Man-in-the-browser attacks against IoT devices: a study of smart homes

Sampsa Rauti[1], Samuli Laato[1], and Tinja Pitkämäki[1]

University of Turku, Finland
sjprau@utu.fi, sadala@utu.fi, tievpi@utu.fi

Abstract. Smart environments such as smart homes are a collection of IoT devices, sensors, artificial intelligence and remote control systems. These technologies come with many possible benefits, such as improved energy-efficiency, easier maintenance and increased living comfort. On the other hand, ubiquitous use of these technologies raise cybersecurity and privacy concerns. Understanding the vulnerabilities, attack vectors and potential exploits of these systems is crucial for enabling precise and effective countermeasures. In this study, we investigate the potential of man-in-the-browser attacks for targeting the remote control systems of smart homes. We implement a malicious browser extension to the Chrome web browser that can alter the user input in a smart home management console. We empirically demonstrate that these browser extensions can manipulate IoT devices in a smart home. We discuss countermeasures for securing the remote control systems of smart homes against man-in-the-browser attacks.

Keywords: IoT security, browser security, cybersecurity, man-in-the-browser, smart home, smart environment

1 Introduction

The Internet of Things (IoT) consists of billions of interconnected devices and sensors – smart home appliances, industrial control systems, smart traffic systems, medical devices, and other networked devices all exchanging data. The fact that IoT makes everyday life easier cannot be denied. However, the security is often poor and technology is prone to advanced attacks. These remaining challenges degrade users' trust towards connected devices. The cybersecurity risks are present at every step along the journey when transferring data in a distributed IoT system, and there are many adversaries that wish to take advantage of a system's weaknesses.

One particular threat in today's internet is the man-in-the-middle attack (MitM) [1], in which the adversary stealthily relays and possibly modifies the data transmitted between two communicating parties. A malicious piece of code can alter or spy on the data in outgoing and incoming messages, while the communicating parties (usually the user and the server) think they are directly exchanging messages with each other and do not notice anything suspicious.

As cryptography and endpoint authentication have made MitM attacks between endpoints more difficult, adversaries are increasingly targeting the endpoints themselves. A man-in-the-browser (MitB) attack is a special subtype of MitM attack taking place at the communication endpoint. As the name suggests, this attack intercepts and possibly modifies the data inside a web browser [7].

There are several reasons why we think MitB attacks are a significant threat for IoT systems. First, the trend of migrating from desktop applications to web environment also affects the IoT environment. IoT devices and smart systems are increasingly being monitored and controlled using web-based user interfaces, which exposes them to MitB attacks. Second, the huge growth in the number of IoT devices is making them common and interesting targets for cyber attacks. Finally, the increasing interconnection between the real world and cyberspace makes the potential effects of a MitB attack much more serious, as altering data can have immediate consequences in the physical world.

The aim of this study is to investigate MitB against IoT devices in the context of smart environments, more specifically, smart homes. To this end, we create a proof-of-concept implementation for a JavaScript-based MitB attack and empirically test it against a web interface for smart home devices, Mozilla WebThings [1]. We also discuss potential MitB attacks scenarios in the smart home environment and assess their consequences. Based on these tests and previous work, we suggest countermeasures for smart device designers, software vendors and smart home residents to guard their systems against MitB attacks.

2 Basics of a MitB attack

Because of point-to-point encryption, modifying or eavesdropping traffic in a network is challenging. Instead, it is much easier for the adversary to target an endpoint of communication – that is, the user's device – where more attack vectors exist and the user can make mistakes that compromise cybersecurity. Infecting the user's web browser and performing a MitM attack before the data is encrypted is a tempting option. The fact that software developers and device manufacturers often do not consider MitB attacks a serious threat [2] is not making the situation any better.

As applications are increasingly being moved to the web environment, adversaries build malicious browser extensions that intercept data inside the web browser. As discussed, this kind of deceitful proxy inside the browser is called a MitB attack. The malicious browser extension aims to spy on or alter the data transmitted between the user and the server [7, 16]. More specifically, the following functionalities are possible:

- capturing and storing sensitive data and delivering it to the adversary's command and control server [4]
- tampering with the data in incoming or outgoing messages [7]
- modifying the contents of web pages before they are rendered

[1] https://iot.mozilla.org/framework/

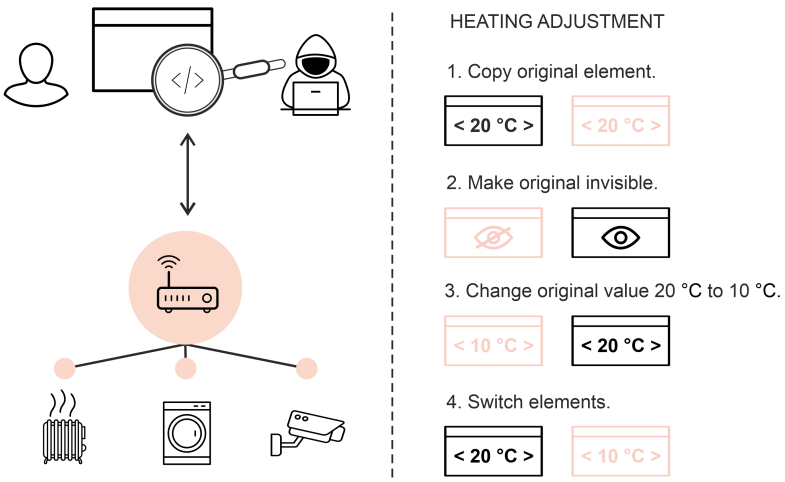

Fig. 1. A MiTB attack in the smart home environment (left) and the phases of an attack adjusting the room temperature (right).

– issuing additional malicious HTTP requests that are not initiated by the user

A MitB malware can contain one or several of the above functionalities. The malicious extension usually does its job silently, without the user or the server noticing its existence. Figure 1 depicts a MitB attack in the smart home environment.

Next, we describe an example scenario clarifying how a MitB attack can be used to stealthily modify the data given by the user and sent to the server, as well as the data later returned by the server. The attack alters the data on the user interface level and proceeds as follows:

1. The user's client device is infected with the malware. Implemented as a browser extension, the malware hides and executes inside the browser.
2. The malware contains a list of web addresses that activate the MitB functionality. In this case, opening the web page (user interface) for monitoring and controlling IoT devices launches the MitB attack.
3. The malware waits until the user causes the web application to make a HTTP request – for instance, the user adjusts the room temperature through a smart home user interface.
4. Before the user's request is transmitted to the server, the malware modifies it according to the adversary's wishes. One way of doing this is to make use of the DOM (Document Object Model) interface to manipulate values of HTML elements. For example, the adversary could change the room temperature set by the user.

5. After intervening with the HTTP request and changing values submitted by the user, the MitB extension allows the browser to proceed with sending the data to the server.
6. The tampered HTTP request with the modified values are transmitted to the server. Naturally, the server does not know the request has been modified by a malicious program and accepts the values as a valid input, trusting it reflects the users real intent.
7. When the user checks the status of an IoT device or sensor, for example the room temperature, the malware changes the value displayed in the user interface to deceive the user. In our example, the user thinks the temperature corresponds to the value they have set previously, while the real temperature controlled by the IoT device is the one set by the adversary.

The user and the IoT devices involved in the data exchange have been deceived. At some point the user may notice that the real room temperature is not the one they have set using the web interface. Even then, realizing where the problem is, finding the malicious extension and getting rid of it can be quite challenging.

3 Study context: smart homes

Smart homes are perhaps the most well-known sub-category of smart environments. We define smart homes as *"houses enhanced with sensors and remotely controllable devices which utilize intelligence to increase the level of automation"*. This definition aligns well with that of Jiang et al. [8] who provide three defining characteristics which must exist within a dwelling for it to be called a smart home: (1) internal network; (2) home automation; and (3) intelligent control. Based on these characterizations, a smart home contains at least two or more smart devices or sensors that are connected to each other, either directly or via a central control system. Because of the strong link between smart home technologies and the real world, stakeholders place high cybersecurity requirements on these systems. As such, a single security solution is not enough and scholars emphasize the importance of implementing multi-layered security measures in smart home technologies [12]. Security of smart homes can also be improved by providing access to remote monitoring, which can further be supplemented with AI solutions to also monitor the behavior of IoT devices [18].

Forcing smart home devices and sensors to connect to the internet only through a central control system can enhance the cybersecurity of a smart home. Lin and Bergmann call this the gateway architecture [10]. Having a central control hub enables a more effective monitoring of online traffic within the house, and offers house residents a way to monitor and control their home through sensor data and connected devices [15]. The sensor data needs to be secured to ensure residents' privacy and the device control needs to be secured to ensure residents' security. However, house residents do not typically directly connect to the control hub with their smart devices, instead, a cloud server is implemented

in most commercial solutions between the hub and the user, which is controlled by the hub provider [15]. This solution compromises user's privacy as their data is shared to a third party. On the other hand, it enables the hub provider to gather training data for smart home AI solutions which can then be used to further optimize, for example, the energy consumption of smart homes.

In a gateway smart home architecture [10], securing the connection between the hub and the cloud server, and the cloud server and the user, is thus paramount for maintaining user's privacy and security. These connections are password encrypted and cannot be interpreted or modified by a malicious agent, but as previously discussed, the MitB attack may be harnessed to interfere with the input of the user as they are accessing their home's control hub [13]. This way, even though the communication between the endpoints is encrypted, the malicious modifications to the user input (conducted by the MitB attack) are already included in the communication on the client device. In the next section, we describe a proof-of-concept implementation of how this kind of an attack can be carried out against smart home control interfaces.

4 The proof-of-concept implementation

In this study, a MitB attack is implemented as a Chrome web browser extension. This extension manipulates the data the user has filled in and the settings they have made on a web site before they are sent to an IoT device. We also carry out experiments with the extension by testing it against popular smart home interfaces.

A MitB attack can be implemented on many layers [13]. For example, the adversary can modify the data in the HTTP requests or use DOM to manipulate data before it is sent to the server. Because intercepting HTTP traffic has been made difficult in Chrome by restricting this functionality for extensions [11], we opt for UI-level DOM modification, which is an easier and browser agnostic method. When manipulating the UI and replacing the user's input with modified data using DOM, the adversary has to do this silently without the user noticing anything out of ordinary. Simply changing the value of an HTML element in the UI which is easily seen by the user, does not work, for example. There are many ways to modify the input data stealthily, our implementation uses the following one:

1. Find the UI element O containing the value that will be modified.
2. Make a copy C of the original element O.
3. Make O invisible.
4. Replace the value of the invisible element O with a modified value.
5. Insert the copied element C in the place of O.

The user will only see the copy of the original element and interact with it. However, only the value of the original, hidden element, modifiable by the malware but unreachable to the user, is going to be sent to the server. The same technique can be used to change values of multiple elements. In a smart

home management console, several settings can be changed and several devices controlled with one attack.

There is one more thing we have to implement: the malicious extension is also supposed to deceive the user when information about IoT devices is displayed. On pages that display status information about the smart home, the extension just searches the elements with the data that was sent to the server earlier (e.g. the real room temperature) and replaces it with the original data the user filled in. This can be achieved by storing the data input by the user, and displaying it in the appropriate element instead of the modified value that was actually sent to the server. The whole modification functionality of the extension can be written in just a couple of lines of basic JavaScript making use of the DOM API. Full implementation of the MitB extension is available upon request.

In the Chrome browser, extensions are normally only installed from the Chrome Web Store. However, users can also experiment with an extensions by turning on Chrome's developer mode. If the adversary cannot slip the extension into Chrome Web Store (which actively tries to screen new extensions and remove the malicious ones, but sometimes fails to detect dubious extensions), they can use social engineering techniques to convince the user to use developer mode to launch the extension. Because our extension uses JavaScript and DOM, the same code could also be used in a Firefox extension with very little effort. In this sense, our implementation of the MitB attack is browser and platform agnostic.

5 Attack scenarios

There exists countless of IoT devices, and in addition, basic home functionality such as doors, light switches, heating and air conditioning can be connected to a control hub and controlled remotely. The exact composition of a smart home varies between houses. Due to this, in the following attack scenarios we focus on some of the more common IoT devices and sensors that are currently popular in smart homes, such as smart plugs, thermostats or fridges.

Scenario 1: Smart plug. A smart plug is essentially a switch that can be used to turn on and off all kinds of electronic devices connected to it. If a MitB attack turns the smart plug on, this can lead to potentially dangerous situations and at the very least, increase energy consumption. One example could be turning on a radiator plugged into smart switch and causing a fire. A fridge could be turned off so that the food in the fridge goes bad or stereos could be turned on with a high volume. Many smart plugs can also be timed and this timing can be programmed through a web interface. The problem with smart plugs is that many devices that were never supposed to be remotely controlled can now be connected to the internet. Had these devices been designed for remote use, many of them would have more sophisticated safety mechanisms.

Scenario 2: Spying on the smart home. Malicious extensions with MitB functionality do not always aim to modify data. Often the mission is to spy on the user and report the information back to the adversary's server. The information

could be sold to third parties or used for criminal purposes (e.g. for finding out when the user is not at home). As more and more devices are added to smart homes, there is lots of data on people's personal lives available for malicious adversaries. Spying can be done very stealthily and cannot be easily detected. When the data is being sent back to the adversary, an intrusion detection system might notice a suspicious connection in some cases depending on how the data is delivered.

Scenario 3: Smart thermostat. If a smart thermostat was allowed to be turned off, a MitB attack could cause the indoors temperature to become freezing. This can of course be prevented by designing the device correctly and setting limits for the room temperature.

Scenario 4: Removing a smart device. A MitB attack can easily modify the user interface so that a certain IoT device is no longer displayed, essentially causing a denial of service. The user can no longer control the device (through the web interface) but the MitB malware can keep controlling the device as long as it is connected to the web. Another option would be to leave the device in the web interface but prevent any changes when the user tries to control it. In both cases, the user would probably think the device is malfunctioning and finding the real culprit would be very challenging.

We tested the feasibility of a MitB attack by building an extension to Google Chrome. The extension was tested against Mozilla WebThings, a piece of software for smart homes allowing users to monitor and control their smart devices over the web. By manipulating the user input, we successfully controlled a smart plug so that it was always on.

6 Suggestions for mitigating MitB attacks

Most scenarios against IoT devices covered in this study are dependent on the actions that the IoT devices offer the user, and consequently, a adversary through the MitB attack. There are several countermeasures that IoT device developers and software vendors can implement to mitigate the threat. Based on these findings, we suggest countermeasures against the MitB attack for (1) IoT device and smart home hub developers; (2) browser vendors and virus protection providers; and (3) individuals using IoT devices.

6.1 Suggestions for IoT device and control hub developers

Smart home device and control hub designers and manufacturers have the obvious burden of creating systems which cannot be exploited by adversaries. In doing so, a solid architecture and design are important. Following the guidelines of Lin and Bergmann, smart homes should not allow individual IoT devices to freely connect to and be visible on the internet, but should rather operate and be accessed through a control hub- a gateway [10]. The cybersecurity considerations of any networked device apply both to the control hub and the individual

IoT devices nonetheless. Furthermore, situations where a user owns only a single IoT device but not a control hub need to be considered, as denying IoT devices from using the internet without a hub in between can limit the usability of these devices and can therefore be a sub-optimal solution. In addition to these considerations, we list the following guidelines for IoT device designers and manufacturers who wish to secure their devices against MitB attacks.

- *Input validation.* The user input should always be validated. The developer has to ensure that the values sent to IoT device are feasible and cannot cause damage, such as allowing the user to set the room temperature to freezing. The user input should never be trusted, users can make mistakes or a malicious program can modify the input on the fly.
- *Out-of-band verification.* Use out-of-band (OOB) verification to verify input that differs from the user's usual behavioral patterns or is potentially harmful (such as shutting down a fridge). In OOB verification, the transaction initiated by the user is verified using a second channel other than the client device [5, 19]. For example, a mobile application can be used to verify a command sent to an IoT device. However, if this kind of verification is used all the time, it becomes a routine and users may just allow any transaction without really thinking. OOB verification also degrades usability, as users may feel it adds unnecessary complexity. Therefore, OOB verification should be invoked sparingly.
- *Anomaly detection.* Use an intrusion detection system and anomaly detection [18, 6] to find unusual patterns in the network traffic between the client and the control hub. For example, the number of commands sent to IoT devices may increase substantially due to a MitB attack or their contents may be unusual.
- *Multi-layered security.* Enhance the multi-layered security of IoT devices by applying proactive cybersecurity measures to their operating systems such as internal interface diversification [12].

6.2 Suggestions for browser developers and antivirus software vendors

In many ways, web browsers are becoming more like operating systems, because they are now platforms for running web applications, and there is the trend of migrating many desktop applications to the modern web. The security of the browser environment, however, is not fully keeping up with this development. Just as programs in an operating system, extensions and applications running inside a web browser should be better monitored.

- *Ensuring DOM integrity.* Inside browsers, the functionality of extensions and flow of data could be better monitored. For example, an extension making critical input fields invisible is suspicious. Also, there are solutions to cryptographically ensure the integrity of web pages [17] so that DOM cannot be manipulated.

- *Hardened browser.* Hardening limits the attack surface by pruning off functionality and implementing other measures that thwart attacks. Ronchi and Zakhidov suggest loading a web browser from an an external tamper-proof device [14]. A hardened browser would not allow installing extensions. Therefore, realizing a man-in-the-browser attack becomes much more difficult for the adversary. Using an external device is a solution that degrades usability and probably is not a good fit for an ordinary home computer. However, browser vendors should still consider minimizing the attack surface. Extensions could also be disabled for certain web pages.
- *Improved malware detection.* The detection rate for malicious browser extensions is still low [3]. Therefore, anti-virus software developers and browser vendors should co-operate in order to monitor and detect malicious extensions.

6.3 Individual's precautions

A MitB attack is invisible to the user when it is happening. However, as the MitB malware used in this work was delivered as a browser extension, individuals need to be careful with the extensions they install. This is a problem that goes beyond the IoT environment, as previous work has shown MitB attacks to be effective against, for example, online banking systems [11]. In addition, users should be encouraged to educate themselves on the aspects of cybersecurity to increase their ability to identify and respond to cybersecurity risks within smart homes [9]. Users should also gain a better understanding about the meaning of permissions given to browser extensions and learn to avoid social engineering schemes that are often used to spread malware.

7 Conclusions and Future Work

In this work, we looked at a JavaScript-based MitB attack and empirically observed how it can be used to exploit currently available browser-based IoT device control interfaces. We have seen that MitB attacks can pose a significant cyber security and privacy treat in smart home environment. Still, it seems MitB attacks are not considered a serious threat by software and device vendors.

Naturally, it is important to understand that IoT devices have numerous other vulnerabilities that were not covered in this study. In addition, there are several smart home solutions and control interfaces we did not test. As such, our results are not aimed to provide a comprehensive and precise description of attacks against individual systems, but rather prove the magnitude and potential of MitB attacks in the context of smart homes.

As IoT devices are becoming popular for smart environments such as smart homes, and real and virtual worlds are getting increasingly tangled together, it is clear that cybersecurity research needs to keep up with the changes. Consequently, further work is needed to devise effective counter-measures against concrete MitB attacks as well as solutions for detecting them.

References

1. Bhushan, B., Sahoo, G., Rai, A.K.: Man-in-the-middle attack in wireless and computer networking—a review. In: 2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA)(Fall), IEEE (2017) 1–6
2. Blom, A., de Koning Gans, G., Poll, E., De Ruiter, J., Verdult, R.: Designed to fail: A usb-connected reader for online banking. In: Nordic Conference on Secure IT Systems, Springer (2012) 1–16
3. DeKoven, L.F., Savage, S., Voelker, G.M., Leontiadis, N.: Malicious browser extensions at scale: Bridging the observability gap between web site and browser. In: 10th USENIX Workshop on Cyber Security Experimentation and Test (CSET 17), Vancouver, BC, USENIX Association (2017)
4. Dougan, T., Curran, K.: Man in the browser attacks. International Journal of Ambient Computing and Intelligence (IJACI) **4**(1) (2012) 29–39
5. Entrust: Defeating man-in-the-browser malware – how to prevent the latest malware attacks against consumer and corporate banking. White paper. (2014)
6. Fernandes, G., Rodrigues, J.J., Carvalho, L.F., Al-Muhtadi, J.F., Proença, M.L.: A comprehensive survey on network anomaly detection. Telecommunication Systems **70**(3) (2019) 447–489
7. Gühring, P.: Concepts against man-in-the-browser attacks. Tech. Report. (2006)
8. Jiang, L., Liu, D.Y., Yang, B.: Smart home research. In: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826). Volume 2., IEEE (2004) 659–663
9. Laato, S., Farooq, A., Tenhunen, H., Pitkamaki, T., Hakkala, A., Airola, A.: Ai in cybersecurity education-a systematic literature review of studies on cybersecurity moocs. In: 2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT), IEEE (2020) 6–10
10. Lin, H., Bergmann, N.W.: Iot privacy and security challenges for smart home environments. Information **7**(3) (2016) 44
11. Rauti, S.: Man-in-the-browser attack: A case study on malicious browser extensions. In: International Symposium on Security in Computing and Communication, Springer (2019) 60–71
12. Rauti, S., Laurén, S., Mäki, P., Uitto, J., Laato, S., Leppänen, V.: Internal interface diversification as a method against malware. Journal of Cyber Security Technology (2020) 1–26
13. Rauti, S., Leppänen, V.: Man-in-the-browser attacks in modern web browsers. In: Emerging Trends in ICT Security. Elsevier (2014) 469–480
14. Ronchi, C., Zakhidov, S.: Hardened client platforms for secure internet banking. In: ISSE 2008 Securing Electronic Business Processes. Springer (2009) 367–379
15. Stojkoska, B.L.R., Trivodaliev, K.V.: A review of internet of things for smart home: Challenges and solutions. Journal of Cleaner Production **140** (2017) 1454–1464
16. Ståhlberg, M.: The trojan money spinner. In: Virus bulletin conference. Volume 4. (2007)
17. Toreini, E., Shahandashti, S.F., Mehrnezhad, M., Hao, F.: Domtegrity: ensuring web page integrity against malicious browser extensions. International Journal of Information Security (2019) 1–14
18. Zainab, A., S Refaat, S., Bouhali, O.: Ensemble-based spam detection in smart home iot devices time series data using machine learning techniques. Information **11**(7) (2020) 344
19. Zhang, P., He, Y., Chow, K.: Fraud track on secure electronic check system. International Journal of Digital Crime and Forensics **10**(2) (2018) 137–144