BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2019
http://excel.fit.vutbr.cz

Automation of MiTM attacks with use of microservices architecture and singleboard computers

Šimon Podlesný*

Abstract
This article is focused on design of MiTM attack with use of modern approaches in IT infrastructure. Especially it's focused on how to simplify configuration of single-board computer for penetration testing purposes by creating scalable infrastructure for device configuration and control. It's also trying to solve problem of disjunction between physical and network security, where professionals for physical security lack technical skills needed for network attacks and vice versa. Proposed solution allows the usage of complicated attacks by trained staff while not limiting users with experience in network security. While today, applications capable of MiTM attacks are monolithic and device-centric, proposed solution considers the device providing MiTM just as one part of the solution and also focuses on other problems like data exfiltration or hash cracking. This was made possible mostly by using Docker, Ansible and Python.

Keywords: MiTM, Ansible, Docker, Raspberry Pi, eaphammer

Supplementary Material: Source Code for: Configurator , Hash storage server

*spodlesny@gmail.com, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Network security is a hot topic. During last few years, companies and states spent large sums of money for defensive and offensive measures in cyberspace. While IDS/IPS systems and firewalls on gateways to the Internet are becoming standards in corporate world, entry-points like Wi-Fi have been mostly forgotten, relying on old security standards that do not take into account modern world and technologies (like credit card sized computers). Since penetration testing is about finding the weakest link, I decided to demonstrate how easy it could be to use devices like Raspberry Pi for penetration testing, while presenting a new approach to penetration testing.

2. Current state

MiTM attacks have been here since the first wired and wireless transmissions were made. As people were finding new ways to eavesdrop communication, new ways how to protect it were created. Today, and for past few decades, MiTM attacks have been focused mostly on Ethernet and Wi-Fi networks.

Most common approach to this type of attacks is that attacker connects to network with a laptop and performs a MiTM attack on the spot. Another common approach is to install malware on a local computer and use this device to do the attack.

While these approaches are relatively well-proven, sufficiently long physical access to network is not al-

ways possible and accessing a computer on local network through the Internet is a difficult task, since most LAN networks are behind NAT.

For the last few years, there is a new trend of using single-board computers for this task. They are small, relatively cheap and can be powered by battery or USB port. The biggest problem of these single-board computers is their real world use, since most applications for penetration testing were not designed to be used without user input in any way.

It's also problematic to verify that configuration works as it was intended due to missing display. Another problem is extraction of data from device if physical access is no longer possible. Last issue that should be also taken into account is how to replicate the configuration process for each new device, so the pentester doesn't have to start with vanilla image of operating system every time a new device is needed. Overall, configuration is a time-consuming, tedious process that can be automated to a degree where connecting a device to keyboard, monitor, and mouse is no longer necessary.

Proposed solution attempts to take care of all of these problems. It does not try to recreate wheel for penetration testing, but it simply shows a better way how to utilize existing tools, while taking into account strengths and weaknesses of single-board computers. One of the goals of this project is to make it more user friendly, so people with other backgrounds (like physical security) can also use it. I believe that this would be a welcomed change, and single-board computers will be utilized in the field similarly as audio/video devices are being used today.

3. Existing solutions

Existing solutions can be divided into following two categories:

1. Open source software
2. Commercially available single-use devices (with their own closed source software)

Notable examples from the category of open source software are the following two tools: Bettercap and WiFi-Pumpkin. Bettercap is a successor of Ettercap and it allows network reconnaissance and MiTM attacks on Ethernet and Wi-Fi networks. Bettercap can be also monitored and controlled through a REST API, which is limited for local network only due to a fact that server must [1] run alongside app. WiFi-Pumpkin is focused on wireless attacks only [2]. Configuration is made through GUI which is limiting its use and CLI is not yet available. The biggest advantage of WiFi-Pumpkin is the possibility to create a transparent proxy [3] that can be hooked on by a Python script for tailored interception and modification of HTTP communication.

From commercially available tools, the most known device is WiFi Pineapple NANO from company Hak5. Biggest advantage [4] of this device is that it's single-use, therefore no initial set up or configuration is required for using it. On the other hand, its use is limited by fact that support for new attacks must be added by the manufacturer.

4. Proposed solution and challenges

Proposed solution can be split into several categories, based on the challenges which they are solving. The first challenge is the configuration of the device. Different hardware requires different configurations (for example due to a different interface names and required patches for Wi-Fi drivers). Writing bash scripts is slow and tedious process and can be a problematic for remote configuration and debugging. Bash scripts were therefore replaced with Ansible roles, as they can be easily read and have uniform interface (modules) for basic configuration (installing packages, user and group management, etc.).

This, however, creates a new problem: How to choose which roles to run and how to make simple changes in configuration easy. For example, in order to change domain, we would have to search for the corresponding variable in one of the many locations where variables and global variables could be stored. Same thing applies for roles, as it does not make sense to set Wi-Fi interface into monitoring mode, if we are doing ARP poisoning through Ethernet interface.

This problem was solved by creating a Django application that works as a configurator and a simple GUI editor. Roles can be uploaded though the Django administration interface and simple changes can be performed through regular expressions that are automatically applied before an Ansible playbook is built.

The next challenge was making the applications used in the attack stable. Development of these tools is quite active and in most cases without backwards compatibility. Same thing applies for their dependencies, which can change drastically from day to day. The solution was dockerization, where an image containing just the selected application and required dependencies is created. This image is then pushed to Docker Hub and the application is frozen in its current state for further use.

The last challenge, which is often overlooked is what to do with intercepted data. A common approach

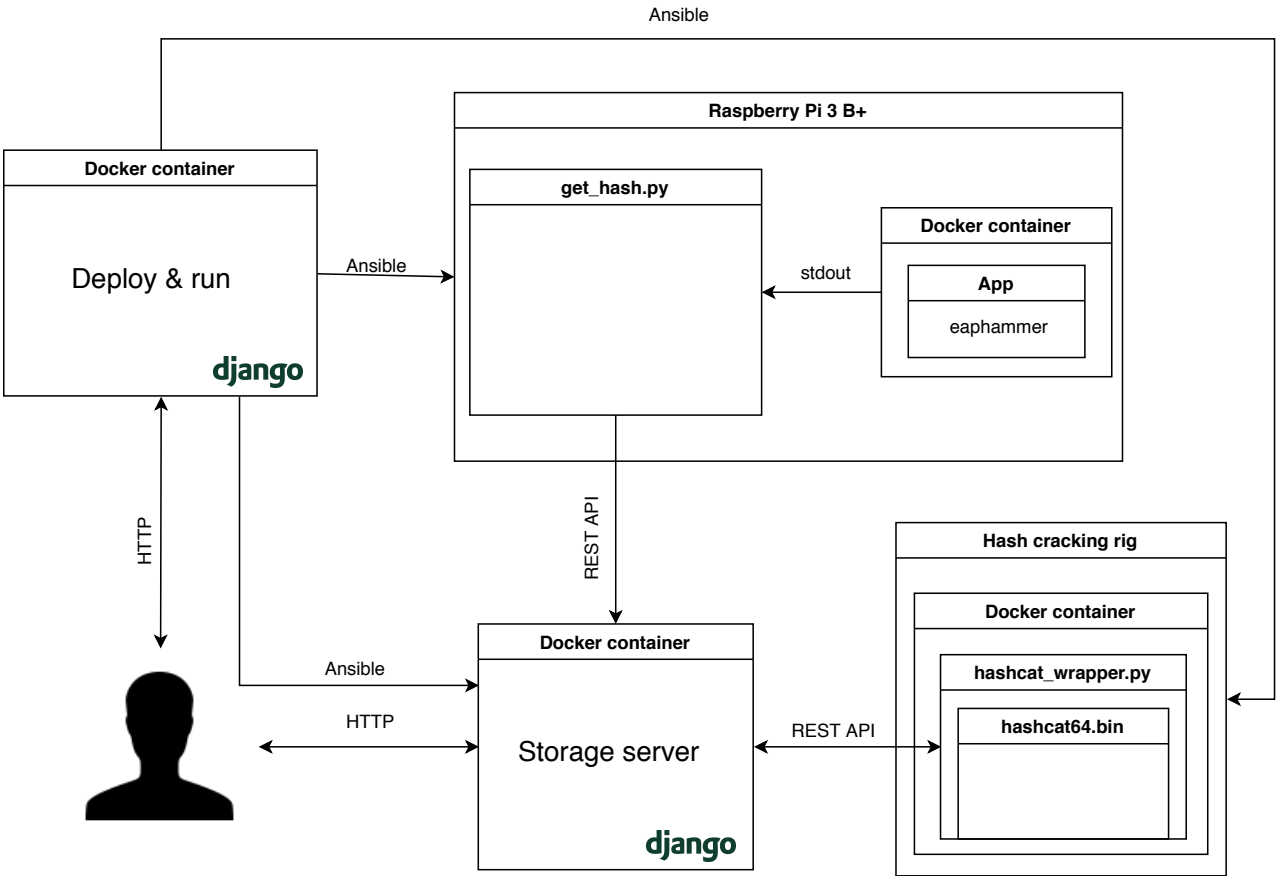

Figure 1. Project infrastructure for IEEE 802.1x protocol attack

is to store the data locally on the device, however, it would be better to have data available instantly and be able to work with them. In this proposed solution a REST API is used for communication with control server, where the data will be stored and available for attacker. The storage server is once again a Docker container which can be started in matter of minutes. Attacker can access this server and create a task for cracking password/handshake on a second, remote machine.

5. Use case: Attacking the IEEE 802.1x protocol

As a demonstration, I decided to perform attack on the IEEE 802.1x protocol, used for example by Wi-Fi networks participating in the eduroam initiative. In this use case, I will use internal Wi-Fi interface for Internet connection and external Wi-Fi interface with bigger gain to do an Evil Twin attack, as it's visualized in Figure 2. Attack will be run though the application eaphammer, which is specialized for attacks on this protocol.

Eaphammer requires that process wpa2_supplicant does not run on the selected interface. The workflow was following:

1. A Docker image (based on vanilla Kali Linux) with eaphammer was created and pushed to Docker Hub.
2. A simple Python script was created, which will run the Docker container with eaphammer command line arguments, and analyze output on stdout. In the case that handshake was captured, the script will send it through the REST API to storage server. If Internet connection is not available, the result is stored locally. Script is quite

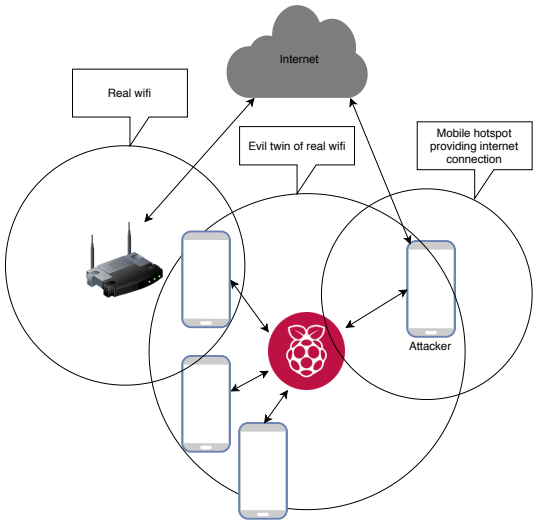

Figure 2. Device will always prefer BSSID with better signal.

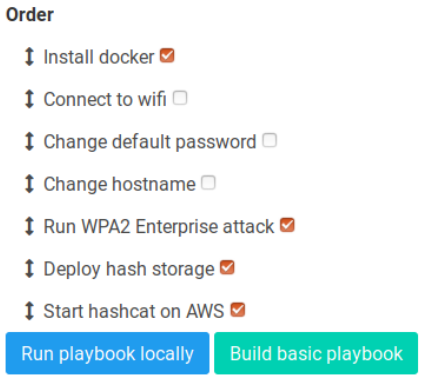

Figure 3. A graphical configurator can be used to choose which tasks should be performed

short, with only 42 lines (see `get_hash.py` for details).

3. An Ansible role was created that will copy the script (`get_hash.py`) into device, install requirements (python, docker) and pull the image from Docker Hub into device. The role also contains instructions for disabling wpa_supplicant on selected interface by modifying the configuration file of the interface, and how to set the other interface to connect to predefined hotspot. Whole Ansible role is shorter then 30 lines of code (see role: `wpa2_enterprise_attack`), making it significantly shorter than similar code written in Bash. It also can be easily read (as shown in Listing 1) and be understood even without previous knowledge of Ansible (which couldn't be said about Bash scripts).

Communication and relationship between services and devices is shown in Figure 1

This new role is then uploaded to configurator and ready for use. The only required action from the end user is to choose which roles should be run (see Figure 3). Device will be restarted when configuration is finished, and if no error occurred, a green LED on device will light up. In case the Docker container fails to start or a captured hash cannot be send to the remote server, the green LED will be turned off.

For sending data, internal Wi-Fi device is used (due to it's lower gain) and will connect to attackers hotspot, giving the device access to the Internet. It's also possible to connect to the Internet via USB tethering, in which case no further action is required. This allows creation of RogueAP for monitoring and modifying Internet traffic in a similar way as Darkhotel APT [5] used to do.

The attack and architecture were tested by faking the eduroam network SSID and the outcome was really interesting. Mobile devices with Android OS will, automatically on background, try to connect to this fake Access Point even when certificate validation failed, making this attack almost invisible for victim. A rather unexpected finding was that Comenius University in Bratislava apparently uses only 6 character long passwords, making it possible to brute force it in a matter of minutes. During 3 hour train ride, I was able to capture 49 different credentials for eduroam network and crack almost 20% of credentials using hashcat. It would be good to point out that this test did not break Slovak laws in any way. Especially §247 law n.o. 300/2005 Z. z. doesn't apply here due to a fact that intercepted data weren't misused in any way, and were used only for a verification of technological concept.

```
- name: Disable wpa2_suplicant for wlan1
  blockinfile:
    path: /etc/dhcpcd.conf
    insertafter: EOF
    block: |
      interface wlan1
              nohook wpa_supplicant
```

Listing 1. Ansible role for inserting block of text into file.

6. Conclusions

The goal of this project is to demonstrate benefits of microservices architecture for penetration testing purposes, compared to device centric approach widely used today. The main benefits of this solution are quick deployment, and simple change of command/storage server. Usage of REST API and making the single-board computer only a part of a larger infrastructure allows better availability in case when Internet is not available 24/7. Two interfaces allow simple access to the Internet, while not limiting the range of possible attacks. Simple web interface for deployment, running and configuration of Ansible roles makes this solution exceptionally easy for use. However, this approach doesn't limit user in any way compared to device centric approach and brings many benefits like scalability and stability, while also automating processes like hash cracking, deployment, and configuration.

Acknowledgements

I would like to thank my supervisor Ing. Jan Pluskal for helping me with this work. His knowledge of network attacks was valuable source of information during the design and development phases of this work. Last but not least, I would like to thank Bc. Adrián Király for corrections and help with LaTeX.

References

[1] All hail bettercap 2.0, one tool to rule them all., 02 2018. https://www.evilsocket.net/2018/02/27/All-hail-bettercap-2-0-one-tool-to-rule-them-all/.

[2] Wifi-pumpkin-ng (python 3), 2018. https://github.com/P0cL4bs/WiFi-Pumpkin/projects/3.

[3] Wifi-pumpkin - transparent proxy, 2018. https://github.com/P0cL4bs/WiFi-Pumpkin#transparent-proxy.

[4] Wifi pineapple website, 2018. https://www.wifipineapple.com/.

[5] Kaspersky Lab. The darkhotel apt. a story of unusual hospitality. Online research paper, Nov 2014. https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/08070903/darkhotel_kl_07.11.pdf.