January 20, 2019
Magnus Bakke & Liang Zhu

Weekly report: Week 2 & 3 of dissertation for Uninett AS on authentication in IoT

by Magnus Bakke & Liang Zhu at Informatics with specialization in network administration at the Norwegian University of Science and Technology

Introduction

We (the authors) have been asked to write weekly reports in connection with our bachelor thesis. Because we began on January 9, a Wednesday, we have decided to write one report and one work schedule for week 2 and 3 of 2019. We will describe what was done on each day.

Activities

Wednesday January 9

A meeting was held and attended by Magnus and Liang (candidates), Stein Meisingseth (NTNU), Jørn Åne de Jong (Uninett), Tom Ivar Myren (Uninett), and Otto Wittner (Uninett). We discussed and agreed upon the scope of the project, which shall include these four stages:

1. Research on the topic
2. Planning of a solution for authentication of IoT devices in a network
3. Development of a prototype of a system for authentication of IoT devices
4. Evaluation of viability

We proceeded to write a work schedule for the two weeks in this period.

Thursday January 10

Following the work schedule, we began researching commonly used types of authentication mechanisms and encryption standards in depth in order to gain a better understanding of the protocols. We believe this understanding will be crucial for the success of the solution we propose.

We also brainstormed possible solutions in order to give ourselves a better sense of direction. Promising solutions involve dynamically generated pre-shared keys (PSK) in conjunction with a user interface and a backend (for example using Django REST Framework). This requires that the access points support forwarding of PSKs, which Cisco and Aruba APs do not. PSKs can be replaced with MAC address authentication, but MAC

addresses can be easily spoofed, making this a less secure solution. Uninett may have to decide whether to replace their APs for better security or accept the vulnerable MAC address authentication solution.

A less user-friendly solution involves using “leased” devices that are configured to securely communicate with the network using WPA2-Enterprise while acting as hotspots for IoT devices. This approach comes with certain limitations (including a lack of support for roaming, unless the hotspots are battery-powered and lightweight enough to carry along with the IoT devices), and results in additional work for the network’s administrators and may be considered a nuisance by the user.

Friday January 11

We decided to research one possible solution involving MAC address authentication. The second solution (involving using Raspberry Pi hotspots) and the third solution (which is currently considered the optimal solution and involves dynamically generated pre-shared keys) will be researched and discussed at a later date.

RADIUS servers can be configured to use MAC-based access control.[1] This solution is described as user unfriendly: When authentication fails, there is no notification or explanation given as to why the authentication failed. MAC address authentication also suffers from extremely poor security: MAC addresses are transmitted unencrypted over the network and are easy to spoof. Therefore, this solution should only be considered if the administrators of the network have full knowledge of each device’s capabilities and potential for abuse, something we consider unrealistic if there are many devices in the network. If the number of IoT devices is very limited, it may be possible to determine if there is any potential for damage should any of the devices be spoofed or otherwise compromised.

One solution that does not compromise security to this degree involves setting up a separate wireless network for IoT devices using WPA2 Personal. The administrator would choose a strong key for this network and configure IoT devices to authenticate using said key. In MAC address authentication, the device typically uses its MAC address as a username or password. Instead, all traffic on this network would be routed through a backend that checks each frame’s MAC source field against a list of approved MAC addresses. Any new device would have its MAC address added to this list by the administrator. If the MAC address is found, communication is allowed. Otherwise, it would be denied.

This solution requires knowledge of the secret key in addition to a valid MAC address. However, knowledge of this key (which could possibly be extracted from the hardware of poorly secured devices) grants an attacker the ability to impersonate any device connected to this network.

[1] Cisco. (2018, November 27). Enabling MAC based access control on an SSID. Retrieved January 14, 2019, from https://documentation.meraki.com/MR/Encryption_and_Authentication/Enabling_MAC_based_access_control_on_an_SSID

Finally, the solution does not support “plug and play” – the administrator must manually add the MAC address to the whitelist (through a GUI). Finding the MAC address of a device is not always easy, and may require analysis of packets using a packet sniffer.

Saturday January 12

Our plan was originally to move on to finding existing solutions today, but our research has revealed that few solutions exist. There is more research to do than we originally anticipated. Therefore, we spent Saturday researching the possibility of using Raspberry Pi machines as hotspots for IoT devices.

We cannot identify any security issues or technical issues when it comes to using Raspberry Pi machines as hotspots for IoT devices. There are a few practical limitations, however: Portability requires the use of compatible batteries; the machine is susceptible to accidents and possibly theft, and this solution will involve a significant amount of work for the person(s) managing and keeping track of the machines. Still, it is a viable temporary solution that can easily be implemented while a more flexible and convenient solution is implemented, as Raspberry Pi machines can easily be configured to act as hotspots.

Monday January 14

We spent the day working at home and communicating over Facebook. We researched the specifics of the MAC address authentication solution and identified the need for a programmable firewall – a firewall whose rules are not exclusively set by rigid policies, but by predicate functions (functions returning Boolean values) that accept the packet as an argument. This is called a software-defined firewall (SDF). This is needed because a packet will be allowed or rejected based on whether or not the MAC address is found in a database. This condition will be checked programmatically.

We have little experience with software-defined firewalls, and have not found any specific examples of software supporting this functionality. We may need to seek help from external sources, such as StackExchange. We did find examples of usage (using Python, which triggered a discussion of the viability of using the Django REST framework for a backend that combines the SDF and database services in one network component), but it was poorly explained in unintelligible English.

Tuesday January 15

We attended a workshop at the Radisson Blu hotel. In addition to gaining some insight into Uninett’s systems and possible future plans, representatives from Aruba Networks demonstrated the capabilities of ClearPass, their platform for policy management, and how it can simplify the process of onboarding IoT devices securely in a network. The most relevant part of their presentation regarded their *multi-pre-shared key* (mPSK) solution, which very much resembles the DPSK solution we researched as a possible (and optimal) solution. We were given the business cards of two Aruba representatives, who invited us to contact them

for more information and mentioned the possibility of providing us with a temporary license for ClearPass.

If mPSK offers the functionality we need (which includes dynamic generation of credentials upon request for display on the front end), we can do away with the MAC address authentication solution and the hotspot solution.

Wednesday January 16

We sent an email to Anders Lagerqvist (Senior Systems Engineer) and Tore Henriksen (Systems Engineer) at Aruba inquiring about the capabilities of ClearPass. We briefly outlined the assignment and asked if ClearPass supported generating PSKs on demand using a function/API endpoint. This is required by the solution we envision.

While waiting for a response, we began planning the hotspot solution. We discussed multiple possible approaches, but landed on a simple one:

1. **A Raspberry Pi is reset to an initial state**
   This initial state will include a web page that is displayed when the user connects to the Raspberry Pi using a web browser.
2. **The Raspberry Pi is handed over to a user upon request**
3. **The user connects to the Raspberry Pi**
4. **The user enters their Eduroam username and password**
5. **The Raspberry Pi connects to Eduroam using the provided credentials**
   If the connection was unsuccessful, an appropriate response is given.
6. **A password is generated**
   This password could be chosen by the user or generated automatically and displayed on the web page.
7. **The Raspberry Pi enables its hotspot**
8. **The user tells their IoT devices to connect to the hotspot using the hotspot password**
9. **Accountability is accomplished by the association to the Eduroam user**
10. **When the user is finished, the Raspberry Pi is returned and reset to its initial state**

The cost of this solution is at most 366 NOK per unit (the current cost excluding VAT). Bulk options may be available. The same result can be accomplished if the user simply configures their laptop as a hotspot, but this solution has the added benefit of longevity (it will not stop working if the laptop is closed, so long as the Raspberry Pi has power and is nearby).

The web server (running on the Raspberry Pi) must be capable of interacting with the file system, executing programs and/or changing system settings. Django is our primary candidate for accomplishing this.

This is still only a backup solution. If ClearPass does not offer the functionality we need for the mPSK solution, we may choose to proceed with this solution.

January 20, 2019
Magnus Bakke & Liang Zhu

We also briefly explored the MAC authentication solution, but decided that the compromise of security is unacceptable. The solution is capable of limiting access to the network for users who do not know how to spoof MAC addresses, but without multiple PSKs, there is no mechanism to prevent packet sniffing or man-in-the-middle attacks. The solution *may* be acceptable if the only devices that will be connected to the network are brought and configured by the network administrators, but it is a requirement that students can get their devices connected as well, which implies that the key must be shared publicly.

Friday January 18

Tore Henriksen from Aruba replied to our email saying that two versions of mPSK will be supported: 1) One where the administrator generates keys for groups of devices, such as all printers or all computers in a department, and 2) one that features a web portal that end users can use to generate a single key for their device.

The latter describes our proposed solution perfectly, though we would prefer a custom frontend for a tailored user experience and possibly extra features or requirements.

Henriksen went on to state that mPSK functionality requires version 6.8 of ClearPass, which is not yet available. For this reason, there is also little or no documentation. Henriksen has attempted to find out when version 6.8 can be expected, but has not been able to do so, though he suspects it will be available soon.

Because of the uncertainty of the release date of version 6.8 of ClearPass, we have decided to research other equivalent solutions, namely *dynamic-PSK* by Ruckus or *private-PSK* by AeroHive. We will research these next week.

Remarks

Because of continued uncertainty regarding which solution we will focus on, we have not been able to write a work schedule for the entire product period. We will likely begin this process after the meeting on Wednesday.