TEEP WG
Internet-Draft
Intended status: Informational
Expires: December 30, 2020

S. Faibish
Dell EMC
M. K. Chowdhury
Deloitte Canada

June 30, 2020

Test Tools for IoT DDoS vulnerability scanning
draft-faibish-iot-ddos-usecases-03

Abstract

This document specifies several usecases related to the different ways IoT devices are exploited by malicious adversaries to instantiate Distributed Denial of Services (DDoS) attacks. The attacks are generted from IoT devices that have no proper protection against generating unsolicited communication messages targeting a certain network and creating large amounts of network traffic. The attackers take advantage of breaches in the configuration data in unprotected IoT devices exploited for DDoS attacks. The attackers take advantage of the IoT devices that can send network packets that were generated by malicious code that interacts with an OS implementation that runs on the IoT devices. The prupose of this draft is to present possible IoT DDoS usecases that need to be prevented by TEE. The major enabler of such attacks is related to IoT devices that have no OS or unprotected EE OS and run code that is downloaded to them from the TA and modified by man-in-the-middle that inserts malicious code in the OS. This draft adds list of MUD files for most IoT devices.

Table of Contents

1. Introduction

Problems with IoT devices arise from the fact that manufacturers ship their devices with almost no security measures and the companies that buy these IoT devices don't have proper visibility/understanding of their networks with these new products. Applications executing in an IoT device are exposed to many different attacks intended to compromise the execution of the application, or reveal the data upon which those applications are operating. The problem is more acute for IoT devices that run low level of OS or no OS at all and have limited ability to prevent malicious network traffic leading to DDoS. These attacks increase with the number of applications running on the device, with such other applications coming from potentially untrustworthy sources or due to man-in-the-middle mangling with the application code inserting random packets in the communication of the IoT back to operator.

The potential for attacks generated by these devices further increases with the complexity of features and applications on devices, with limited OS capabilities, running code that is downloaded from untrustworthy operators.

The danger of attacks on an OS-less system increases as the data transmitted by the devices to the operator increases. There is provision in the MUD protocol [RFC8520] to add security measures but it does not replace other security measures and only complement existing security measures if they are already in place.

But for MUD to work, every IoT device requires a unique MUD-URL specific to the kind of device type/model and matches the needed configuration manifest. There is a MUD file that SHOULD be provided by the device manufacturing that contains instructions for the expected behavior of the device. For cheap OS-less devices the manufacturer does not always provide the accurate behavior and require the network routers to detect malicious traffic and stop it. Abnormal behavior could be limiting the peak request rate of how many requests per minute is normal for the device that is used as prevention control of DDoS. Unfortunately IoT devices manufacturers do not always generate MUD profiles specific to their devices or even their companies. MUD in itself is an important step towards securing IoT devices but it is not enough for preventing DDoS attacks.

As an example, an IoT device that sends pollution data each minute from city wide sensors to a cloud application that analyses city air quality and generate reports and warning to the public can be used to send random data at much higher frequency like 1000 per second. This malicious transmission can shut down the cloud receiving this data. The worst part of this is that the IoT device OS has no idea that the transmission is wrong and is creating DDoS for the cloud used by the IoT devices. Additional there could be coordinated attacks coming from many IoT devices connected to the same cloud and shut down all the cloud services.

In general case there is an edge server to which the IoT devices are connected and the server is managing the management of the data transmitted to the OA. In this case the edge server has an OS and a TEE that can prevent DDoS attacks that were generated by the IoT devices if the transmission is malicious. Moreover the edge server will facilitate the code upgrade and prevent malicious code being stored on the device code. So, the edge server will become the TEE for all the devices connected to it. Moreover if the code of the device is compromised the edge server will block the packets that were generated by the IoT devices connected to it.

According to analysts study DDoS originated from IoT devices accounted for 90% of all the DDoS attacks and increased 10x in 2018 ([1]) and the majority of the attacks were from devices with limited compute and OS resources as well as webcams with REE.

This will require special TEE protocol support preventing the use of these devices for DDoS attacks. This draft is trying to present the usecases that enable such attacks with the intention to request that TEEP WG addresses this special security loophole. And the major problem resides in the inability of IoT devices to prevent broadcasting network packets generated by unauthorized code, inserted at upgrade time, to execute on devices with low compute capabilities.

Trusted Execution Environments (TEEs), including Intel SGX, ARM TrustZone, Secure Elements, and others, can enforce that only authorized code can execute within the TEE, and any memory used by such code is protected against tampering or disclosure outside the TEE. This observation is only true if there is awareness that IoT devices are enabled to send data back to the cloud and or the SP that did the upgrade. In such environments malicious code includes a method of external triggered or time based attacks.

In most such devices there is none or limited "Trusted Agent" or "Trusted Application Manager (TAM)" on the client side running inside the TEE. The purpose of this draft is to present 3 DDoS usecases that TEEP needs to address prevention of using the IoT devices as the origin of such attacks.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document also uses various terms defined in [I-D.ietf-teep-architecture], including Trusted Execution Environment (TEE), Trusted Application (TA), Trusted Application Manager (TAM), Agent, and Broker.

3. Assumptions

This draft assumes that an applicable device may or may not be equipped with any TEEs nor pre-provisioned with a device-unique public/private key pair, which is securely stored.

A TEE uses an isolation mechanism between Trusted Applications to ensure that one TA cannot read, modify or delete the data and code of another TA. We also assume that there can be a TEE running in a edge server to which the devices may be connected. The edge server will include such a TEE and will become the secure gateway as client/agent.

4. Usecases

4.1 Upgradable OS less IoT devices

The simplest IoT device we refer to here is a device that has enough OS and EE to perform a single function like sending back to the broker time series at given time intervals, Figure 1. As an example an IoT device that monitors the air quality in a city and send back to the cloud this data that will be aggregated with many sensors around the city. The device will run simple code that can be executed on the device and at a minimum it will be capable to receive and install code upgrades from the Broker. Such devices have very limited or no security or trust protection and it can be exposed to man-in-the-middle attacks target by malicious actors that are trying to insert malicious code MA (Malicious Application) in the upgraded code.

One example of such code may include a trigger, that can be activated in a similar manner as the code upgrade request, and used to start DDoS attacks coordinated as a cluster. As the device function is to send time series data to the cloud the malicious code can send same data 1000s of times fludding the recipient cloud from all the devices in the cluster. As a second example the malicious code can use a timer and start sending empty network packets back to the provider network and also targeting a given IP address of a victim. Such examples are attackes related to Mirai botnets also in [2] as similar attacks were uncovered tergeting for example financial institutions in order to hide cyberattacks for stilling money or even crypto-currency.

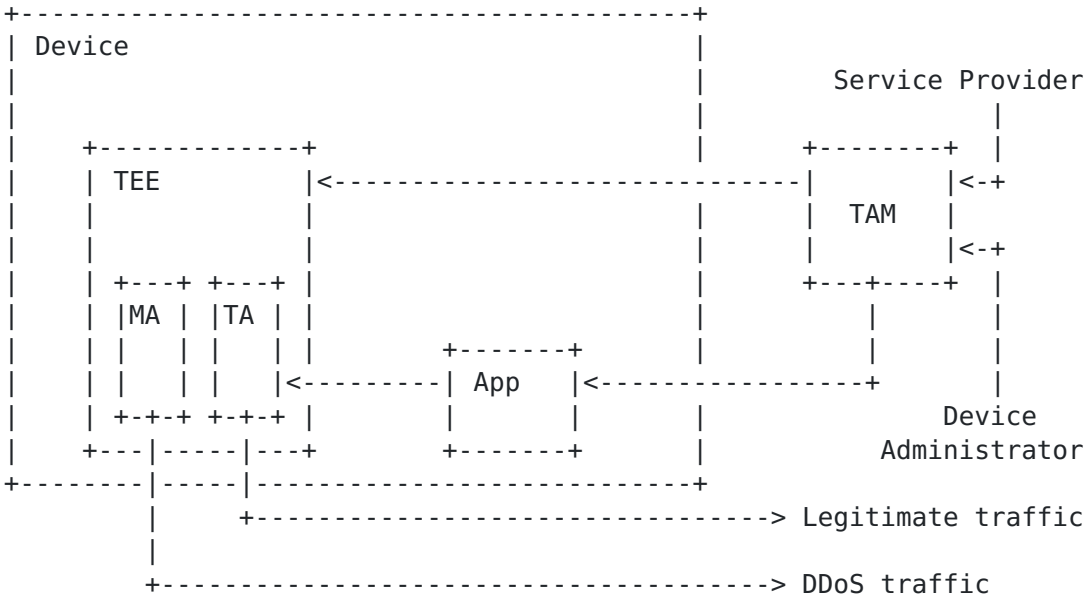

Figure 1: OS less IoT devices diagram

4.2 IoT devices connected to a gateway server

In this case the OS less IoT device is connected to a local edge TEEP server which has rich execution OS and acts as a bridge between the device and the cloud collecting the time series from the device. In this usecase the upgrades are done via the edge gateway server that has full TEEP capabilities and can detect DDoS attacks and prevent the DDoS traffic to escape outside the edge server. For example the edge gateway server could be a computer with full security protection or a mobile device such as a tablet or even a cell phone. We assume that in this case the edge server has a full TEE and can manage several IoT devices running multiple different applications. We also assume that the edge server is connected to a TEEP Broker. For example webcams can be such IoT devices connected to gateway server used for DDoS attacks [1] spoofing [4]. A list of such IoT devices MUD files is found in [3] used by python tool [5] to test vulnerabilities is available.

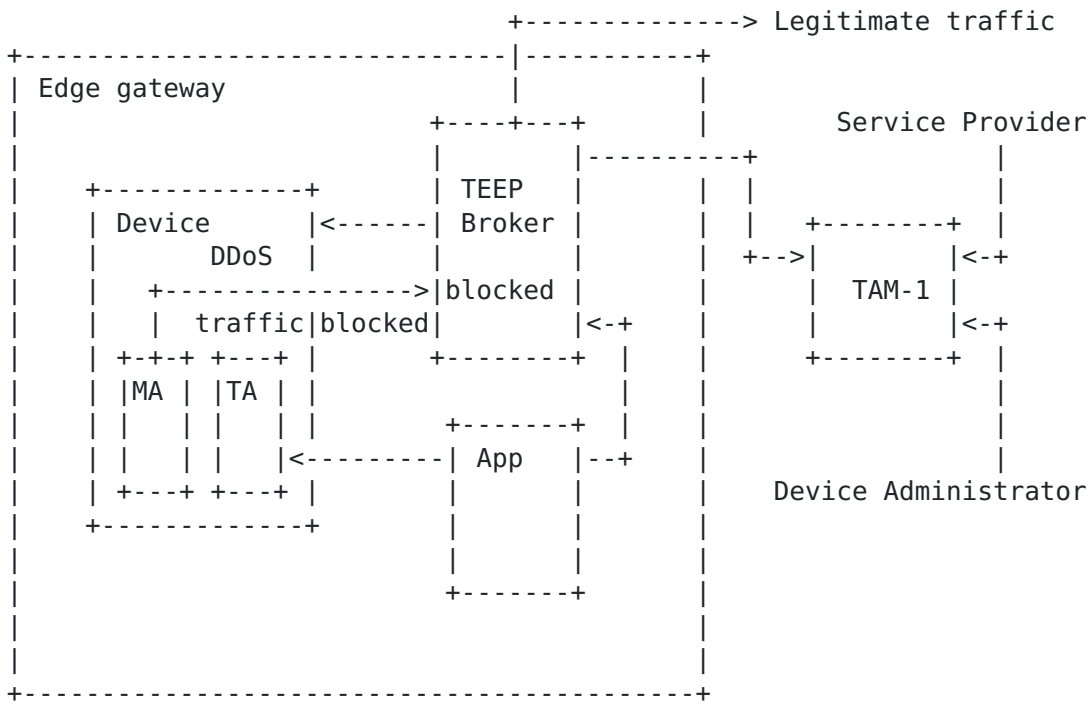

Figure 2: OS less IoT devices connected to gateway server

In this scenario the DDoS traffic is only generating network traffic inside the edge limits and can be stopped by the TEE inside the server. For example when an edge server is connected to home appliances such as home temperature control or electricity and water meters that are supposed to send time series to the cloud, triggering a DDoS will not be allowed to send packets outside the gateway limits.

It can still prevent sending the sensing data to the cloud destination but TEEP will prevent DDoS traffic outside the edge server. Additional to this using TEEP will prevent code upgrades done from untrusted sources and even detect malicious code to install on the device. In this configuration (Figure 2) SPs do not directly interact with devices. DAs may elect to use a TAM for remote administration of TAs instead of managing each device directly. Moreover the Legitimate traffic can be sanitized to prevent malicious code spread to other devices.

4.3 Smart IoT devices with full OS

The Internet of Things (IoT) has been posing threats to networks and national infrastructures because of existing weak security in devices. But there are IoT devices and systems that have the OS and compute power to detect and prevent malware for generation DDoS attacks. It is possible that for such devices can implement measures to prevent malware from manipulating actuators (e.g., IoT controlling computer assisted automobiles or self driving cars), or forcing such cars into accidents and damage infrastructures and even lose life.

Such an experiment was done in the research communities and there was even a contest about how fast hackers can take control of a car using automatic driving. The results were that the current security of such cars is not strong enough to prevent taking control over the internet. A TEE can be the best way to implement such IoT security functions for "smart" environments using advanced OSes such as cars.

TEEs could be used to store variety of sensitive data for IoT devices. For example, a TEE could be used in smart cars to store a driver's biometric information for identification, available in some new cars, and for protecting access driving wheel control mechanism. Figure 3 presents the architecture of such a self driving car. In this usecase the applications run inside the TEE and are connected to the service provider's cloud similar to some "connected" cars (BMW for example). The applications running inside the TEE can be either monitoring functions or car status (TA1) or diagnostic malfunctions (TA2). All these applications can be vital to the operation of the car and the safety of the drivers and roads. In general in this usecase the Service provider and the Device Administrator are represented by the vehicle manufacturer during the warranty time and after that they can be a different service provider doing maintenance.

There are additional usecases similar to this one like electric power and grid monitoring and control that have rich compute and memory resources running in a centralized location (secure) or in the cloud (unsecure) but need high levels of security.

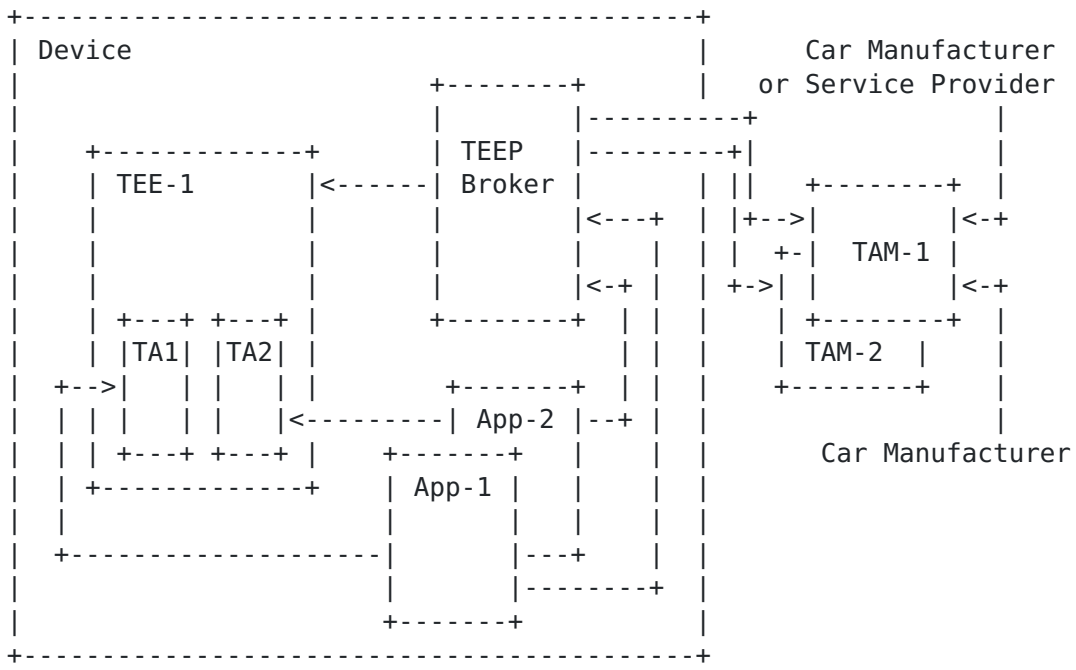

Figure 3: OS capable IoT devices for connected cars

There are additional security models of IoT devices that can fit in these 3 examples and we will extend the protocols to apply to as many as we can consider as useful.

5. Security Considerations

Although TEEP architecture document [I-D.ietf-teep-architecture] addresses some IoT devices examples there are IoT usecases that require more detailed design and better definitions of the Broker behavior in different usecases discussed in this draft. As such, Broker implementations MUST support many of this usecases critical for security and safety.

6. IANA Considerations

This document does not require actions by IANA.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8520] Lear, E., Droms, R. and Romascanu, D., "Manufacturer Usage Description Specification", March 2019, <https://www.rfc-editor.org/rfc/rfc8520.txt>

7.2. Informative References

[1] Vlajic, N., and Zhou, D., "IoT as a Land of Opportunity for DDoS Hackers", Computer Magazine, July 2018, pp. 26-34.

[2] Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J., IP Spoofing In and Out of the Public Cloud: From Policy to Practice,

[3] MUD files for testing DDoS vulnerabilities of IoT devices, https://iotanalytics.unsw.edu.au/mudprofiles/

[4] Vlajic N., Chowdhury M., and Marin Litoiu, "IP Spoofing In and Out of the Public Cloud: From Policy to Practice", Computers 2019, 8(4), 81; https://doi.org/10.3390/computers8040081

[5] IoT devices scanner for DDoS vulnerabilities test tool, python code, https://github.com/mashrufkabir/IoTScanner

[I-D.ietf-teep-architecture] Pei, M., Tschofenig, H., Wheeler, D., Atyeo, A., and D. Liu, "Trusted Execution Environment Provisioning (TEEP) Architecture", draft-ietf-teep-architecture-10 (work in progress), June 2020.

Acknowledgments

This draft has attempted to capture many IoT security usecases known to the author and presented in the literature as well as discussed in the security forums. These usecases present challenges both for DDoS attacks that became critical as well as applied security for new autonomous devices. We proposed to add these usecases to the TEEP Architecture draft.

Author's Address

Sorin Faibish
Dell EMC
228 South Street
Hopkinton, MA 01774
United States of America

Phone: +1 508-249-5745
Email: faibish.sorin@dell.com

Mashruf Kabir Chowdhury
Deloitte Canada
Apt 2112 - 70 Temperance St
Toronto, Ontario M5H 0B1
Canada

Phone: +1-416-388-5146
Email: mashrufkabir@icloud.com