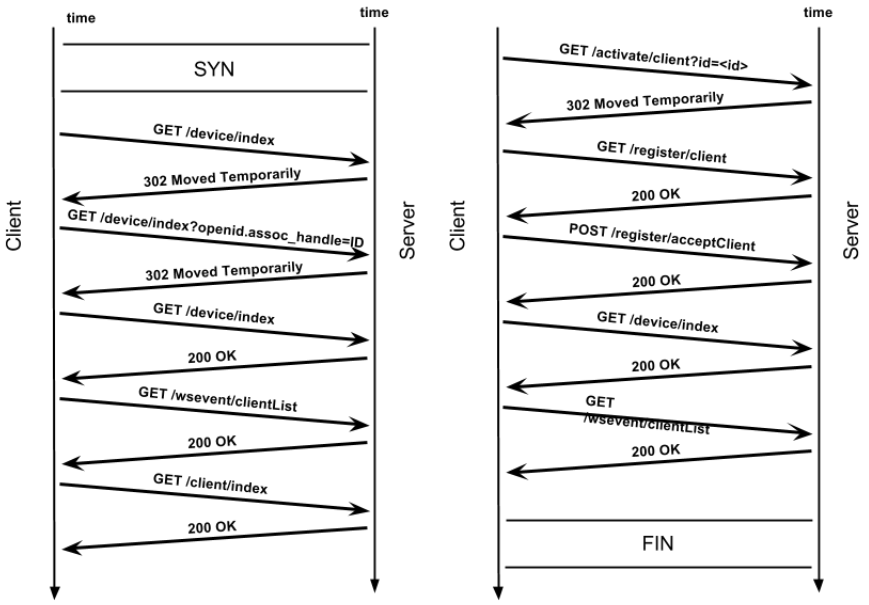

Figure 9: The communication between the server and the Web Client when a Tellstick is first installed

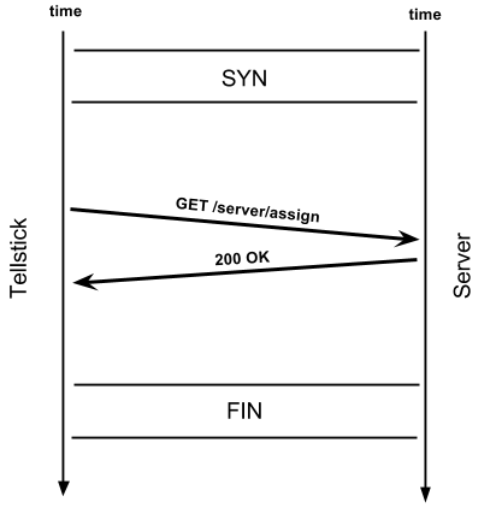

Figure 10: The communication between the server and the Tellstick when the Tellstick is first installed

connection by sending out *TCP keep-alive segments* to the server that contained data with a length of 1. The server then answered with a TCP keep-alive ACK segment. Two of these segments were sent out from the Tellstick Net every 10:th second. Further, the Tellstick Net sent a TCP segment with PSH and ACK flags set at seemingly random times that contained some data. These segments were answered by the server with a TCP ACK segment.

4.2 Risk Analysis

In the following sections the attack trees are presented and the nodes which were hard to determine are discussed.

4.2.1 Control Receiver

In Figure 11 and 12 you can see the attack tree for controlling a receiver. The tree was divided in two for the purpose of its presentation. The node *Control receiver by resetting and taking over it by sending reset request with 433.92Mhz signal* was deemed to be impossible as the reset is made by pushing a button on the receiver itself and not by radio signals. The node *Control receiver by resetting and taking over it by pretending to be the owner of it, contacting support and letting them remotely reset it* was deemed to be highly unlikely. However, during our discussions it was not completely discarded and we agreed it was a possibility given specific circumstances.

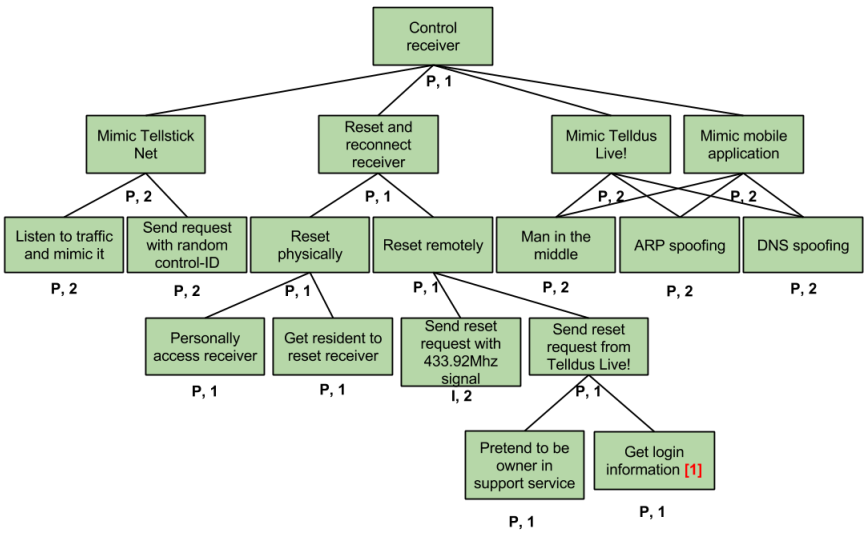

Figure 11: The first part of the attack tree with the goal to control a receiver

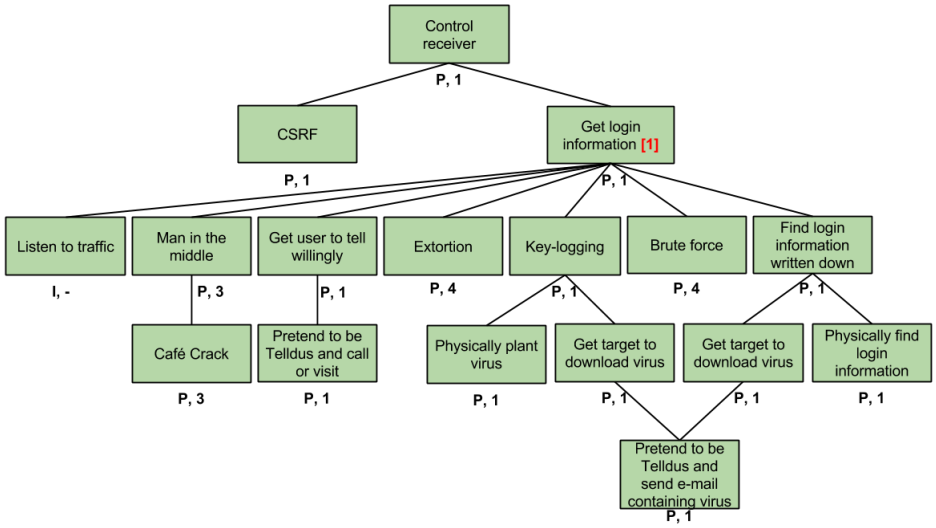

Figure 12: The second part of the attack tree with the goal to control a receiver

4.2.2 Block Access

In Figure 13 we present the attack tree for blocking access. The node *Block access by stopping receivers from responding to requests* is possible to some extent as the attacker simply could jam signals and thereby disrupt the flow of information.

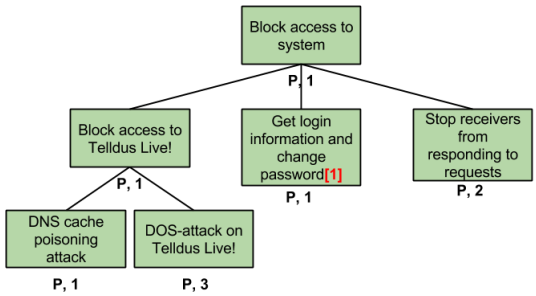

Figure 13: Attack tree with the goal to block access to the system

4.2.3 Monitor User

In Figure 14 we present the attack tree for monitor a user. When considering *Monitor user by implementing a logger virus in the Tellstick Net device* we determined that, even though we are not sure of the how to, this could be achieved and thereby we set the node as possible.

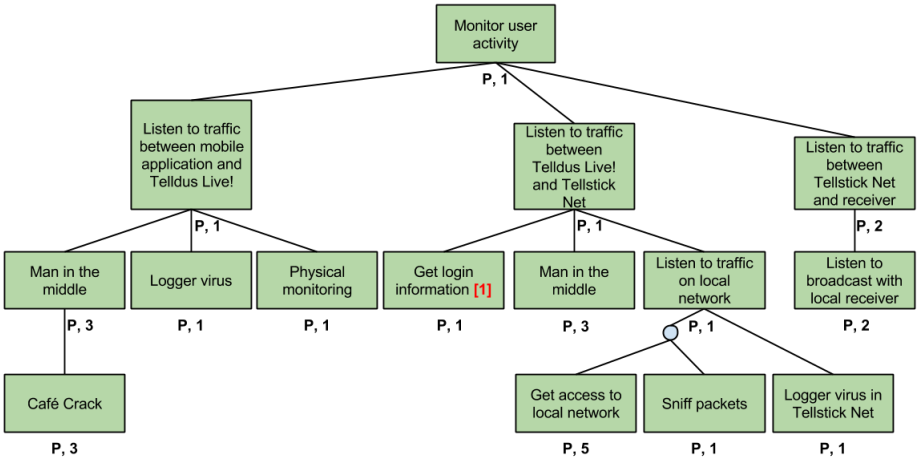

Figure 14: Attack tree with the goal to monitor a user using the system

5 Implementation of Attacks

After successfully completing our system analysis as well as the attack trees, we found the attacks most suited for our project based on required skill level, cost to perform and gain to the attacker. We found that cross-site request forgery and replay attack were the most suited because both of them are cheap to perform and requires little preparation while they still can cause damage to the targeted users. The method for performing these attacks are explained here as well as the results of the attacks.

5.1 Cross-Site Request Forgery

During the system analysis we discovered that the receivers were controlled by sending a GET request from the client to a specific URL. The request contains two URL parameters, one is called *id* and the other is called *mode*. The id parameter is the identifier of the device and the mode is the command sent to the device. Using this knowledge we could write a script that found the connected receivers and then control these devices. The script we used can be seen in Appendix A. The script was then added to a simple Hello, World! HTML page. The HTML page can be found in Appendix B.

To test the attack we used *Google Chrome* and *Internet Explorer 7 and 9*. We signed in to our account at Telldus Live! in one tab and in another tab we opened our Hello, World! page. After analyzing and trying to perform the CSRF we discovered that we needed to disable web security in order for the attack to succeed as described in Section 3.2.2. Therefore, the attack was done with and without the flag *–disable-web-security* in the browser.

When executing the attack with security enabled the response to the request was a "HTTP/1.1 302 Moved Temporarily" with a redirect to the login page. We discovered that both Google Chrome and Iternet Explorer 9 adds an *origin* field when a request is done cross origin. This means that even though the browser has an active session and thereby can make requests to the Telldus Live! server, the server will not accept a request from another origin like our Hello, World! page. But when you turn off the browser security the server begins to accept the request made by the script. The reason for this behavior is that the browser does not add the origin header field when security is turned off and then the Telldus Live! server assumes the request is a same origin request and accepts them. When browser security was disabled the attack performed as intended. The script was able to find all the connected devices and then control them. Internet Explorer 7 uses a same origin policy with a referer header which makes the results the same as with Google Chrome and Internet Explorer 9.

5.2 Replay Attack

A replay attack was performed, aimed at the Tellstick Net device with the goal to turn the receiver on. During the system analysis we discovered that when a request to turn a receiver on is sent, four TCP packets are sent between the

Tellstick Net and the server, see Figure 7. As described in the system analysis, the first packet contain application data as well as a PSH and ACK flag which suggest the packet contains the command to turn the receiver on. This packet is followed by an ACK packet and another packet, similar to the one sent from the server. Finally, the two packets from the Tellstick is answered with an ACK from the server. A man in the middle, that utilized two ethernet ports in the attack host, was set up as seen in Figure 4 and described in Section 3.1. The packets that passed through the attack host were recorded with Wireshark. In order to replicate the traffic described in the previous paragraph we replayed the first packet from our position as man in the middle. We managed to send the packets by using a packet infusion program called *Bittwist*. Bittwist is used to capture data from .pcap files and send the content into a selected network interface. We produced a list of available network interfaces by entering "bittwist -d" into the Windows Command Prompt. The interface used towards the Tellstick Net was noted for later use. The PSH packet from the server was singled out in Wireshark and exported into its own .pcap file. Bittwist was then used to infuse the .pcap file into the previously noted network interface, by entering "bittwist -i <interface> <.pcap file>" into the Command Prompt. Then we waited for the Tellstick to reply with the expected ACK packet as well as the PSH packet containing data, to then send the final ACK packet that is expected from the server.

When we infused the first TCP packet of the four-packet conversation, we could see the conversation taking place by recording all packets with Wireshark. We could then see that the Tellstick received the infused packet and answered the server with a TCP packet. This packet contained information saying that the TCP segment ACKed an unseen segment. Therefore, the ACK from the Tellstick device was not followed by the expected PSH packet and no other packets were sent from either the Tellstick or the server. As described in Section 4.1.6, the Tellstick Net keeps the communication open which allow the server to send an ACK packet instead of having to initiate the communication when a new action is requested. This means that the server will not initiate the communication and instead it has to respond to a packet sent by the Tellstick Net. Meanwhile, the infused packet is an ACK of a packet that was sent from the Tellstick Net before the packet was originally captured. When the packet was infused it ACKed an old packet instead of the packet that was most recently sent to the server.

6 Discussion

In this section we will discuss our method and our results. We start by looking at the system and risk analysis methods as well as the methods of the two implemented attacks. We then move on to discussing the results from the system and risk analysis and the attacks.

6.1 Method

6.1.1 System Analysis

We chose to repeat every test three times. In almost every test the first test would have been enough. Since we found a few differences in a couple of tests the tests had to be done more than once. We can not be sure that our observations of the system behavior are correct but due to the small differences between each test we think three times is sufficient.

Since the analysis of the system can be divided in smaller and smaller pieces this work could go on for a long time. The work done in this report has given us some understanding of how the system works and how it can be abused to control and monitor. The structure of the presentation of the used method is divided in first how we setup our man in the middle to be able to see the traffic in the system and second how we used this setup to analyze the system. Since we only used one instance of the system in the analysis we can not be 100% sure that the system works the same way in all cases.

6.1.2 Risk Analysis

The attack trees gives an overview of what can be done to abuse the Tellstick Net system and by which means. It shows us what is worth doing and what is possible for us to do. By breaking down the goals we were able to expose weaknesses and together with the system analysis, find suitable attacks for this system. Something that is important to remember when reading the attack trees is that the cost for the attacks are our own estimates and are not supposed to be seen as a reference since they not have been verified.

6.1.3 Cross-Site Request Forgery

This attack was performed logged in to our own account at Telldus Live! and the attack site was run on a local web server. With this in mind you need to be able to distribute this attack to potential users and also since the browser security must be disabled in modern browsers this attack would have to be expanded to work in a real life situation. The results of the method are replicable as long as security protocols remains the same as they are today. However, there are of course no guarantees that they will be, as new security protocols are constantly being developed and extra security against CSRF could become a new Internet standard.

Our implementation shows that the system rely on a security feature in the browser. If the feature is non existing the system assumes everything is in order which this report has proven it should not be doing. Therefore the server should rely on security mechanisms that it can trust and control.

6.1.4 Replay Attack

The attack was simple in the sense that it did not try to counter any security measures that the targeted system might have. To set up a man in the middle as described in Section 3.1 was easy as we had complete access to the network. The software needed for the replay attack are open-source and the devices needed for our man in the middle setup are cheap. Just like CSRF this attack would need to be expanded in order to work in a real life situation. That is because the attacker is unlikely to have complete, physical access to the targeted network. Therefore the attacker would need to find a way to remotely implement a man in the middle before continuing with the attack.

6.2 Results

6.2.1 System Analysis

The results from the system analysis were not only essential when deciding which attacks we were going to test but it also gave us a good overview of what we were working with. To clearly see all the sequences and how different parts of the system communicate, made it possible to find where the system was most vulnerable. Looking at the encrypted communication between phone application and server showed us early that targeting the non encrypted communication between client computer and server would probably be more successful. Since all communication goes through the online server, we realized that a man in the middle attack could be very effective. We also realized that the unencrypted HTTP requests, sent from the server, could be exploited.

6.2.2 Risk Analysis

We chose to use attack trees since it is a compact method used to analyze a system like the Telldus Net. It was a good way to structure the potential risks and how they related to each other. The attack trees demonstrate that many attacks can potentially be made against a Telldus Net system. As has been described in Section 2.4.1, that is because all devices that are connected to the Internet become vulnerable to the same attacks as an ordinary computer. By utilizing the knowledge gained from the system analysis we were able to come up with an array of attacks. We were also able to acknowledge that these attacks could be made against different parts of the system which led to a deeper analysis. After researching different attacks and creating the attack trees we were able to decide which of the attacks that were more beneficial to us. Cross-site request forgery seemed especially prominent when regarding

our findings during the system analysis. The replay attack also seemed like an effective attack against the Tellstick Net system.

6.2.3 Cross-Site Request Forgery

The attack is not so powerful by itself as it requires certain conditions to be true. The behavior with cross-origin request sharing (CORS) and the *origin field* tells us that the server rely on security functions implemented in modern browsers. At the website www.caniuse.com you can see that all modern browsers implements CORS which is responisble for adding the origin header. The use of browsers that implement CORS is more than 95% [40]. Older browsers such as Internet Explorer 7 instead use the same origin policy and referer field that can be exploited. Removing the referer is one way to circumvent it as many websites will accept requests without it. Other websites will not even check the header. More importantly, referer headers can not be sent when the request is issued from a website using HTTPS to another website that does not use HTTPS [41]. In our case, if the request had been sent from an HTTPS website the referer header would have been dropped and the request would have been accepted.

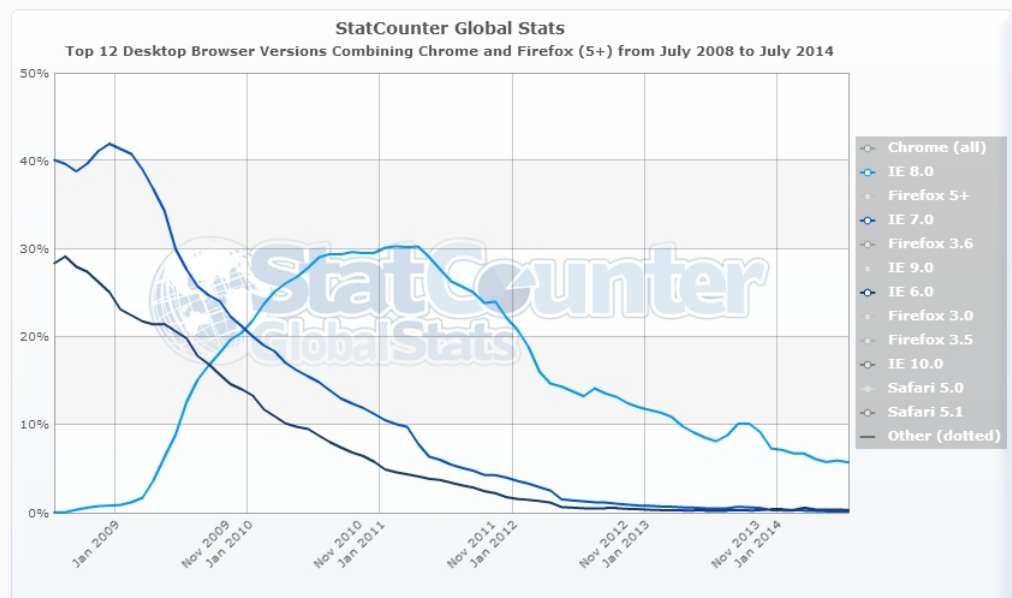

Figure 15: Market shares of Internet Explorer 6,7 and 8 from July 2008 to July 2014. Source: gs.statcounter.com

However, this means that only some of the users on the Internet are vulnerable to this type of attack. But the number of people who use the Internet cause the probability of the deployment of a successful CSRF to become substantial over time. This contributes to making CSRF an actual threat against home

automation technologies. Since the security against this threat is on the client-side, the service can not know if its users are protected or not. This is a common problem with client-side security when a service relies on another party. In this case Telldus Live! is secure as long as the browser is secure and up-to-date. Thereby the service makes the browser responsible for security. Although a system is secure on the server-side it can overall be vulnerable as a system only is as secure as its weakest link. Unsecure web browsers are widely used even after their unsafety have been made publicly known and newer versions have been released. CSRF has been around since before year 2000 [42] and most of the web browsers that lack CORS are no longer being used. However, considering that older web browsers without CORS was widely used long after CSRF had made its appearance means that a lot of hosts have been vulnerable to the attack and still are because of the slow pace in which older versions of web browsers are phased out. This can be seen in Figure 15. For instance, Internet Explorer 6 which does not have CORS, peaked in market shares in 2002 and 2003 [43].

Furthermore, the security threat that CSRF poses is also described in Section 2.4 where even a financial institution has been subjected to a successful attack. The existing security standards that protect users against CSRF is not enough as HTTP headers can be manipulated or left out and extra security measures should be implemented. Consequently, we believe that using up-to-date software is key in defending against attackers. Keeping software updated is important to hinder attackers from finding weaknesses in the system that could be turned into threats, as has been described in Section 2.4.

6.2.4 Replay Attack

The Tellstick Net and the server keep up a constant flow of packets by sending ACKs. Before we examined the result of the replay attack, we did not realize the impact that this behavior would have. As seen in the results, the ACKs caused the replay attack to fail. However, this behavior serves a purpose as it allows the Tellstick and server to communicate without configuring the firewall at the Tellstick side of the network to allow access to the server. In Section 3.2.2, it is described how a replay attack can be countered by implementing a timestamp or sequence number. The constant stream of TCP packets with ACK flags set, effectively work as sequence numbers which shuts down the attack.

In the case of this system, a replay attack was not possible to perform between the server and the controlling unit because of the way the system utilizes the ACK flag. If a system instead would make the user allow the server through the firewall a similar replay attack as was tested here might work. In that case, the communication between the server and the device would need to be secured by some kind of encryption which would require more advanced software in the controlling device. Without security measures, the system would then be vulnerable and a replay attack could then pose as a real threat.

6.2.5 Other Feasible Attacks

The attacks described in this section were deemed prominent after the system and risk analysis but unfit for implementation considering some constraints of this project or the illegality of the attack.

Denial of Service Attack

We recognized a DoS attack to be very prominent. Tellstick Net is server based and rely 100% on the server being up. Since all traffic is going through the server regardless of the client's position, bringing down the server would cause none to be able to access their devices. Performing a DoS attack on Telldus Live! would possibly hurt the credibility of the company which in turn could result in financial losses. As the company Telldus grows they will be more and more vulnerable to a DoS attack and there is no way to entirely protect against such an attack. One recourse is, of course, to avoid using the servers and possibly introduce an offline mode. However, this would in turn mean losing some of the advantages with having an all online service.

Local 433.92MHz Attack

The short range wireless communication between the controlling unit and other connected receivers in a smart home network, lacking security, can easily and above all cheaply be exploited. A small computer with an attached *433.92MHz* antenna will suffice to listen to all traffic sent in the area as well as replicating traffic to cause unwanted behavior in nearby receivers. The required configuration would not be very complex either. The attacker would need basic technological knowledge as well as some programming skills. When the device is set up there is only a matter of deploying the device in range of the target of the attack.

In this report we chose not to perform an attack against the communication between a receiver and the Tellstick Net as we knew the communication between the two is not encrypted and therefore, further investigation and exploitation in that area would not reveal something we did not already know. However, this should not be interpreted as if the short range, 433.92Mhz, communication between a receiver and a Tellstick Net is not a major security risk. A simple replay attack would suffice to control an entire home network. Of course, the attacker will need to be physically close to the network which limits the attack. Without proper security against a replay attack against the 433.92Mhz communication, e.g. timestamps, security measures taken in other parts of the system will be next to obsolete given a determined attacker.

We can assume no ordinary person, using the technology to control non-sensitive electronics e.g. lights, is at risk of being subjected to these kinds of attacks. On the other hand, this kind of technology is relatively new and with time people will expand and find other applications which will most likely become of increasingly sensitive nature while not ensuring they use a product with sufficient security for their needs. In some cases, this is certainly a reality

even today. With increasing public interest and rise in popularity this might even turn into commonplace considering competition to gain market shares is bound to increase. This will undoubtedly cause cheaper alternatives to the now relatively secure product on the market to rise at the expense of their security.

7 Broader Perspective

The Internet of things have made us much more vulnerable to certain types of violations. It is easier to spy on you as a person, finding patterns in your behavior and subconsciously controlling you, thus violating your integrity. Smart home systems have over all made the customer more exposed to risk concerning personal integrity, since monitoring can be done in a larger scale and it is easier to track and predict your actions. This can be seen out of two perspectives, both as a security measure to prevent terrorism and making it harder for people to commit crimes but also, as it is possible to track your location, sensitive information is at risk of being exposed to people who can use it to cause harm to others.

8 Conclusion

The idea of this project was to determine how vulnerable systems and devices we let into our everyday life actually are. The primary purpose of the report became to "establish the necessity for well thought-out security policies and mechanisms when it comes to design of smart technologies for the home". This is something we have been aiming for during the entire project by searching for and testing attacks that might be easy and cheap to do against smart home systems. If the attacks are easy and cheap to perform their threat level increases as they become available to more people and in turn will be executed more frequently. Our attempts made by implementing a CSRF and a replay attack were not very complex. The CSRF required the target of the attack to have disabled browser security which is not something a lot of people browsing the Internet have. However, inevitably there are people who do, simply because of the sheer number of people who browse the Internet and consequently the attack poses a threat nevertheless. Furthermore, some users could be tricked into disabling their browser security themselves. CSRF also constitute a larger threat against those who are using older web browsers that lack CORS. A replay attack is easy to implement and luckily, easy to avoid as described in Section 3.2.2.

When we made the attack trees we wanted to know what threats there are against home automation technologies using wireless communication. In response, we have concluded that there are a lot of threats ranging from violations against the personal integrity of the customer to connected devices being damaged by a hacker gaining access and inducing harmful behavior. As seen in the attack trees, not only does there exist a lot of different attacks to be made but also several different areas of the system can be subjected to these attacks to obtain several different attack goals. Among these are threats against the companies themselves considering they provide a service which they are held accounted for. Consequently, shutting down the company's servers with a DoS attack could prove fatal, especially to new or already financially unstable companies.

Our findings boil down to that security certainly is needed on these smart home systems as they possibly constitute a lucrative target for hackers. Also, there are some easy to do attacks to be made against these kinds of systems which open up for more sources of possible threats. We have also concluded that utilizing HTTPS and SSL goes a long way in securing the traffic being sent to and from the server and these protocols should be enforced where possible. The communication between the server and the web client does not use SSL and are therefore much more vulnerable than the communication between the server and the phone application. This flaw helped us understand the system better and helped us when implementing the attacks. This flaw could be avoided only by updating the web client since the server already supports communication using SSL.

Even though the local network is less exposed due to the relatively short range of the radio communication used there are severe security risks if left

without any security. As with any online service there are always threats of the servers being subjected to DoS attacks which are hard to protect against. Part of the solution could be to employ several servers to serve as back-ups. In all three of the produced attack trees, "Get login information" was included. Therefore, the protection of user passwords should be of high priority. This includes making sure users selects a proper password, protecting it in all aspects including when entering it at the login screen and when storing hashed value of the password in a database. It is also important to stay updated since software is constantly evolvinging to keep up with new threats. Using outdated versions of software can be a huge security risk and lead to problems that easily could have been avoided. Lastly, security policies should be well written and enforced in order for attacks utilizing human error or emotional misjudgment, such as an attacker impersonating a customer or employee of the company to achieve their goals, to be subdued.

Finally, we would like to point out that as long as you keep a system online there will always be a risk of someone finding a way to control and abuse it. The Internet of things evolve all the time and with it new problems arise. Ultimately the security needed in smart home technologies is very much dependent on the risk the user is willing to take and for instance the value he or she set on their integrity being maintained. The user might not care if someone else knows when he or she turns the light out and so forth, but someone being able to control their home might be another thing entirely.

References

[1] Neil Gershenfeld, JP Vasseur, *As Objects Go Online*, Foreign Affairs March/April 2014

[2] Gan Gang, Lu Zeyong, Jiang Jun, *Internet of Things Security Analysis*, Internet Technology and Applications (iTAP), August 2011

[3] Carlo Maria Medaglia, Alexandru Serbanati, *An Overview of Privacy and Security Issues in the Internet of Things*, Springer New York, 2010

[4] The Guardian, *NSA files decoded*, Retrieved: 2014-06-16, URL: http://www.theguardian.com/world/interactive/2013/nov/01/snowden-nsa-files-surveillance-revelations-decoded#section/1

[5] Telldus Technologies, Tellstick Net Overview, Retrieved: 2014-05-26, URL: http://www.telldus.se/products/tellstick_net

[6] Telldus Technologies, Tellstick Net Technical Specification, Retrieved: 2014-05-26, URL: http://www.telldus.se/products/technicalspecification_net

[7] Z-Wave Alliance, *Z-Wave For Developers And OEMs*, Retrieved: 2014-04-10, URL: http://www.z-wavealliance.org/z-wave-for-developers-oems

[8] International Telecommunication Union (ITU), *Short range narrow-band digital radiocommunication transceivers - PHY and MAC layer specifications*, Recommendation ITU-T G.9959, Feburary 2012

[9] Behrang Fouladi, Sahand Ghanoun, *Security Evaluation of the Z-Wave Wireless Protocol*, SensePost UK Ltd., BlackHat 2013 USA

[10] ZigBee Alliance, *ZigBee Specification Overview*, Retrieved: 2014-04-10, URL: http://www.zigbee.org/Specifications/ZigBee/Overview.aspx

[11] Institute of Electrical and Electronics Engineering, *IEEE 802.15 WPAN^{TM} Task Group 4 (TG4)*, Retrieved: 2014-04-10, URL: http://www.ieee802.org/15/pub/TG4.html

[12] Brad Bowers, *ZigBee Wireless Security: A New Age Penetration Tester's Toolkit*, Pearson Education, Cisco Press, January 2012

[13] Symantec Corporation, *Internet Security Threat Report 2014 Volume 19* Symantec Security Response, April 2014

[14] Steve Hawins, David C. Yen, David C. Chou, *Awareness and challenges of Internet security*, MCB UP Ltd, 2000

[15] Mikko T. Siponen, *Five dimensions of information security awareness*,

[16] Hasan Cavusoglu, Huseyin Cavusoglu, Jun Zhang, *Security Patch Management: Share the Burden or Share the Damage?*, Management Science, Informs, April 2008

[17] Min Wu, Simson Garfinkel, Rob Miller, *Secure Web Authentication with Mobile Phones*, Springer, 2001

[18] William Zeller, Edward W. Felten, *Cross-Site Request Forgeries: Exploitation and Prevention*, The New York Times, 2008

[19] Nenad Jovanovic, Engin Kirda, and Christopher Kruegel, *Preventing Cross Site Request Forgery Attacks*, Securecomm and Workshops, 2006

[20] Martin Johns, Justus Winter, *RequestRodeo: Client Side Protection against Session Riding*, OWASP Europe Conference, May 2006

[21] Pablos Holman, *Wireless Security*, TEDxMidwests, August 2012, URL: https://www.youtube.com/watch?v=hqKafI7Amd8

[22] Peter Bergstrom, Kevin Driscoll, John Kimball, *Making Home Automation Communications Secure* IEEE Xplore, 2001

[23] Bruce Schneier, *Attack Trees*, Dr Dobb's Journal, December 1999

[24] Margaret Rouse, Search Security, *man in the middle attack (fire brigade attack)*, June, 2007, Retrieved: 2014-05-26, URL: http://searchsecurity.techtarget.com/definition/man-in-the-middle-attack

[25] Paul Moceri, Troy Ruths, *Cafe Cracks: Attacks on Unsecured Wireless Networks*, December 2007

[26] Vivek Ramachandran, Sukumar Nandi, *Detecting ARP Spoofing: An Active Technique*, Springer Berlin Heidelberg, December 2005

[27] Men & Mice, DNS Spoofing, Retrieved: 2014-05-26, URL: http://www.menandmice.com/resources/dns-spoofing/

[28] Margaret Rouse, Search Security, *cache poisoning (domain name system poisoning or DNS cache poisoning)*, September 2005, Retrieved: 2014-05-26, URL: http://searchsecurity.techtarget.com/definition/cache-poisoning

[29] Joe Stewart, *DNS Cache Poisoning - The Next Generation*, Dell SecureWorks, 2002

[30] Microsoft Developer Network, Replay Attacks, Retrieved: 2014-05-26, URL: http://msdn.microsoft.com/en-us/library/aa738652%28v=vs.110%29.aspx

[31] Margaret Rouse, Search Security, *cross-site request forgery (XSRF or CSRF)*, October 2006, Retrieved: 2014-05-26, URL: http://searchsoftwarequality.techtarget.com/definition/cross-site-request-forgery

[32] W3C, *Cross-Origin Resource Sharing*, W3C Rekommendation, January 2014

[33] Kaspersky, *Trojan Virus*, Retrieved: 2014-05-26, URL: http://www.kaspersky.com/se/internet-security-center/threats/trojans

[34] GFI White Paper, *The corporate threat posed by email trojans*, GFI Software, July 2003

[35] Margaret Rouse, Search Security, *keylogger (keystroke logger, key logger, or system monitor)*, September 2010, Retrieved: 2014-05-26, URL: http://searchmidmarketsecurity.techtarget.com/definition/keylogger

[36] Matt Smith, *4 Ways To Protect Yourself Against Keyloggers*, May 2011, Retrieved: 2014-07-13, URL: http://www.makeuseof.com/tag/4-ways-protect-keyloggers/

[37] Mircrosoft Developer Network, *Denial of Service*, Retrieved: 2014-05-26, URL: http://msdn.microsoft.com/en-us/library/aa702790%28v=vs.110%29.aspx

[38] Margaret Rouse, Search Security, *distributed denial-of-service attack (DDoS)*, May 2013, Retrieved: 2014-05-26, URL: http://searchsecurity.techtarget.com/definition/distributed-denial-of-service-attack

[39] Jacques E. Belissent, *Method and apparatus for preventing a denial of service (DOS) attack by selectively throttling TCP/IP requests*, Sun Microsystems, September 2004

[40] Can I use Cross-Origin Resource Sharing? Retrieved: 2014-05-19, URL: http://caniuse.com/cors

[41] R. Fielding, J. Reschke, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content (RFC 7231)*, Internet Engineering Task Force, June 2014

[42] Jesse Burns, *Cross Site Request Forgery An introduction to a common web application weakness*, ISEC Partners, 2005

[43] Kammy Pow, *The Development of IE6*, eHow

Appendices

A csrf.js

```
/*
 * csrf.js is a "Cross-site request forgery" attack against the
 * the Telldus Live! server. To be able to use this you have to
 * disable security in your browser.
 */

//Finds all the devices
var devices = getAllDevices();

//Tries to turn all the devices off, on, off, on.
changeAllDevices(devices,'off');
changeAllDevices(devices,'on');
changeAllDevices(devices,'off');
changeAllDevices(devices,'on');

/*
 * changeAllDevices() changes the mode on all the given devices to
 * the given mode.
 *
 * @param <Array> devices, <String> mode
 * @return <void>
 */
function changeAllDevices(devices, mode) {
    for (var i = 0; i < devices.length; i++) {
        switch(mode) {
            case 'on':
                devices[i].turnOn();
                break;
            case 'off':
                devices[i].turnOff();
                break;
            default:
                console.log("%s is not a valid mode".
                    replace("%s",mode));
        }
    }
}
```

```
/*
 * Device() creates a new Device object with two properties,
 * <String> onURL and <String> offURL which can be used to turn on
 * or off the device if the user has signed in.
 *
 * @param <int> id
 * @return <void>
 */
function Device(id) {
    this.onURL =
        'http://live.telldus.com/device/switch?mode=on&id='.
        concat(id);
    this.offURL =
        'http://live.telldus.com/device/switch?mode=off&id='.
        concat(id);
    this.state = undefined;
    this.turnOn = turnDeviceOn;
    this.turnOff = turnDeviceOff;
}

/*
 * turnDeviceOn() is a function associated with the Device object
 * and sends a request to the Telldus Live! server to turn the
 * device on. If successful it changes the state on the device.
 *
 * @return <void>
 */
function turnDeviceOn() {
    var response = httpGet(this.onURL);
    console.log(this.onURL);
    console.log(response);
}

/*
 * turnDeviceOff() is a function associated with the Device object
 * and sends a request to the Telldus Live! server to turn the
 * device off. If successful it changes the state on the device.
 *
 * @return <void>
 */
function turnDeviceOff() {
    var response = httpGet(this.offURL);
    console.log(this.offURL);
    console.log(response);
}
```

```
/*
 * httpGet() makes a HTTP GET request to the given url.
 *
 * @param <String> url;
 * @return <XMLHttpRequest> xmlHttp
 */

function httpGet(url)
{
    var xmlHttp = new XMLHttpRequest();
    xmlHttp.open( "GET", url, false );
    xmlHttp.send();
    return xmlHttp;
}


/*
 * getAllDevices() makes a request to the Telldus Live! server
 * and returns an array of all the connected devices.
 *
 * @return <Array> devices
 */
function getAllDevices() {

    var url = "http://live.telldus.com/device/index";
    var response = httpGet(url);

    var parser = new DOMParser();
    var htmlResponse = parser.
        parseFromString(response.responseText,"text/html");
    var elements = htmlResponse.getElementById('maincontent').
        getElementsByClassName("listRowContainer");

    var devices = [];

    for (var i = 0; i < elements.length; i++) {

        var id = elements[i].id;

        // Checks if id is not an empty string to get rid of some junk.
        if (id != "") {

            // Extracts the device id from html tag's id
            id = parseInt(id.match(/\d+/g));
```

```
            // Checks if id is not 0 to get rid of some junk.
            if (id !== 0) {
                var device = new Device(id);
                devices.push(device);
            }
        }
    }
    return devices;
}
```

B index.html

```
<!doctype html>
<html>
    <head>
        <script type="text/javascript" src="csrf.js"></script>
    </head>
    <body>
        <p>
            Hello, World!
        </p>
    </body>
</html>
```