PENIOT Design Overview Document

The purpose of this document is to clarify and explain design and development phases of project PENIOT. Respectively, the document will focus on brief product description, high level system design, overall design and alternative design options in a detailed way. Briefly, PENIOT is an extensible penetration testing tool for the Internet of Things devices, and the rest of the document will focus on technical description of it.

1. Product Description

PENIOT is a penetration testing tool for Internet of Things (IoT devices). Penetration testing is the practice of controlled security testing of a digital system to find possible security breaches. It is usually done manually by penetration testers, not by dedicated software. What we are building is a helper tool, a first step for penetration testers that want to do penetration tests on IoT devices. We are not trying to replace penetration testing personnel, rather we want to give them a go-to tool to use in the beginning of their work.

We design our attacks for different IoT communication protocols rather than specific hardware devices. By this way, we make our tool much more general. However, due to time constraints, among a myriad of IoT protocols we will only choose several important ones. For example, we implemented MQTT protocol penetration testing functionalities until now.

When finished, our tool will have an easy to use graphical user interface (GUI) that the user will be able to choose which protocol they want to test. After that, depending on the protocol, user will feed the necessary parameters and then the user will be asked to choose the attacks they want to apply. Depending on protocol, PENIOT will offer more than one type of attack that tests for security breaches. Some of the attacks we employ are fuzzing, DoS or sniffing attacks (depending on the protocol). Finally, our tool will report the result of the penetration test attack to the user. Also, our tool will be extendable rather than static. That is, penetration testers will be able to append their own penetration testing attacks to our tool.

2. High Level System View

In our design, there will be our project PENIOT in the central position. A penetration tester will interact with PENIOT's graphical user interface such that s/he selects IoT protocol of the target device, and then selects which attack will be performed by the tool after giving necessary information related to device or discovering the properties of the device by means of other attacks. All the functionality will be implemented in the tool so there will not be a need for external modules (other than imported libraries, of course).

Also, there will be a device to be tested and it should be functioning normally (in an isolated environment and normal communication protocol is conducted).After all the setup is completed, a penetration tester will perform the attacks and get the results or see the actions of the device under penetration conditions.

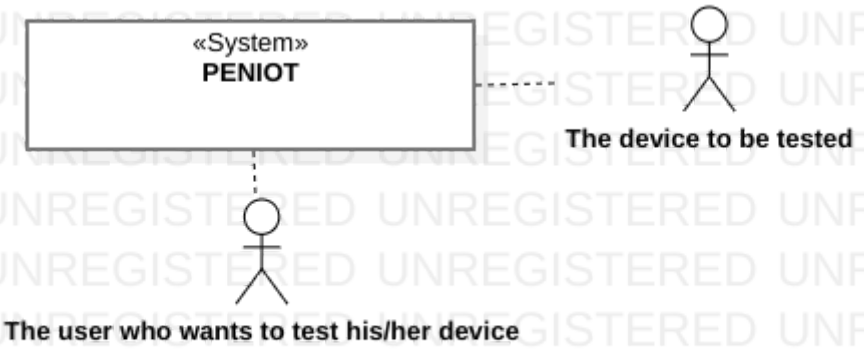

Figure 1: The Context Diagram of PENIOT

3. Overall Design

There will be one major component in PENIOT, but there will be four sub-components to test different IoT protocols and the graphical user interface. From now on, they will be described in a detailed way.

Firstly, there are four sub test modules which are decided to the IoT protocols for Zigbee, Bluetooth Low Energy, RPL and MQTT. In each test module, a protocol has at least one attack which the user can choose in the context of penetration testing. Depends on the selected attack, we may need to get additional information such as IP address or MAC address from the user since we need these information to implement them.

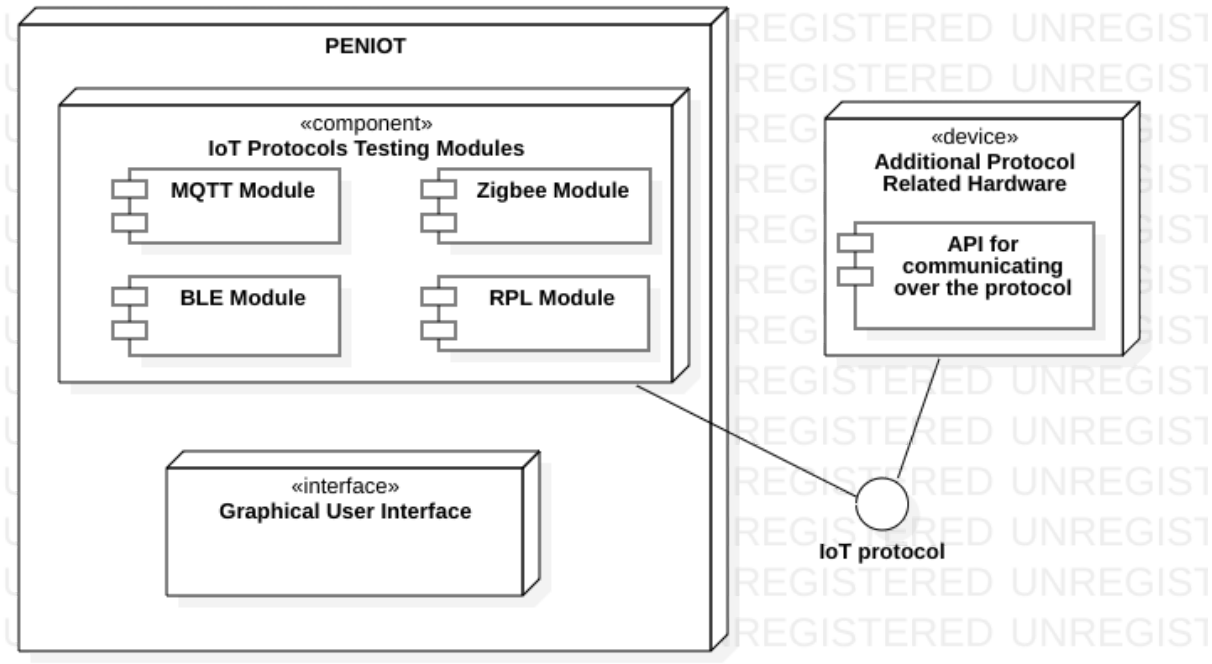

Figure 2: Deployment Diagram for PENIOT

Moreover, we can learn much more information by sniffing the network (but also we can use previously mentioned properties to get information) if the sniffing attack is implemented or provided by examining valid packets. Example of sample attack procedure will be explained at the last paragraph of this section. Also, reporting mechanism is planned to be added for the possible observed results, but for some types of attack, PENIOT may not be able to produce resulting report since vulnerable target devices could be forced to shut down or break down in some test cases, which results in losing connection to it. In these cases, result of the test should be inferred by the penetration tester by observing the target device rather than being reported by the PENIOT.

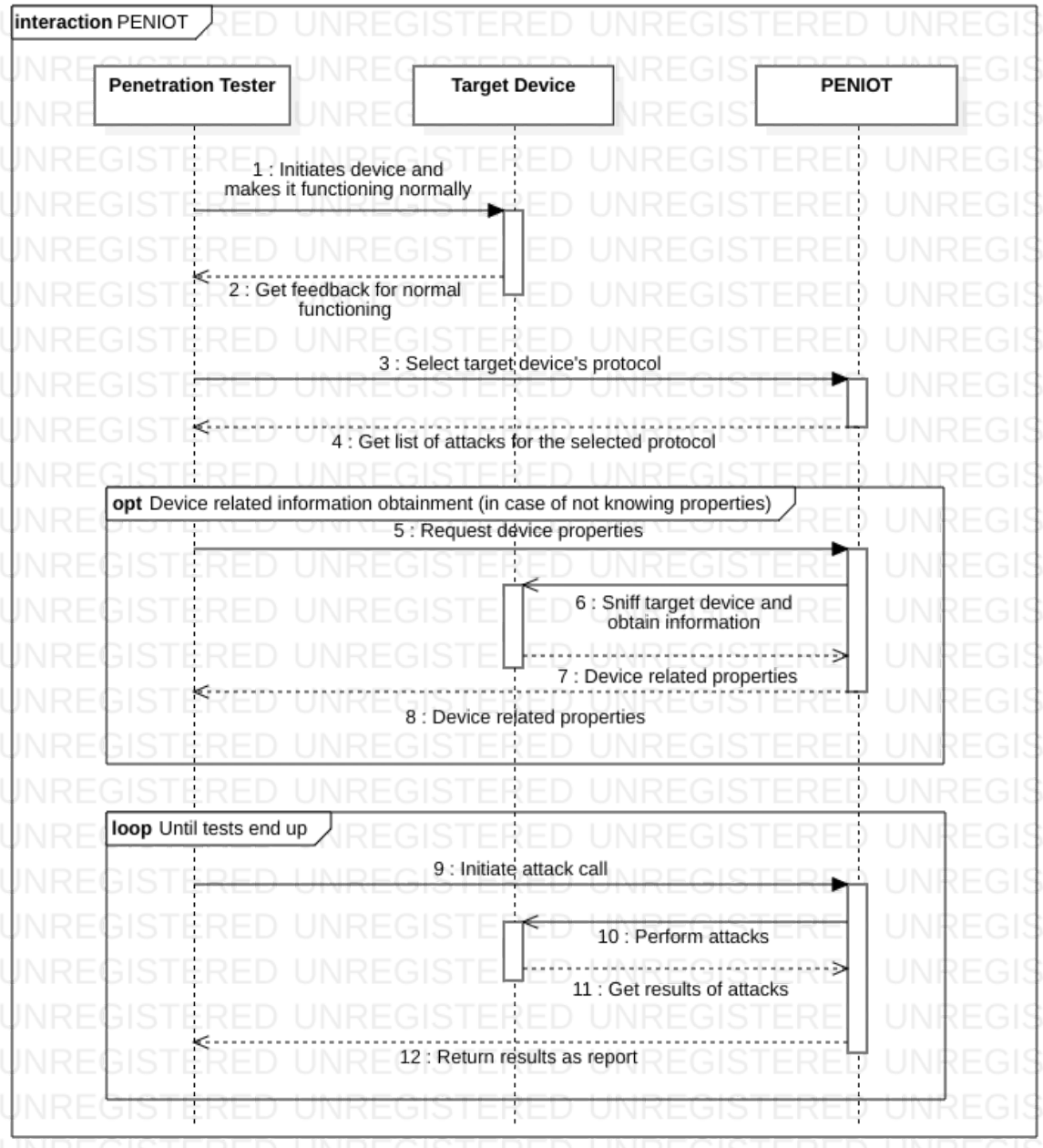

Figure 3: Sequence Diagram for performing attack with PENIOT

Graphical user interface (GUI) is the entry point to PENIOT. It enables the user to choose which protocol is used to test his/her device. According to the selection, corresponding module will be called. To minimize the dependencies between the implementations, we decided to have separate modules for each protocol. Moreover, GUI enables extending PENIOT without affecting other protocol implementations. We planned to import user defined modules and enable him/her with the own implementation to test target device.

Also, we may need to use additional hardware related to the selected IoT protocol. Since some protocols are implemented on lower layers in networking layers, PENIOT needs adapter hardware to communicate. If they are needed, the hardware for that protocol will be specified in the respective protocol's documentation. An example of a case that needs such hardware is, we can sniff a valid packet and use it to perform a replay attack. Some protocols needs additional hardware tools such as dongles to sniff communication.

Depending on cases, the report may give a warning message denoting a security breach or the attack may result in the break down of the tested device. In the second case, penetration tester may need to infer the result of the attack from the malfunctioning of the tested device.

4. Alternative Design Options

We could make our tool final, static and non-extendable but this would not be realistic for several reasons. First of all, this would make PENIOT as a very limited and powerless tool for its purpose because we will be cover only a small portion of IoT protocols due to time limitations. Secondly, IoT technologies are quickly evolving and a static tool would be quickly outdated. Finally, most users of this tool will be people who have good knowledge of programming (penetration testers) and some of them may be willing to extend this tool in ways that will suit their needs better.

Another design choice is about the choice of protocols, but this depends more on available hardware rather than our choice. PENIOT implementation is pretty straightforward. What we do is choosing the protocols to implement attacks for, studying the protocol and choosing and implementing some security attacks depending on the weaknesses of the chosen protocol. We could choose different set of attacks other than those what we planned to do in future or we could choose very different IoT protocols, however as our tool is extendable necessary modules can be integrated easily in the future.

From programming perspective, we choose Python as our development programming language. There are a few good reasons for this. First of all, there are many third party Python libraries which were developed for IoT protocols. Secondly, being a scripting language, Python is very suitable for implementing an extendable tool. We could choose another programming language, but Python stands at an optimal point for the development ease-functionality trade off.

The main constraint for not having alternative method is the fact that actually PENIOT will be a penetration test framework such that there will not be much interaction

except for the tester and the device to be tested. Since PENIOT is a encapsulated testing package, finding possible alternative methods is pretty difficult.