Mon(IoT)r Testbed Manual

Last update: April 14, 2019

1. Overview

The purpose of this document is to give an overview of the experimental IoT testbed we have developed for the Mon(IoT)r IoT Lab. The testbed has been designed with the following requirements in mind:

- Provide wired and wireless network connectivity to IoT devices and their companion devices.
- Automatically collect traffic from devices connected to the testbed connectivity.
- Automatically try to perform man-in-the-middle of TLS connections, depending on the TLS interception policy. For example, if the auto-exception policy is used: if interception is possible, the decrypted data is collected, if not possible, disable the man-in-the-middle for devices and destination domains that do not allow it.
- Provide an arbitrary number of different independent connectivities that guarantee no interference among them. This allows researchers to run multiple independent experiments at the same time. In our specific setup, we use three connectivities.
- Automatically organize the data by device and automatically tag it by MAC Address, IP address, and a customized device name.
- Allow the researchers to mark experiments with arbitrary keywords to help relating collected traffic to what caused such traffic. This simplifies the organization of experiments.

2. Software components

Our testbed is based on the following software components:

- Operating System: Ubuntu Linux Server
- Mitmproxy man-in-the-middle software version 4.
- ISC DHCP server to assign IP addresses and notify dhcp events to our scripts.
- Tcpdump to collect IP and Ethernet network traffic.
- Netfilter/iptables, to redirect network traffic for man-in-the-middle purposes.

And the following components developed by us:

- A patch for Mitmproxy 4 for the automatic addition of exceptions when the man-in-the-middle fails, depending on the TLS policy used.
- Scripts in "**/opt/moniotr/bin**":

    - **extract-mitm**: extract MITM traffic that has been generated at a determined time. Useful to get the traffic generated by
    - **mitm-ctrl**: enable/disable the MITM (for both http and https traffic) capability of the testbed
    - **mitm-exception**: add/remove mitm exceptions and to set a TLS interception policy (i.e., when TLS exceptions are added, and how they are used).
    - **moniotr-ctrl**: enable/disable all the components of the testbed. This is a wrapper script for mitm-ctrl, networking-ctrl, and tcpdump-ctrl.
    - **moniotr-info**: get information about the monitored devices. For example, to check whether they are connected or not, or the last time they were connected.
    - **networking-ctrl**: enable/disable the networking component of the testbed. This includes configuring network interfaces such as traffic copy.
    - **setup-device**: this script gets executed when a new device is connected to the network and sets up the network collection.
    - **tag-experiment**: start/stop/cancel tagged experiments.
    - **tcpdump-ctrl**: enables/disables traffic capture using tcpdump

The directory organization and important files are the follows:

- **/opt/moniotr**: main directory used by the testbed.
- **/opt/moniotr/etc/moniotr.conf**: testbed configuration. Information about the configuration options is explained in the file. An example file can be found in "**moniotr.conf.example**" in the same directory.
- **/opt/moniotr/etc/devices.txt (from now on /traffic/devices.txt)**: contains a mapping between a MAC Address and the name of the device. This file is used by other components of the testbed to relate mac addresses to names. This file is managed manually. Names can only contain alphanumeric characters and "-".
- **/opt/moniotr/bin**: user invokable scripts.
- **/opt/moniotr/lib**: support scripts which are not supposed to be used by the users.
- **/opt/moniotr/log**: log files. Each log file has the name of the testbed subcomponent (setup-device.log, for connecting devices, mitmproxy.log, for the man-in-the-middle).
- **/opt/moniotr/tmp**: temporary files.
- **/opt/moniotr/examples**: example configuration files for external components.
- **/opt/moniotr/mitmproxy**: certificates for mitmproxy (initially empty, they are generated the first time it is started).

- **/opt/moniotr/traffic** (from now on, simply /traffic): all traffic and information captured by the testbed

3. Tested hardware components

The system we use to run the Mon(IoT)r testbed is composed of the following hardware components (other hardware configurations may work as well but have not been tested):

- **Analysis server**: x86_64 physical machine with Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz and 16GB of RAM. The machine has five Intel gigabit ethernet ports and three Qualcomm Atheros 9K Wi-Fi adapters, each one supporting three additional virtual Wi-Fi interfaces.
- **Smart switch**: Netgear JGS524PE 24-port managed PoE gigabit ethernet switch.

The analysis server is responsible for providing Wi-Fi and ethernet connectivity, for collecting and saving the network traffic, for assigning IP addresses, for routing the traffic through the Internet, for managing the different networks, for performing man-in-the-middle, for offering the tools for experiment management and data analysis.

The smart switch is used to increase the number of wired ethernet client that can be connected to the wireless router at the same time.

4. Network configuration

The network is assumed to be configured as follows. The server has the following physical network interfaces:

- **eth0**: network interface connected to WAN (the Internet)
- **eth-switch**: network interface connected to port 22 (VLAN9/VLAN10/VLAN11/VLAN12) of the smart switch. This interface hosts the following virtual interfaces: **switch-vlan10**, **switch-vlan11**, **switch-vlan12**
- **eth-mirror**: network interface connected to port 24 (MIRRORING) of the smart switch. This interface hosts the following virtual interfaces: **mirror-vlan10**, **mirror-vlan11**, **mirror-vlan12**
- **eth-copy**: network interface connected that contains a copy of all the traffic, which copies all the traffic from every device. This interface hosts the following virtual interfaces: **copy-vlan10**, **copy-vlan11**, **copy-vlan12**
- **wlan0**: Wi-Fi adapters on 2.4GHz network. This interface hosts the following virtual Wi-Fi interfaces: **wlan0.1**, **wlan0.2**, **wlan0.3**

- **wlan1**: Wi-Fi adapters on 5GHz network. This interface hosts the following virtual Wi-Fi interfaces: **wlan1.1**, **wlan1.2**, **wlan1.3**
- **wlan2**: Wi-Fi adapters on 2.4GHz network. This interface hosts the following virtual Wi-Fi interfaces: **wlan2.1**, **wlan2.2**, **wlan2.3**

All the interfaces above have the following purpose:

- **eth0** (untagged): network interface used for Internet traffic. This network is used to exchange traffic from/to the Internet. We call this network **WAN**. The current WAN connection has full outgoing connectivity to the Internet through a NAT. A firewall will block incoming connections.
- **eth-switch** (untagged): network used for administration traffic. This network allows researchers to connect to the server without being subject to traffic collection or being part of an experiment. We call this network **ADM** (192.168.9.0/24).
- **switch-vlan10** (tagged as VLAN10): network used for uncontrolled experiments traffic. This network collects the traffic from devices connected to it. We call this network **UNCTRL** (192.168.10.0/24).
- **switch-vlan11** (tagged as VLAN11): network used for controlled experiments traffic. This network collects the traffic from devices connected to it in an environment isolated from the other controlled or uncontrolled experiments networks. We call this network **CTRL1** (192.168.11.0/24).
- **switch-vlan12** (tagged as VLAN12): network used for controlled experiments traffic. This network collects the traffic from devices connected to it in an environment isolated from the other controlled or uncontrolled experiments networks. We call this network **CTRL2** (192.168.12.0/24).
- **eth-mirror** (untagged): network used for the administration (ADM) traffic mirrored by the smart switch. This interface can only be used to receive data.
- **mirror-vlan10** (tagged as VLAN10): network used for the uncontrolled (UNCTRL) traffic mirrored by the smart switch. This interface can only be used to receive data.
- **mirror-vlan11** (tagged as VLAN11): network used for the controlled (CTRL1) traffic mirrored by the smart switch. This interface can only be used to receive data.
- **mirror-vlan12** (tagged as VLAN12): network used for the controlled (CTRL2) traffic mirrored by the smart switch. This interface can only be used to receive data.
- **eth-copy** (untagged): network used for the administration (ADM) traffic copied to be analyzed by an external analyzer. This interface can only be used to send data.

- **copy-vlan10** (tagged as VLAN10): network used for the uncontrolled (UNCTRL) traffic copied to be analyzed by an external analyzer. This interface can only be used to send data.
- **copy-vlan11** (tagged as VLAN11): network used for the controlled (CTRL1) traffic copied to be analyzed by an external analyzer. This interface can only be used to send data.
- **copy-vlan12** (tagged as VLAN12): network used for the controlled (CTRL2) traffic copied to be analyzed by an external analyzer. This interface can only be used to send data.
- **wlan0**, **wlan1**, **wlan2**: these Wi-Fi networks (essid: Mon(IoT)r_adm), running on the 2.4GHz frequency, are used for administration (ADM) traffic.
- **wlan0.1**, **wlan1.1**, **wlan2.1**: these Wi-Fi networks (essid: Mon(IoT)r_adm), running on the 2.4GHz frequency, are used for uncontrolled (UNCTRL) traffic.
- **wlan0.2**, **wlan1.2**, **wlan2.2**: these Wi-Fi networks (essid: Mon(IoT)r_adm), running on the 5GHz frequency, are used for controlled (CTRL1) traffic.
- **wlan0.3**, **wlan1.3**, **wlan2.3**: these Wi-Fi networks (essid: Mon(IoT)r_adm), running on another 2.4GHz frequency, are used for controlled (CTRL2) traffic.

The various interfaces are bridged together within the server in the following way:

- **br9**: bridge used by the administration network (ADM). Its interfaces are: **eth-switch**, **wlan0**, **wlan1**, **wlan2**.
- **br10**: bridge used by the uncontrolled network (UNCTRL). Its interfaces are: **switch-vlan10**, **wlan0.1**, **wlan1.1**, **wlan2.1**.
- **br11**: bridge used by the controlled network (CTRL1). Its interfaces are: **switch-vlan11**, **wlan0.2**, **wlan1.2**, **wlan2.2**.
- **br12**: bridge used by the uncontrolled network (CTRL2). Its interfaces are: **switch-vlan12**, **wlan0.3**, **wlan1.3**, **wlan2.3**.

Finally, the smart switch has 24 ports, named from S1 to S24, that are configured as follows:

- **S1, S3, S5, S7, S9, S11**: may be connected to devices under uncontrolled experiments. These ports are bridged with UNCTRL (VLAN10) and support *power over ethernet*.
- **S13, S15, S17, S19, S21, S23**: may be connected to devices under uncontrolled experiments. These ports are bridged with UNCTRL (VLAN10), but do not support power over ethernet.
- **S2, S4, S6**: may be connected to devices under controlled experiments 1. These ports are bridged with CTRL1 (VLAN11) and support power over ethernet.

- **S0, S10, S12**: may be connected to devices under controlled experiments 2. These ports are bridged with CTRL2 (VLAN12) and support power over ethernet.
- **S14, S16, S18, S20, S22**: These ports do not support power over ethernet and are configured as follows:
    - Untagged VLAN is bridged with the ADM (VLAN9) network.
    - Tagged VLAN10 is bridged to the UNCTRL network.
    - Tagged VLAN11 is bridged to the CTRL1 network.
    - Tagged VLAN12 is bridged to the CTRL2 network.

  Any of these port (for example, port S22) must be connected to the physical port named **eth-switch** of the server.
- **S24**: this port is the mirroring port of the smart router, it is unidirectional, and its outgoing traffic consist of a copy of all the traffic transiting the switch from all the other ports. VLANs are preserved in traffic mirroring. This port is physically connected to the port **eth-mirror** of the server.

5. Experiments management

The default testbed configuration provides three networks. One for each different type of experiment:

- *UNCTRL*: **generic uncontrolled experiments** network. This is the default choice for connecting a device to the testbed. All devices connected to the uncontrolled experiment networks will be able to see each other and to interact. This network should be used when there are no requirements of isolation when testing the device.
- *CTRL1*: **first controlled experiments network**. This network is used when testing a device in an isolated environment, or with a controlled set of devices. Tests in this network guarantee no interference from the devices on the uncontrolled experiment network. This network is designed to be used only for a limited amount of time, and should not be used at the same time for more experiments, unless it is known a priori that there are no risks of interference.
- *CTRL2*: **second controlled experiments network**. This network is used for controlled experiments in the same way as CTRL1, when CTRL1 cannot be used because it is already being used and it is a risk of interference.

The testbed also allows to *tag experiments* using the **tag-experiment** script, which is available to every user in the system. Tagging is supported by all experiment types (UNCTRL, CTRL1, CTRL2) and allows users to put all the data generated during an experiment in its tagged directory in **/home/traffic-tag**. By using the script on a particular device, a researcher can start and stop a tagged experiment. Once a tagged experiment is stopped, all the data related to such experiment is created in its tagged directory. The tag-experiment script allows to tag a single device with or without a

companion device (to capture the interactions among two devices that are related, for example a phone with a companion app used to control an IoT device). Multiple tagged experiments can be run at the same time and on different networks. A guide that explains how to use the script is available by running the script without arguments. Additional information on what data is generated and where it is stored is normally printed after an experiment is stopped.
Additional information on data generated by each type of experiment is reported in the next section.

6. Collecting experiment results

The results of all the uncontrolled experiments can be found in: **/traffic/by_mac**, **/traffic/by_ip**, **/traffic/by_name**, **/traffic/by_all**. The difference among these directories is that:

- **/traffic/by_home** uses MAC Address as device id
- **/traffic/by_ip** uses the last IP address as device id
- **/traffic/by_name** uses the device name (as reported in **devices.txt**) as device id.
- **/traffic/by_all** can use either MAC, last IP, and the device name (as reported in **devices.txt**) as device id.

Each of the above directories contain a directory named with the device id.
Each device id directory contains several text files and three subdirectories:

- **ip.txt**: contains the last IP addresses of the device.
- **name.txt**: contains the name of the device.
- **mac.txt**: contains the MAC address of the device.
- **monitor-if.txt**: contains the last network interface the device was/is connected to
- **mitm-exceptions.txt**: contains the list of exceptions (in terms of ip addresses and domain names) from the MITM.
- **dhcp-info.txt**: contains the device description given by the device to the DHCP server.
- **unctrl**: subdirectory containing the data of uncontrolled experiments (i.e., from devices connected to a network bridged with UNCTRL)
- **ctrl1**: subdirectory containing the data of controlled experiments 1 (i.e., from devices connected to a network bridged with CTRL1)
- **ctrl2**: subdirectory containing the data of controlled experiments 2 (i.e., from devices connected to a network bridged with CTRL2)

Such subdirectories contain the following files:

- **Pcap files named XXX-YYY.pcap**, where XXX is the date the file was created or rotated and YYY is the IP address of the device. These *pcap* files contain all the

Ethernet and IP traffic that the device has exchanged with other devices or over the Internet. These files are created when a device connects the first time after a restart of the analysis server, and they are rotated every 24 hours or after the file size reaches 128MB, whichever happens first.

- **http-XXX.log** and **https-XXX.log**, where XXX is the date the file was created or rotated. These files contain respectively the outgoing http requests headers sent in cleartext, and https requests headers that have been successfully intercepted by the MITM. The format of the file is the one used by the software Bro and they are rotated at midnight (Eastern Time).

Traffic from controlled experiments can be found in **/traffic/tagged**. This directory contains a list of device names that have been used to perform tagged experiments. Each device directory contains a list of directories named after each tagged experiment. Finally, each tagged experiment directory contains Pcap and http(s).log files of the device under test and of his possible companion device. If a tagged experiment is run more than once, the resulting pcap and http(s) files are created in the same directory. Since these files contain a timestamp in their name, and the fact that a device cannot have two tagged experiments with the same tag running at the same time, it is impossible to have duplicated files or filename collisions in the tagged experiment directories.

Finally, the directory **/traffic/mitm** contains the traffic captured by mitmproxy in raw format, organized by date.

7. Installing the testbed

1. Extract the content of the “moniotr” directory to any location, for example on /opt/moniotr, then enter the directory:
   ```
   cd /opt/moniotr
   ```

2. Install python3.6 (or later) ISC DHCP server and tcpdump.
   If using Ubuntu Linux:
   ```
   apt install python3 python3-pip isc-dhcp-server tcpdump
   ```

3. Install the following python dependencies (please do NOT use a different version of mitmproxy):

```
pip3 install mitmproxy==4.0.4
```

4. Apply the Mon(IoT)r modifications to mitmproxy:
   ```
   cp lib/server.py /usr/local/lib/python3.6/dist-packages/mitmproxy/proxy/
   ```

5. Configure app-armor in such a way that dhcpd is excluded from its checks:
   mkdir /etc/apparmor.d/disable
   ```
   ln -s /etc/apparmor.d/usr.sbin.dhcpd /etc/apparmor.d/disable/usr.sbin.dhcpd
   service apparmor restart
   ```

6. Configure netplan and isc-dhcp-server for the network interfaces that will host the monitored devices.
   For each monitored network, add to dhcpd.conf the following lines inside the "subnet" block:

   ```
   on commit {
       set clip = binary-to-ascii(10, 8, ".", leased-address);
       set clhw = binary-to-ascii(16, 8, ":", substring(hardware, 1, 6));
       execute("/opt/moniotr/bin/setup-client", clip, clhw);
   }
   ```

   If Mon(IoT)r was not installed in /opt/moniotr, replace /opt/moniotr with the installation directory.

   Finally, restart isc-dhcp-server:
   ```
   service isc-dhcp-server restart
   ```

7. Copy etc/moniotr.conf.example to etc/moniotr.conf and edit its values according to the local configuration of the system. Information about the meaning of each configuration parameter is explained inside the file itself.

8. You can start Mon(IoT)r using:
   ```
   /opt/moniotr/bin/moniotr-ctrl start
   ```

   The same command can be added to /etc/rc.local to activate Mon(IoT)r at startup.
   If Mon(IoT)r was not installed in /opt/moniotr, replace /opt/moniotr with the installation directory.

You can find example configuration files for netplan, hostapd (in case you use Wi-Fi access point), isc-dhcp-server, iptables-save, and sysctl in the examples directory.

All Mon(IoT)r scripts are in the /opt/moniotr/bin directory. To know the usage of each script, execute the script with “help” as the only parameter. The usage will explain all configuration options and some examples of use.

Please do not edit moniotr.conf while moniotr is started, to avoid problems.