Transferring and Updating Individual Package for Linux-based IoT Devices by Calculating Connection Speed

Hoai-Nam Nguyen[1†] **and Truong-Thang Nguyen**[2††]

nguyenhoainam@ioit.ac.vn ntthang@ioit.ac.vn

Thu-Nga Nguyen Thi [3††]**, Manh-Dong Tran** [4††]**, Ba-Hung Tran**[5††]

nttnga@ioit.ac.vn dongtm@ioit.ac.vn tbhung@ioit.ac.vn

Institute of Information Technology, Vietnam Academy of Science and Technology, Hanoi, Vietnam

Summary
The updating IoT device issue is one of the hot topics in the IoT development field in recent years. Up to 2020, one of the popular solutions is updating the whole operating system to reduce the risk of failure and recover the device faster. However, that solution is not always suitable, especially in the case of low connection speed and limited storage. For that reason, in this paper, we would like to suggest another solution that allows uploading and updating an individual package for Linux-based IoT devices by calculating connection speed.

Keywords:
IoT, updating, automation, network, Linux.

1. Introduction

1.1. The importance of updating IoT devices

According to Security Today [1], IoT refers to connected physical and digital components. Each IoT component has a Unique Identifier (UID) that makes it recognizable. The number of IoT devices is increased every year. In 2018, there were 7 billion IoT devices. In 2019, the number increased to 26.66 billion. In 2020, the estimated number is 31 billion. Every second there are 127 new IoT devices connected to the Internet.

In 2020, there are 10 IoT risks to watch out as the following:

- Weak password: In a lot of cases the users forgot to change the default simple password.
- Insecure or unneeded network services: Those services are installed on the device and may expose sensitive data.
- Insecure ecosystem interfaces: External interfaces connecting to the device may cause the devices to be compromised.
- Use of insecure or outdated components: Those components that were not scanned for vulnerabilities make it easier for the hacker to access the devices.
- Insufficient privacy protection: Private information that is stored in the device is not protected.
- Insecure data transfer and storage: Lack of access control and encryption while the data is transferred and stored.
- Lack of device management: The devices are not managed remotely which leads to a lack of updates.
- Insecure default settings: The default settings are not changed that make the devices easy to be exploited.
- Lack of physical hardening: That issue makes the hacker easier to access and take control of the devices.

One of the solutions for those 10 above risks is keeping the IoT devices up to date. However, manually updating IoT a large number of IoT devices takes a lot of time and has many potential risks. That issue led to a demand for producing solutions to update automatically.

1.2. Some existing solutions

Up to now, many solutions that can be categorized into two groups:

Commercial solutions: Most of the commercial solutions are made to serve the devices of a manufacturer. The advantage of that is the solution are optimized because the manufacturers understand most about the characteristics of the devices. The disadvantage of that is it cannot apply to a group of different brands of devices. However, some of the solutions became open and expanded to serve a wide range of devices.

Some of the commercial solutions:

- mender.io [2]: Mender allows the developers to mitigate risks at a large scale based on a predefined update deployment plan. The advantages of this solution are: (i) Provide source code on GitHub; (ii) Easy to use UI;

Manuscript received December 5, 2024
Manuscript revised December 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.12.4

(iii) Provide package for updating. The disadvantage of this solution is not all the updates support the roll-back strategy.

- balena.io [3]: The previous name is Resin. Balena is a complete tool that helps the developers build, implement and manage connected devices using Linux. The advantage of Balena is it supports small size update to reduce the risk of power. The disadvantage of Balena is the update contains the whole operating system that might not be suitable for unstable network connection.

Open-source solution: The advantage is they serve a lot of brands, however, the disadvantages are the optimization and a complete solution. Most of them only have some functions for specific purposes.

Some of the open-source solutions:

- Ansible [4]: Ansible is a Python-based solution that helps the developer build the automation tasks. The advantages of Ansible are: (i) Agentless architecture reduces the maintainance tasks on the devices because there is no client application installed on the devices; (ii) Connection using SSH allows Ansible to connect to a large type of device. The disadvantage of Ansible is there is no direct function for updating the device. For that reason, the developers have to write themselves the updating part.

- eNMS [5]: eNMS is an enterprise vendor-agnostic network automation platform. The advantage of the solution is it has the pre-defined flows to manage the devices. The disadvantage of the solution is it does not contain the updating flow, it only contains the backup flow. Therefore the developers have to define or build their updating flow.

1.3. Updating IoT device issues

All the solutions have to meet the following requirements [6]:

- Secure: Ensure the update is made via an encrypted connection.
- Authentication: Only suitable roles have the right to execute the update.
- Robust: Ensure the devices behave normally after the update.
- Atomic: The update is either installed completely or not at all.
- Fail-safe: The update has a roll-back strategy.
- Monitoring: The update status has to be monitored and logged.

Besides, transferring and updating solutions for IoT devices have to face the following issues:

1. The updating process might be disturb when the network has problems.

2. The transferred data is too large that might leads to the updating process to retry many times until the transferred data is completed.

3. The developers have to customize the updating module for IoT devices with the situation that not all the organizations have suitable skillful developers.

4. The lack of a validating and evaluation process to make sure the update is completed in the right way and to make sure the devices can be recovered in failure cases.

With the above issues, we focus on solving issue (1) and issue (2) that ensure the update package is transferred completely in an unstable network and reduce the retry times.

2. Technical Concept

The proposed transferring and updating process is as the following:

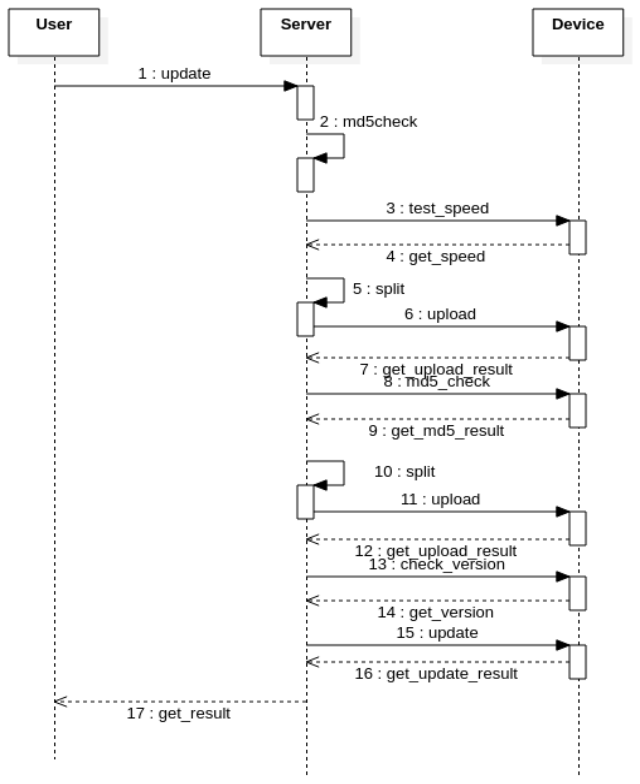


Fig 1. Updating process

Step 1: Prepare the updating file and generate MD5 hash of the file to validate after the file is transferred.

Step 2: Transfer the updating file to each device.

- Evaluating the connection speed.
- Split the file if required to have the transfer time less than the timeout value (normally 30 seconds).

Step 3: Perform the loop:

- If the transfer is successful, validate the file via MD5 checksum.
- If the transfer is unsuccessful, split the file into smaller pieces.

Step 4: Validate the current version of the installed package.

Step 5: Update the package.

Step 6:

- If the update is completed successfully, inform the user.
- If the update fails, restore the previous package version.

3. Implementation

In this paper, we use Spatie SSH, a Laravel package to connect to the remote device. Spatie SSH is a package based on Symfony's Process. The default syntax to connect to a device is:

Ssh::create('user', 'host')->execute('command');

The advantage of this package is it can run asynchronously so we can inject the sub-sequence actions after any command is executed. In addition, the package supports copying files from a localhost to a remote host without using a direct "scp" command in the "execute" function.

Below is a custom private function that we use for general SSH commands in the mentioned steps to upload and update a package.

```
private function execute_ssh_command($device,
$commands){
        return Ssh::create($device->username,
$device->host)
                                        -
>usePort($device->port)
                                        -
>disableStrictHostKeyChecking()
                                        -
>execute($commands);
}
```

3.1 Checking MD5 hash of the upload file

```
$commands = ['md5sum ' .
env('REMOTE_UPLOAD_PATH') . $device-
>username . '/' . str_replace("uploads/", "",
$task->file)];
Log::debug('Commands' . implode(",",
$commands));
$remoteChecksumResult = explode(' ', $this-
>execute_ssh_command($device, $commands)-
>getOutput());
$checksumResult = $remoteChecksumResult[0];
Log::info('MD5 checksum: ' . $checksumResult);
```

3.2. Check the current version of the package on remote server

```
$commands = [$device->package_name . '
version'];
Log::debug('Commands' . implode(",",
$commands));
$version = $this->execute_ssh_command($device,
$commands)->getOutput();
Log::info('Current Version: ' . $version);
```

3.3. Update the package on a remote device with the condition that the uploaded file has the same MD5 hash as the file on the server.

```
if (md5_file($task->file) == $checksumResult)
{
$commands = ['sudo apt install ./' .
str_replace("uploads/", "", $task->file)];
$update = $this->execute_ssh_command($device,
$commands)->getOutput();
Log::info('Update result: ' . $update);
}
```

4. Evaluation

To evaluate the concept, we have built a lab using GNS-3 (Graphical Network Simulator-3) with the endpoint is the IoT device.

Below is the network model created by using GNS3. For demonstration purposes, we only create 1 server (for transferring the update file) and 4 IoT devices. The number of IoT devices in real situations will be much higher than that, however, the concept is using "worker" function of Laravel, a PHP framework that allows the upgrade tasks can be scheduled and run asynchronously without waiting for one by one to be finished. However, there is a pre-defined number of failed updates that the "worker" should be stopped and run the roll-back step.

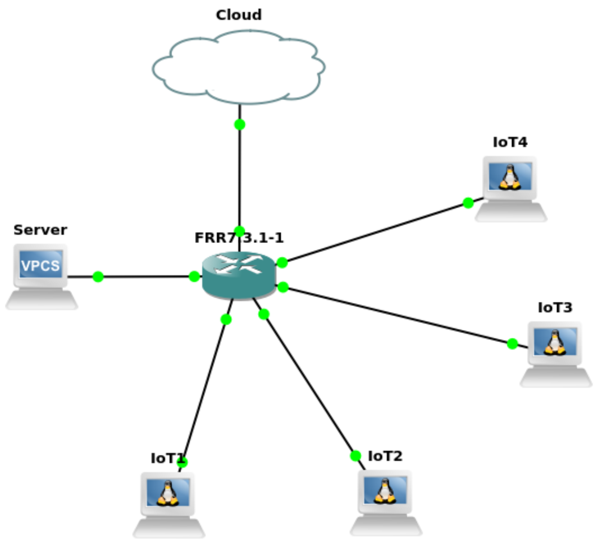
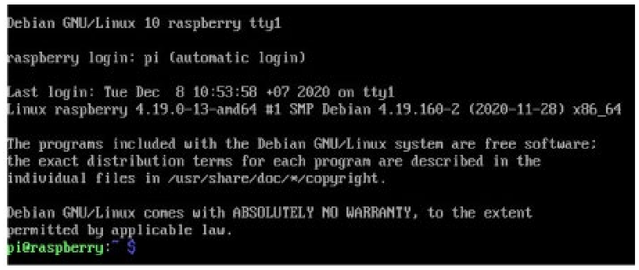


Fig 2. GNS-2 device model

The operating system used for the IoT devices is Raspberry Pi OS (the previous name is Raspbian). This operating system is optimized for ARM-based chipset so that it can run in a limited capacity device.



```
Debian GNU/Linux 10 raspberry tty1

raspberry login: pi (automatic login)

Last login: Tue Dec  8 10:53:58 +07 2020 on tty1
Linux raspberry 4.19.0-13-amd64 #1 SMP Debian 4.19.160-2 (2020-11-28) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberry:~ $
```

Fig 3. Raspberry Pi OS Console

5. Conclusion

In this paper, by using the concept of "testing connection - splitting - uploading - evaluating", we solved the issue of updating the Linux-based IoT devices in a specific case of low speed and unstable network connection. In comparison to the method of updating the whole "image" of the operating system, updating the whole "image" can reduce the problem of failure by a faster recovery (roll-back) process. However, we can also clone the operating system and install into a different partition of the device, implement the updating process, configure the boot loader to use the new partition and do the roll-back step to switch back to the previous partition. That is our future step to optimize the roll-back step.

Acknowledgments

We would like to thank CS'20.10 project of the Institute of Information Technology, Vietnam Academy of Science and Technology that has provided funding for the study.

References

[1] The IoT Rundown For 2020: Stats, Risks, and Solutionshttps://securitytoday.com/articles/2020/01/13/the-iot-rundown-for-2020.aspx
[2] mender.io https://mender.io
[3] balena.io https://www.balena.io/
[4] Ansible https://www.ansible.com/
[5] eNMS https://github.com/eNMS-automation/eNMS
[6] Software update for IoT https://elinux.org/images/f/f5/Embedded_Systems_Software_Update_for_IoT.pdf
[7] OTA update solutions - The ultimate guide https://witekio.com/ota-update-solutions-the-ultimate-guide/
[8] Test ssh connection speed http://www.alecjacobson.com/weblog/?p=635
[9] Install Raspbian Desktop on a Virtual Machine https://roboticsbackend.com/install-raspbian-desktop-on-a-virtual-machine-virtualbox/
[10] GNS3 Lab http://internetoftheo.blogspot.com/2019/01/gns3-lab.html
[11] Laravel PHP Framework https://laravel.com/
[12] A lightweight package to execute commands over an SSH connection https://github.com/spatie/ssh

Hoai-Nam Nguyen received the BSc. in University of Engineering and Technology of Vietnam National University in 2009, MSc. in Hof University of Applied Science, Germany in 2015. Currently working at the Institute of Information Technology of Vietnam Academy of Science and Technology. Research fields: Cybersecurity, software engineering.

Truong - Thang Nguyen received a Ph.D. in 2005 at the Japan Advanced Institute of Science and Technology (JAIST), Japan. Currently working at the Institute of Information Technology, Vietnam Academy of Science and Technology. Research fields: software quality assurance, software verification, program analysis.

Thu-Nga Nguyen Thi received MSc. in University of Information and Communication of Thai Nguyen. Currently working at the Institute of Information Technology of Vietnam Academy of Science and Technology. Research fields: Cyber security, crypto.

Manh-Dong Tran received a M.S. degree in 2013 at the University of Engineering and Technology, Vietnam National University, Hanoi. Currently working at the Institute of Information Technology, Vietnam Academy of Science and Technology. Research fields: software quality assurance, software verification, program analysis.

Ba-Hung Tran receivved a Master degree in 2014 at Military Technical Academy working at the Institute of Information Technology. Research fields: Network Infrastructure, software verification, program analysis.