Data Exfiltration in IoT Protocols

Daniel Uroz[1], Ricardo J. Rodríguez[1]

[1] Distributed Computing Group (DisCo)
Instituto de Investigación en Ingeniería de Aragón (I3A)
Universidad de Zaragoza, Mariano Esquillor s/n, 50018, Zaragoza, Spain.
Tel. +34-976762707
e-mail: duroz@unizar.es, rjrodriguez@unizar.es

Abstract

We investigated how IoT protocols can be used for data exfitration. We theoretically define how they can be used to exfiltrate data and we compare them to traditional protocols. Finally, we present the Python library *chiton* and we empirically measure the data exfiltration elapsed time for different amounts of data.

Introduction

Data exfiltration is the unauthorized transfer of information from one information system to another [1]. As mostly enterprises deploy some sort of defense mechanism (like a firewall), adversaries try to mask that transference is taking place using covert channel techniques. A covert channel is any communication to transfer information in a manner that violates the systems security policy [2]. The most common example is to tunnel one protocol into another, being the latter the covert channel.

Internet of Things (IoT) networks deploy various sensors, objects and smart nodes that are capable of communicating with each other without human intervention [3]. Most commonly, these things generate information that is lately sent to an outside network responsible of recollection and processing. In order to communicate, they rely on IoT protocols specially design for constrained devices.

We call traditional protocols to those included in the Internet protocol suite and usually allowed in all networks, regardless of its nature. Namely, we focus on Internet Control Message Protocol (ICMP), Network Time Protocol (NTP), and Domain Name System (DNS). Whereas IoT protocols are specially design for constrained devices. In this study, we focus on Constrained Application Protocol (CoAP), Message Queuing Telemetry Transport (MQTT), and Advanced Message Queuing Protocol (AMQP).

Characteristics of Protocol Exfiltration

In the context of data exfiltration, a protocol can be described based in three main characteristics:

1. *Packet type*: each protocol defines a series of packet types, each one tailored for a specific purpose. Depending of the type, it can be classified as:
   a. *Payload*: how much data a protocol is able to carry in a single packet.
   b. *Overhead*: every byte not representing the actual data to exfiltrate.
2. *Transport*: depending on the protocol, it can rely on a connectionless transport layer like UDP, which is lightweight for constrained devices but it incurs in a probability of lost. On the contrary, a connection-oriented protocol like TCP can add more latency thanks to error detection or retransmission for packet reliability.
3. *Error detection*: as exfiltration consists in sending data over a network, a desirable characteristic is to count on any type of checksum redundancy to spot when received data have been corrupted.

Comparative

For the sake of space, we have omitted the tables with the information, which can be found in [4]. Both tables represent the transmission of 1MiB of data and they are divided in two main columns, one column for a *stealthy adversary* and another column for a *rough adversary*. A stealthy adversary tries to adequate outgoing packets to commonly used sizes, whereas a rough adversary maximize the possible payload for every packet.

Experimentation

We developed the Python library *chiton* to exfiltrate data encapsulating the data into IoT protocol's packets. This library is licensed under GNU GPL v3, and it can be accessed alongside its documentation in:

https://github.com/reverseame/chiton

In this experiment scenario, we simulated a basic adversary model. We tested how IoT protocols perform for different types of data varying from 1, 10, 100, 1000, 10000 to 100000 KiBs.

Results and Discussion

The results are presented in Figure 1, where a big difference between the data exfiltration time from CoAP protocol to the MQTT and AMQP protocols can be spotted. We can interpret three main reasons for these results:

- CoAP protocol needs to send more packets for the same amount of data. This behavior implies more time overhead since the OS, in conjunction to the network device, must deal with the underlying I/O network more frequently.
- For the CoAP protocol, we kept IP packets under the 1280 bytes length limit. However, there is a lack of knowledge about the maximum transmission unit (MTU) used in the network. Thus, despite using the upper-bound limit established in the CoAP standard specification, UDP does not count with any mechanism to recover the MTU in use, and UDP packets can be bigger than the actual MTU. This limitation of the transport layer implies, for example, that an intermediate node with a smaller MTU needs to fragment UDP packets during transmission while the final end hostnode is responsible to reassemble these fragments, consuming more time.This specific problem is addressed with the use of TCP/IP, as in AMQP and MQTT, thanks to Path MTU Discovery mechanisms.
- Depending of how the intermediate nodes are configured, they may apply more priority to TCP traffic over UDP. Hence, UDP traffic could be being buffered by intermediate nodes and thus arrive later.

Conclusions

While there are works focusing in how the protocols can be use to transmit data; to the best of our knowledge, this is the first study to extensively compare these IoT protocols from the point of view of data exfiltration, focusing on characteristics such as overhead and useful payload for every available packet.

Additionally, we empirically measure and compare the time necessary to exfiltrate files of different data size. The results show that CoAP is the less suitable protocol for data exfiltration, and being outperformed by both MQTT and AMQP. This preference is also supported from the point of view of an adversary, since this protocols are usually used to connect enterprise IoT networks to IoT cloud providers, so they are more likely to be allowed in the network of the organization. In this matter, MQTT is the protocol supported for the thre emajor cloud providers (Amazon Web Services, Microsoft Azure, and Google Cloud).

Acknowledgments. The research of D. Uroz was supported by the Government of Aragón under a DGA predoctoral grant (period 2019-2023).

References

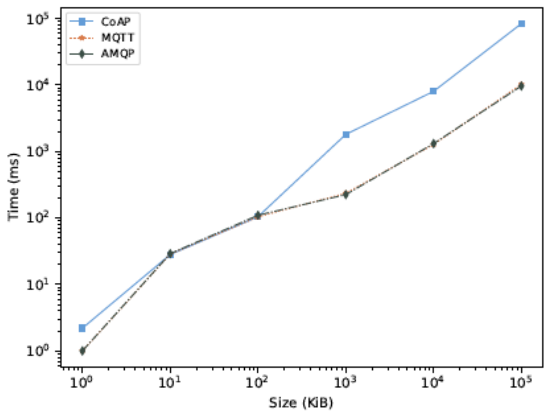
[1] NIST, Glossary: Exfiltration. URL: https://csrc.nist.gov/glossary/term/exfiltration (visited on 2020-05-21).

[2] U.S. Department Of Defense, Trusted Computer System Evaluation Criteria, pp. 1–129. London: Palgrave Macmillan UK, 1985.

[3] M. Conti, A. Dehghantanha, K. Franke, and S.Watson, “Internet of Things security and forensics: Challenges and opportunities,” Future Generation Computer Systems, vol. 78, pp. 544 – 546, 2018.

[4] D. Uroz. Data Exfiltration in IoT Protocols. Master's Thesis, University of León, Spain, Sept. 2020. Online; https://webdiis.unizar.es/~ricardo/files/TFMs/Exfiltracion-Datos-Protocolos-IoT_TFM_ULE.pdf. Accessed on December 10, 2020.

Figure 1. Comparison of data exfiltration times by IoT protocol.


CoAP
MQTT
AMQP
Time (ms)
Size (KiB)
10^0
10^1
10^2
10^3
10^4
10^5