JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Kai Knabl, BSc (k1457305)

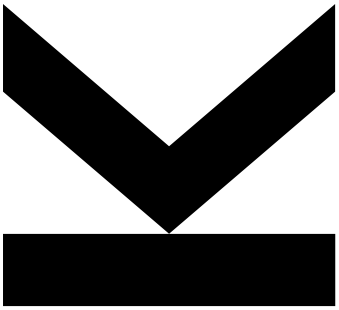
Submitted at
Institute of Networks and Security - JKU

Supervisor
Assoz.Prof.Mag.Dipl.-Ing. Dr. Michael Sonntag

Co-Supervisor
Assoz.Prof.Dipl.-Ing. Dr. Karin Anna Hummel

June 2020

Design, Implementation and Evaluation of a Mobile Security Scanner App for Smart Home Users



Master Thesis
to obtain the academic degree of
Diplom-Ingenieur
in the Master's Program
Computer Science

JOHANNES KEPLER UNIVERSITY LINZ
Altenberger Str. 69
4040 Linz, Austria
www.jku.at
DVR 0093696

Table of Contents

ACKNOWLEDGEMENTS

First and foremost I would like to thank my supervisors Assoz.Prof.Mag.Dipl.-Ing. Dr. Michael Sonntag from the Institute of Networks and Security and Assoz.Prof.Dipl.-Ing. Dr. Karin Anna Hummel from the Institute of Telecooperation – both of them regularly provided essential input and constructive feedback. Furthermore, their substantial patience throughout the entire course of the project enabled me to create a thesis that I am personally content with.

I would also like to thank Ass.Prof.Mag. Dr. Elisabeth Kapsammer and a.Univ.-Prof.Mag. Dr. Werner Retschitzegger from the Institute of Telecooperation, a.Univ.-Prof.Dipl.-Ing. Dr. Wolfram Wöss from the Institute for Application Oriented Knowledge Processing, Univ.-Prof.Mag. Dr. Alois Ferscha from the Institute of Pervasive Computing and Univ.-Prof. Dr. Gerhard Widmer from the Institute of Computational Perception. They gave me the opportunity to present the user study – which is an essential part of the thesis – in their courses.

Additionally, I would like to thank the numerous participants of the user study. Special appreciation also goes to the persons that took part in the preliminary user study. Without the contributions of all participants and their interest towards the project, an objective user experience evaluation of the app that was created in the course of this thesis, would not have been possible.

Furthermore I would like to thank my grandfather Josef who has been an important role model for as long as I can think, not least due to his light-hearted and courageous attitude towards life. Finally I would like to express deep appreciation for my parents Karin and Klaus – their caring and encouragement enabled and accompanied my entire academic development.

Abstract

In recent years the growing utilization of a large variation of Internet of Things (IoT) devices in the end-user domain has proven to provide an extensive and highly problematic attack surface. Very often the devices in question utilize network services with easily accessible and poorly secured authentication mechanisms, frequently making use of easy-to-guess standard credentials. In numerous incidents adversaries were able to infect a high number of insecure IoT appliances with malware. They could do so by scanning large portions of the Internet for appropriate devices and by employing brute-force attacks to gain full access. Then cyber criminals mostly utilized the hijacked devices to carry out devastating distributed denial of service (DDoS) attacks. It has become increasingly difficult for end-users to keep track of the number of appliances in their home networks, and in particular to make sure that each of these devices is secure.

The main-part of this thesis is therefore the implementation of a contemporary end-user oriented solution for detecting security threats and explaining them to the user. Due to the widespread use of mobile devices, the security scanner is implemented as a native and easy-to-use Android application. By employing brute-force password-guessing techniques the app is able to detect if File Transfer Protocol (FTP), Telnet, Secure Shell Protocol (SSH) and Hypertext Transfer Protocol (HTTP) services are using standard authentication credentials. It can also detect Universal Plug and Play (UPnP) profiles that weaken a device´s security-state.

To make the application easily understandable for users with different degrees of technical knowledge, three application modes are introduced. An “Expert” mode lets more advanced users display detailed device information. The other two modes are dedicated to users with less technical knowledge. One of them reduces information which might be too complex or confusing for novice users. The other mode is making use of a more novel approach, representing devices and inherent vulnerabilities in an easily understandable graphical fashion. The app also provides a history feature which enables users to keep track of changes across multiple network scans. The thesis is concluded by an elaboration on the results of a user-study.

Kurzfassung

In den letzten Jahren hat sich herausgestellt, dass der zunehmende Einsatz von verschiedensten Internet of Things (IoT) Geräten im Endbenutzer Bereich, eine weitreichende und sehr problematische Angriffsoberfläche darstellt. Die betroffenen Geräte nutzen sehr häufig Netzwerk Dienste, welche einfach zugängliche und unzureichend gesicherte Authentifizierungsmechanismen verwenden. Sehr häufig sind diesbezüglich einfach zu erratende Standard Anmeldedaten im Einsatz. In zahlreichen Fällen konnten Angreifer große Mengen an IoT Geräten mit Schad-Software infizieren. Derartige Angriffe konnten gelingen, indem größere Teile des Internets gescannt und mittels der Durchführung von Brute-Force Angriffen der vollständige Zugang zu entsprechenden Geräten ermöglicht wurde. Danach werden die infizierten Geräte von Cyber Kriminellen meistens dazu verwendet um verheerende verteilte Denial of Service (DDoS) Angriffe durchzuführen. Für Endbenutzer wird es zunehmend schwieriger den Überblick über die Anzahl der Geräte in ihren Heim-Netzwerken, oder gar über Sicherheits Probleme einzelner Geräte, zu bewahren.

Der Großteil dieser Arbeit erläutert daher die Umsetzung einer modernen, Endbenutzer-orientierten Lösung mit der gängige IoT Sicherheitsprobleme aufgespürt und erklärt werden können. Aufgrund der weiten Verbreitung von mobilen Endgeräten wird die Lösung in Form einer nativen, einfach zu benutzenden Android Applikation implementiert. Anhand der Durchführung von Brute-Force Techniken ist die App in der Lage herauszufinden ob File Transfer Protocol (FTP), Telnet, Secure Shell Protocol (SSH) und Hypertext Transfer Protocol (HTTP) Dienste Standard Authentifizierungsdaten verwenden. Weiters kann die Anwendung Universal Plug and Play (UPnP) Profile erkennen, welche den Security-Status eines Geräts schwächen.

Um die Anwendung für Benutzer mit unterschiedlichem technischen Wissensstand möglichst leicht verständlich zu gestalten, werden drei Anwendungsmodi eingeführt. Ein „Experten“-Modus erlaubt es technisch versierteren Benutzern, Detailinformationen zu Geräten einzusehen. Die anderen zwei Modi sind auf Benutzer mit geringeren technischen Kenntnissen ausgerichtet. Der erste dieser Modi reduziert Informationen, die für Laien zu komplex oder verwirrend sein könnten. Der zweite Modus stützt sich auf eine neuartige, einfach verständliche grafische Darstellung von Geräten und Schwachstellen. Weiters bietet die App Historisierungs Funktionalität welche es Benutzern erlaubt Änderungen über mehrere Netzwerk-Scans hinweg, nachzuverfolgen. Die Arbeit wird mit Erläuterungen der Ergebnisse einer Benutzerstudie abgeschlossen.

1. Introduction

Initially getting larger media attention in late 2016 with the Mirai malware incident [1], standard network-devices such as printers, routers and Internet Protocol (IP) cameras have been misused for creating large botnets. All of the devices were infected with malware via Telnet due to easy-to-guess standard authentication credentials, provided by the manufacturers during production. Those botnets were mainly used to run devastating DDoS attacks as seen with the attack on the "Dyn" Domain Name System (DNS) provider on 21 October in 2016 [2].

However, with the emergence and increasing popularity of crypto currencies, said botnets have also been utilized to perform crypto-mining activities [3]. Regarding the recent IoT trend it becomes very obvious that a multitude of other, connected devices based on the TCP (Transmission Control Protocol)/IP protocol stack, will be susceptible to similar attacks. IoT appliances that might be problematic, once they are in wider use, include smart meters, smart fridges, various home sensors and actuators or smart buildings.

In general as with many software projects, the inherent problems with applications running on IoT devices can mostly be traced back to problematic development processes. In many cases, secure coding practices and common security measures like regular code-reviews fall short, often due to restrictions in terms of time and resources [4]. Therefore in many software development projects, the main focus is put on the quick and inexpensive implementation of a functioning, stable and easy-to-use application up to a certain deadline. The creation of a product like that, still tends to be seen more economically viable than software that also employs sufficiently sustainable security measures [5]. This often results in the release of products which – besides other security problems – employ insecure authentication mechanisms. This is especially true for the use of standard-passwords, which provide an uncomplicated but extremely insecure possibility to access a device´s configuration [6]. On top of that, IoT vendors often support access protocols for the low-level configuration of devices, such as Telnet or SSH. This might be useful for development purposes and for certain device-types such as routers. However, for many device types – such as IP cameras – the necessity of such functionality for the final product is questionable at best. Still, device vendors often fail to remove unnecessary low-level configuration possibilities which make use of easy-to-guess credentials, after development. This effectively results in backdoors within shipped products.

To worsen the situation, appropriate regulations in the IoT sector are still not in place. Even if device vendors would take security more serious in the near future, the problem persists with vulnerable devices that are still in use. IoT vendors also often seem to lack the resources for proper maintenance and patching procedures. This is especially true when a larger variety of IoT gadgets was already sold and employed. From the consumers´ perspective, patching also leads to system downtimes. End users and administrators often perceive the risk of service disruption due to patching to be higher than due to a cybersecurity incident. The result is that even if there are patches available, consumers might apply them in bulk and at sporadic intervals. Sustainable patching activities for devices in the IoT domain have therefore proven to be challenging tasks – from both the vendor and the consumer perspective [7].

Furthermore internal networks behind Network Address Translation (NAT) are often viewed as secure – many people don't see the need to properly configure the services of devices which are not directly accessible from the Internet.

However, examples for malware do exist that do not directly attack an infected device but other devices within the same network. The “Switcher” Android malware for instance, does not directly target the user of an infected smartphone but rather the wireless network that the phone is connected to. The malware does so by brute-forcing access to the corresponding router and subsequently changing its settings to make use of a rogue DNS server controlled by the attacker. The rogue DNS server can then resolve the domain names of benign services to IP addresses of malicious services such as web sites containing phishing pages or malware. This effectively means that a single infected smartphone could result in a tremendous security risk for all other devices connected to the same access point – in cases where the corresponding router is making use of standard credentials [8].

Research Questions

The research questions for the thesis are formulated as follows:

1. How to design and implement a contemporary application for enumerating network devices and for detecting and explaining common IoT security issues?
2. How to make the application appropriately usable for end-users with different levels of technical knowledge?

Due to the widespread use of mobile applications in the end-user domain, the decision is made to implement the solution as a native Android application. The intention of the app which is created in the course of this thesis is to help users to detect security risks and solve problems before any of the afore-mentioned scenarios can occur. The application should not search for device-specific exploits as those problems can often only be fixed by patches from the vendor. Instead the application should try to uncover more general problems that apply to a large number of devices and that the user is more likely to be able to solve by him/herself. By employing brute-force approaches the solution should be able to detect the use of standard credentials in the authentication of HTTP, Telnet, SSH or FTP services. Those services are chosen because their common use in combination with standard credentials has proven to be one of the biggest IoT security threats. The app should also able to detect several UPnP profiles which have proven to introduce security problems.

After getting informed about the number and security state of the devices in their home networks, users should then be able to take appropriate actions to solve the problems found with reasonable effort. Mitigations suggested by the app are changing user credentials or disabling corresponding services. It is also mentioned that the user might take the device offline entirely, which might be viable in certain scenarios. The number of devices in home networks is steadily growing, making it increasingly harder to keep track of the number of connected devices. Because of this, there might even be usage scenarios where problematic devices are discovered that the user does not need anymore and has forgotten about. In those cases simply taking the affected devices offline entirely, would mitigate the problem.

For making the app appealing to users with more technical knowledge, as well as users with limited technical knowledge, different application modes are introduced and studied with respect to their effect on user performance and user acceptance.

- Expert Mode: This mode is targeted towards technically experienced users and displays all the information that the app is able to gather regarding networks, devices and vulnerabilities.

- Normal 1 Mode: This mode is targeted towards technically inexperienced users and is mainly reducing information which could be considered too complex, such as IP- or Media Access Control (MAC)-addresses.
- Normal 2 Mode: This mode is also targeted towards novice users but offers a graphical visualization of networks and inherent vulnerabilities. A graphical network representation is chosen to provide a novel, visual and more intuitive alternative to just textually listing devices.

To make changes in networks and individual devices visible over time, a dedicated history feature should be implemented within the app. Every time a scan on an entire network or a scan of an individual device takes place, snapshots of the corresponding devices are created. The data are then visualized via three different timeline charts, based on the selected application mode. This enables users to draw important conclusions from multiple scans over time.

For being able to objectively determine the user experience of persons interacting with the application, a usability study with a total of 30 test-users is conducted. The study experiment consists of three different tasks that the participants have to solve by using the app. In addition, the test users are asked to rate their experience with the app along well-known user experience properties. One major goal is to compare the users´ interactions regarding the three different modes of the application, which are briefly described above.

2. Related Work

In the following sections related scientific work and other applications are described, which provide functionality similar to that of the project created in the course of this thesis. In one sub-chapter an application created by the author of this thesis is described. It served as prototype and basis for the current implementation of the IoT Scanner App.

2.1 Related Scientific Work

Related to this work is a publication by Ruth M. Ogunnaike and Brent Lagesse [9]. Their paper is discussing the requirements and implementation of an IoT architectural framework based on Software Defined Networking (SDN) which should enable consumer-friendly security in smart environments. In this system IoT devices are scanned for security problems by using custom vulnerability scanners and penetration testing tools before being allowed to communicate with other devices. Custom vulnerability scanners include programs that detect common usernames and passwords. In cases where the use of weak/default credentials is detected, the system automatically changes the password to a secure one and sends it to the user via email. The solution described in this paper requires a software setup running on a virtual machine. The paper also states that while a SDN was used for its ease of deployment and testing, it would be possible to create a similar system using traditional routers. While the scanning for default credentials described in the paper is also similar to the approach described in this thesis, the software/hardware setup is clearly not targeted towards a mobile solution. However, this thesis specifically targets a mobile solution mainly for usability reasons.

A paper by Gudrun Jonsdottir et al. [10] also describes a non-mobile system called “IoT Network Monitor” which provides a user friendly interface for consumers to visualize vulnerabilities of IoT devices in their homes. The system runs on a Raspberry Pi device configured as a router and supports several network-security related operations.

It performs deep packet analysis for detecting the transmission of potentially sensitive personal information in cleartext. The solution is also able to detect botnet traffic originating from IoT devices connected to the network. Lastly, the IoT Network Monitor detects devices with default passwords that might be exploited by attacks such as the Mirai botnet. This again is similar to the functionality used in the project that is described in this thesis. The program then also changes default device passwords to randomly generated strings and reports the new passwords to the user.

Another approach for an IoT vulnerability scanning system is proposed by Yu-rui SU et al. [11]. The system consists of several components including a traffic sniffer, an IoT vulnerability scanning server and a web visualization terminal. Therefore the proposed architecture is also clearly not meant to be implemented as a mobile solution. However, the paper describes a feature that is able to detect the use of default credentials for the access of a device´s web-based management interface. This functionality appears to be similar to the HTTP authentication brute force functionality that is implemented for the project described in this thesis.

It is important to note that all systems mentioned above are utilizing additional hardware that would need to be installed in a user´s home network. Those setups have the considerable advantage that the network can be constantly monitored and checked for anomalous traffic, attacks, vulnerabilities and new or missing devices. It becomes very apparent that systems which utilize dedicated hardware, are far more adequate for providing an overall security-monitoring solution when compared to network scanner applications that run on mobile devices. For one thing, mobile apps are problematic in terms of constant network monitoring, as the device that runs the app might frequently get disconnected from the network. The device might often also get powered off, not least due to low battery times. Secondly, full access to low level network-traffic is often harder to obtain on mobile devices due to platform restrictions. On Android systems for instance it is only possible to fully gather all incoming traffic by enabling the “promiscuous mode” of the device´s Wi-Fi adapter. However, it is only possible to enable this mode with root user rights. For security reasons most off-the-shelf Android consumer devices are utilizing just regular user accounts with restricted rights. This means that users would first have to put additional effort into rooting their device in order to make use of apps that utilize more advanced networking features. Expecting users with limited technical knowledge to have the abilities and the will to do this, does not seem reasonable.

However, when considering the recent increase in the distribution of insecure IoT consumer devices – many of which make use of standard authentication credentials – one might come to the following conclusion: While dedicated security hardware allows more sophisticated vulnerability detection, the use of a mobile app that helps users to quickly detect at least the most severe security problems is still viable. This is especially true regarding the home-networks of end-users, not least due to the following additional reasons. One of the biggest advantages of an app that runs on a non-rooted Android device over a solution that requires a dedicated hardware appliance is the considerably easier procurement and the low configuration effort for users. Many people already are in possession of one or more mobile Android devices that are connected to their home network. This means that a person could simply download the app from the Google Play Store, run it and perform a first scan. By doing so the user can quickly get an overview on the devices in their network and can reveal severe vulnerabilities such as the use of standard authentication credentials. Besides the easy distribution of the app, the utilization of Google infrastructure provides further advantages such as the considerably easy detection of bugs in the app and the roll-out of fixes or new features.

Another obvious advantage of a mobile solution is the fact that the app might easily be used on different networks. Granted that the user has permission to do so, scans on someone else´s network might help the owner to detect and fix vulnerabilities.

2.2 Other Mobile Network Scanner Applications

Among the most related applications, is the Zanti Diagnostic Pentesting app which is able to perform a range of detailed network scans and vulnerability tests, due to the internal use of the Network Mapper (Nmap) and Metasploit projects. The app includes functionality such as traffic hijacking or man-in-the-middle attacks and enables the user to perform password audits on individual devices or to manually select exploits that in turn might be executed on a target device. Users require more advanced knowledge on those topics in order to successfully use the Zanti app. This is the reason why it can be considered an expert tool. Moreover, the app can only be used to full extent on a rooted Android device [12].

Another related application is a very popular, free network scanner called Fing which displays lots of network-information including IP, MAC and host name. Overall the Fing app detects devices quickly and reliably and convinces with its thoughtfully crafted user interface. It also automatically saves scan-results, marks missing devices and provides a basic scan-timeline for individual devices. During recent updates, the developers even included a feature that lets users edit the appearance of device and network lists (“Standard”, “Simplified” and “Technical”). Those modes differ in terms of the presented information which is quite similar to the functionality of the “Expert” and “Normal 1” modes described in this paper. However, Fing does not provide a graphical network representation (“Normal 2” – mode) and also offers no timeline for entire networks and services for individual devices. Furthermore the app´s device history entries are merely listed textually. The app described in this paper on the other hand, makes use of a visual approach by introducing three different history-chart types. Fing also lacks functionality in reliable port scanning and does not detect any security related aspects, such as weak passwords [13].

Several other “general” network scanners, that don’t claim to detect security issues, have been tested, including ezNetScan [14] and Net scan [15]. Both show deficiencies in reliability, possible network scan-range or displaying sufficient device-information.

The “Dojo by Bullguard” application is primarily meant to be used alongside a dedicated hardware appliance, which is directly integrated into home networks. The vendor claims that the product is able to “protect all of the user´s connected home devices from malware, viruses and any cyber attacks”. The app on its own is able to detect network devices and to subsequently scan devices for open ports. After detecting an open port the app assigns a severity level and provides a generic message, which states that “open ports can be exploited by hackers and potentially enable them to gain remote control over the device”. However, the app does not actually check the service for insecure authentication – it merely detects that the corresponding standard port is open. The app shows very limited additional device-information and does not provide a scan history [16].

Similarly, the “IoT Scanner” app by Kaspersky detects devices in home networks and scans for open ports which might run security critical services. The publishers claim that the app is able to detect weak or default logins on devices. This is not true – the app merely suggests that the user should disable certain services, without actually checking any user credentials. The app informs the user when new devices connect to a network.

It does so in seemingly random intervals without providing any mapping from devices to networks. The scanner also provides very little additional device information and has no history functionality [17].

Another similar application is called “IoT Security”. Its publishers claim that the app is able to list all devices in a network and reliably detect security problems and weak passwords. The app supports IoT devices such as routers and IP cameras and also should provide instructions on how to fix any found problems. However, the app provides insufficient device information and no history. It also failed to detect any problems when used to scan networks, containing multiple devices which make use of standard credentials for SSH and Telnet [18].

Another app meant to detect vulnerabilities in home-network devices is the IoPT (“Internet of Protected Things”) scanner. Here a scan is performed in two steps. First device “fingerprints” are created, containing information like open ports and running services. In a second step the app tests devices for a range of potential vulnerabilities. The app´s description states that the software is able to detect Server Message Block Protocol (SMB) vulnerabilities like “Eternal Blue” and the “Double Pulsar” backdoor installations. Other examples include the “String Bleed” exploit, which utilizes unauthorized Simple Network Management Protocol (SNMP) read/write and most recent router authorization bypass vulnerabilities. However, most checked vulnerabilities seem to be very specific which introduces issues as described in Section 3.1.3, including annoyingly long scan times. On a positive note, this app – compared to most others that have been mentioned – actually scans devices for actual problems. Yet, the used terminology definitely appears to be too technical for end-users with non-technical backgrounds. Furthermore the app seems to only list devices that could be successfully fingerprinted, meaning that a device provides at least one service which is considered relevant enough for further investigation. On top of that devices are only identifiable via their IP addresses. Scan-history functionality is not existing. The convenient use and meaningful interpretation of vulnerabilities by non-expert users therefore seems questionable. Furthermore, when tested with devices that contain insecure Telnet and SSH authentication the app failed to detect any problems [19].

2.3 Own IoT Security Related Work

Own previous work on the IoT security topic includes the implementation of a very basic network scanner created in the course of the Mobile Computing combined lecture in 2016. In the following subsections elaborations on this prototype and how it is extended in the course of this thesis are given – mostly regarding the deficiencies found in the related apps described in Section 2.2.

2.3.1 First IoT Scanner App

The app enables users to enumerate network devices and check them for obviously insecure HTTP and Telnet authentication. Its code is used as a first experimentation prototype and basis for the current implementation. The screenshots on Figures 1, 2, 3 and 4 illustrate the app´s UI and functionality.

Figure 1 – Main screen

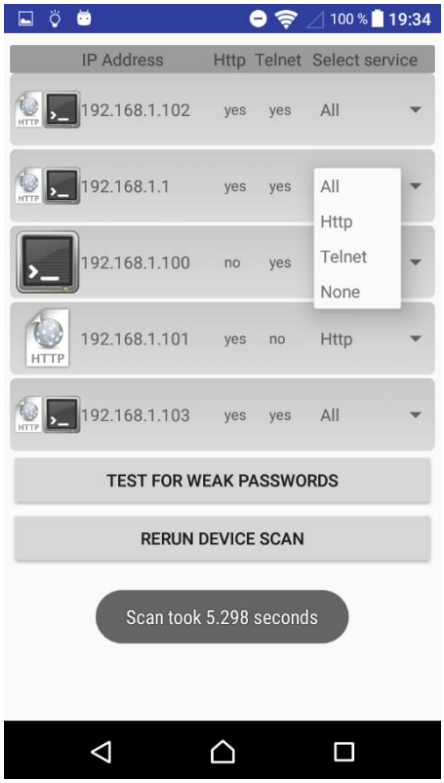


Figure 2 – Device detection

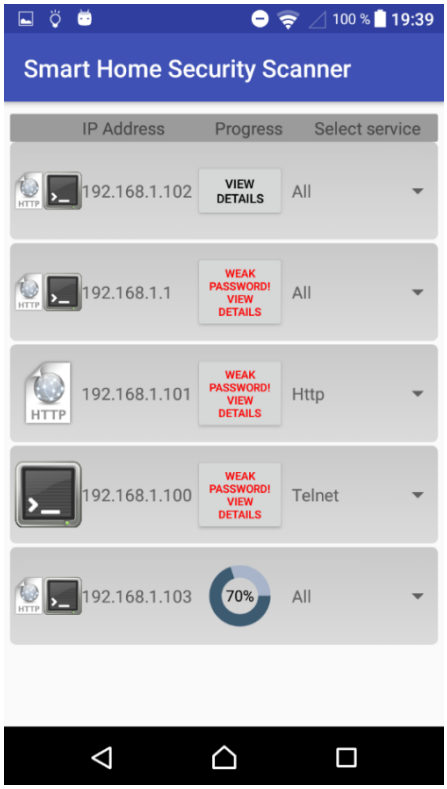


Figure 3 – Brute force on Telnet and/or HTTP authentication

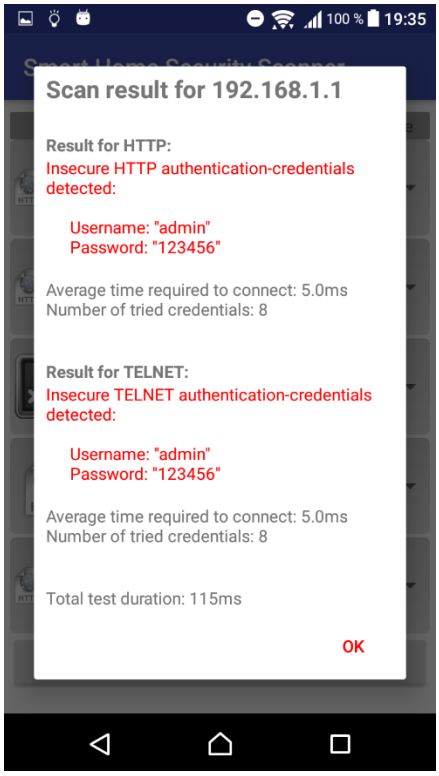


Figure 4 – Results of the brute force attacks

2.3.2 Extensions of the First IoT Scanner App

The app that is created in the course of this thesis is introducing new functionality or is significantly improving multiple features that are insufficiently implemented in the first prototype and in most comparable apps listed in Section 2.2.

Application Modes for Adapting the App to Fit the User´s Technical Knowledge

The first prototype of the IoT Scanner App and other apps are mostly either too technical or provide too little information on devices/vulnerabilities. The second prototype solves this disparity by introducing different application modes. An “Expert” mode, a “Normal 1” mode and a “Normal 2” mode. This makes it possible to easily adapt the app´s availability of functionality and presentation of information appropriate to the user´s level of technical knowledge. The Expert mode provides detailed information on networks and devices and retains all of the app´s functionality such as the possibility to perform extended port-scans on individual devices. Both Normal modes remove functionality and hide or re-name information that might be considered too complex or confusing for users with limited technical knowledge.

Graphical Network and Vulnerability Visualization

The second prototype´s Expert and Normal 1 modes that were mentioned in the last section as well as the first prototype and all other comparable apps that are described in Section 2.2 are merely textually listing a network´s devices. Here the second prototype´s Normal 2 mode introduces an innovative feature by providing a carefully constructed, interactive graphical visualization of devices in a network and inherent vulnerabilities. Furthermore the visualization is specifically designed to be well understood by non-expert users.

History Functionality

The first prototype and most comparable apps are completely missing scan-history functionality – an important feature which makes it possible to keep track of a network´s development over time and for comparing individual scans. This is especially true for security-related information, regarding important questions such as:

- When did the scan take place where a specific device was first detected in a network?
- When did the scan take place that first revealed a security-problem with a specific device?
- When did the scan take place that first revealed that a security problem is not existing anymore with a specific device?

The implementation described in this thesis enables users to easily answer those questions by providing three different, interactive history charts which are based on snapshots that are created for each scan. These charts allow users to drill-down on snapshots of entire networks or to go even further by drilling down on snapshots of individual devices within that network. This enables users to track changes over time on a very granular level, including detailed information on found vulnerabilities.

Improved Information Gathering on Networks and Devices

The first prototype as well as many comparable apps generally provide very limited information on individual networks and devices. Some apps only provide a network´s Service Set Identifier (SSID) and make individual devices identifiable by only their IP and MAC addresses for instance. The second prototype however, automatically extracts additional data such as Internet provider information if the network´s router is connected to the Internet. For individual devices the app tries to extract data such as the MAC-vendor, Netbios name, host name, UPnP information and SNMP information. This approach not only provides additional, potentially interesting data for technically advanced users but also helps novice users to easier identify individual devices. The afore-mentioned additional data often contains human-readable information that can be understood considerably better compared to primarily technical identifiers such as IP or MAC addresses.

Scanning for Weak Authentication in Common Network Services

The first prototype described in Section 2.3.1 already provided reliable functionality for detecting whether HTTP and Telnet authentication uses standard credentials. It is worth mentioning, that most of the comparable apps from Section 2.2 do not even attempt to detect weak authentication in common services such as HTTP, Telnet, SSH or FTP. Some of those apps detect the use of those services by performing port-scans. But then instead of actually testing those services for weak authentication via brute force – like the second prototype does – the other apps merely display warnings that those services might be attacked by adversaries.

Different Modes for Controlling Scan Behavior

In order to give the user better control over scans on entire networks, the second prototype provides three different scan modes which differ in the kind of tests that are automatically performed on devices after they were detected. This for instance provides the possibility to disable port- or vulnerability-scanning which might be interesting for scenarios where the user just requires a regular network scanner. This might be useful for cases where the user is merely interested in which devices are online or where the user does not want to perform aggressive tests on all devices of a network. After such a regular scan it is still possible to perform extended port- or vulnerability scans on individual devices. The first prototype already supports some level of control by providing the possibility to exclude services from vulnerability scans. However, the apps from Section 2.2 that claim to perform vulnerability scans on all devices of a network are not providing any means to control what kind of scanning actually takes place.

3. Background and Design

In the following subsections elaborations are given on the reasoning and theoretical background behind decisions that are made regarding important parts of this project.

3.1 Vulnerabilities

In the subsequent sections an overview is provided on prominent IoT security issues and their severity ratings. Reasons are given why some vulnerability types are chosen to be addressed within the application while other types aren’t.

3.1.1 Overview and Rating of IoT Vulnerabilities

The Open Web Application Security Project (OWASP) organization known for numerous web application security projects and guidelines started an Internet of Things project in 2014. Similar to its renowned web application vulnerability top 10 lists, OWASP also released two IoT vulnerability top 10 lists so far. The project team responsible for the lists consisted of volunteer professionals from within the security industry, with experience spanning multiple areas of expertise, including manufacturers, consultants, security testers, developers and more. Table 1 shows the most recent list which has been released in 2018. The ranking reflects the most actual impact and damage caused by individual issues [20].

Rank	Title	Description
1	**Weak, Guessable, or Hardcoded Passwords**	Use of easily bruteforced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorized access to deployed systems.
2	**Insecure Network Services**	Unneeded or insecure network services running on the device itself, especially those exposed to the Internet that compromise the confidentiality, integrity/authenticity, or availability of information or allow unauthorized remote control.
3	**Insecure Ecosystem Interfaces**	Insecure web, backend Application Programming Interface (API), cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. Common issues include a lack of authentication/authorization, lacking or weak encryption, and a lack of input and output filtering.
4	**Lack of Secure Update Mechanism**	Lack of ability to securely update the device. This includes lack of firmware validation on device, lack of secure delivery (un-encrypted in transit), lack of anti-rollback mechanisms, and lack of notifications of security changes due to updates.
5	**Use of Insecure or Outdated Components**	Use of deprecated or insecure software components/libraries that could allow the device to be compromised. This includes insecure customization of operating system platforms, and the use of third-party software or hardware components from a compromised supply chain.
6	**Insufficient Privacy Protection**	User's personal information stored on the device or in the ecosystem that is used insecurely, improperly, or without permission.
7	**Insecure Data Transfer and Storage**	Lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.
8	**Lack of Device Management**	Lack of security support on devices deployed in production, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities.
9	**Insecure Default Settings**	Devices or systems shipped with insecure default settings or lack the ability to make the system more secure by restricting operators from modifying configurations.
10	**Lack of Physical Hardening**	Lack of physical hardening measures, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device.

Table 1 – OWASP top 10 IoT vulnerability list [20]

Other institutions besides OWASP also confirm that brute force attacks on authentication mechanisms (number 1 issue in Table 1) by far still seem to be the most prominent attack vector for infecting IoT devices with malware. Kaspersky lab has published an article which states that in the second quarter of 2018 over 75.40% of infection attempts on their research-honeypots were conducted by brute-forcing the authentication of Telnet services. 11.59% of all attack attempts were made against the SSH service. Only 13.01% of all attacks made use of other attack vectors [21].

An article from F5 Labs, regarding their research on the most commonly attacked services in IoT devices, describes different – but equally relevant – results. Instead of Telnet the article states that brute force attacks are the global number one attack type on IoT, followed by HTTP. In F5 labs´ research, attacks on Telnet are found to be on the third place. They state that the reason for this might be the shift to SSH for remote administration of devices [22].

3.1.2 Covered Vulnerabilities

Based on the OWASP vulnerability list described in Section 3.1.1, vulnerabilities are selected along two considerations. Firstly, the attempt is made to only cover relevant issues which can be programmatically detected with a mobile solution within reasonable time and effort. Secondly, one of the major goals of the project is to create a solution which reveals problems that end-users can solve themselves, shortly after detection. This requirement is therefore also taken into account when selecting vulnerabilities to be covered in the project. It is assumed that generally even users with limited technical knowledge will be able to change authentication credentials or look for device-settings that let them disable services. To help users in understanding respective problems and services, the app provides brief explanations and additional links to online resources.

Weak, Guessable or Hardcoded Passwords

The OWASP IoT project´s top 10 list as well as the research of Kaspersky and F5 Labs are underlining the importance of securing authentication mechanisms against a multitude of threats, including numerous malware variants. The application created in the course of this thesis helps to quickly and reliably detect the use of standard credentials for services prominent with many IoT devices:

- Telnet and SSH authentication are chosen to be covered because they are actively exploited by IoT malware on a large scale.
- HTTP is chosen to be covered because it is generally very frequently used for the configuration web-interfaces of network devices.
- FTP is chosen to be covered because it is the most prominent service for file upload/download, which – if not properly secured – can result in a significant security risk.

If the respective service is making use of standard authentication credentials, an attacker in many cases might gain control over the device by employing simple brute force attacks. In terms of severity, weak authentication for Telnet and SSH services is generally viewed more severe than weak authentication for HTTP and FTP services. This is because Telnet and SSH access grants the attacker full control over the device in most cases. Gaining the same control over a device via its web configuration interface (HTTP) or via FTP (file upload / download) may often require some additional exploitation steps or might not be possible at all.

The check for standard authentication credentials is directly addressing the rank 1 security issue of the most recent OWASP IoT top 10 list as described in Section 3.1.1. It should also be noted that the two most prominent problems in the previous top 10 list released in 2014 are described as “Insecure Web Interface” and “Insufficient Authentication/Authorization” [23]. Both issues are partly addressed by the aforementioned functionality: Detection of weak authentication in Web Interfaces via HTTP and detection of insufficient authentication due to standard credentials in Telnet, SSH and FTP.

Insecure Network Services

With the app, users can determine if UPnP is active on a device and whether problematic profiles are used (“LANHostConfigManagement”, “WANIPConnection” and “WANPPPConnection”). This functionality is considering the rank 2 issue of the most recent OWASP top 10 list. The UPnP profile detection is included because of the following reasons:

1. The UPnP service is frequently used in IoT devices (especially routers).
2. UPnP has generally known to be problematic for a long time [24].
3. The service is often directly accessible from the Internet.
4. UPnP can often be disabled via a device´s web configuration interface, which means that the user can take care of the problem him/herself.

3.1.3 Not Covered Vulnerabilities

Searching for other issues described in the OWASP IoT vulnerability list (Section 3.1.1) – besides authentication vulnerabilities and potentially problematic UPnP profiles – proves to not be viable, for multiple reasons. In general, providing a high coverage of specific vulnerabilities does not scale as the number of different devices is steadily increasing. Additionally, testing a possibly large number of devices for many potential problems would in most cases result in disproportionally long durations for network scans to complete. By introducing excessively long scan times, the app´s user experience would suffer. As stated in Section 3.1.2, a major goal of the project is to reveal problems that users can solve themselves. More specific problems would often only be fixable by patches from the vendor or by applying workarounds that require considerably more effort and technical knowledge. Providing notifications about those kinds of problems would undoubtably have informational value. However, in terms of encouraging users with little technical knowledge to increase the security in their home networks themselves, the practical value of such information would be rather limited. In detail, the following vulnerabilities are not covered in this thesis' project.

Weak, Guessable or Hardcoded Passwords (Other than Telnet, SSH, HTTP and FTP)

Other services that require authentication are generally not as common as the four covered ones and can often be considered rather specific. Trying to attack more specific services upon detection would result in increased scan durations, even though there is less chance of finding standard credentials. Regarding the studies described in Section 3.1.1 it also becomes obvious that for IoT devices, authentication mechanisms of other services are not attacked as frequently. This means that brute-forcing Telnet and SSH authentication is generally seen as one of the most viable attack vectors, by malware authors.

Insecure Network Services (Other than UPnP)

Besides insecure authentication, network services might be susceptible to a large variety of other security problems. UPnP was specifically chosen to be covered due to its potentially negative security impact, its frequent use in all kinds of network devices and because the service is frequently accessible via the Internet. Furthermore users can often disable UPnP on their devices, meaning that they could easily fix the problem themselves in many cases. The combination of those properties which makes UPnP relevant for this project is simply not found in other services. This circumstance is the reason why the decision is made to not check other network services for security problems.

Insecure Ecosystem Interfaces

Trying to programmatically check the ecosystem outside of a device – such as web, backend API, cloud or mobile interfaces – is not feasible within the boundaries of this project, for obvious reasons. In order to implement such functionality, an enormous collection with information regarding all kinds of specific device-types and their fingerprints, together with their potential ecosystem interfaces would be necessary. Then additional data would have to be available which somehow describes a multitude of potential security problems that may arise when a specific device interacts with one of its ecosystem interfaces in one way or another. This kind of functionality implicates seemingly limitless complexity. However, it might be possible to detect lacking or weak encryption of a device´s network communication. For instance, detecting unencrypted traffic is possible without the necessity to possess detailed information about individual devices and their interactions with certain interfaces. However, this again is only possible by inspecting the device´s outgoing and incoming traffic – something that requires systems such as the ones described in Section 2.1.

Lack of Secure Update Mechanism

Similar to the previous issue it is not feasible to try to programmatically check whether a device can be securely updated or not. Trying to examine the points mentioned in the description of this issue from a blackbox perspective for a large amount of different device-types would also implicate seemingly limitless complexity. However, as mentioned in the previous issue, the general detection of unencrypted traffic with systems described in Section 2.1 could also lead to conclusions about whether or not updates get encrypted during delivery.

Use of Insecure or Outdated Components

This issue considers the security of a device´s system-internal software components. Trying to programmatically test devices for such specific vulnerabilities would imply in-depth knowledge of a great number of different devices, which is beyond the scope of this thesis.

Insufficient Privacy Protection

Issues of this kind are very specific to a device and its ecosystem and would be extremely hard to programmatically detect on a general level from a blackbox perspective. The fact that those kind of issues also consider the security of the device´s ecosystem, implies almost limitless complexity.

Insecure Data Transfer and Storage

This issue considers the lack of encryption or access control of sensitive data anywhere within the ecosystem. While this is very hard to detect from a blackbox perspective for data that is at rest or during processing, it might be possible to detect unencrypted data in transit with systems described in Section 2.1.

Lack of Device Management

This issue describes primarily organizational problems. Trying to detect those problems programmatically from a blackbox perspective for a wide range of devices would not make any sense whatsoever.

Insecure Default Settings

Issues regarding default settings – besides the detection of default authentication credentials – are very device specific and therefore very hard to programmatically detect for a wide array of different devices.

Lack of Physical Hardening

Security problems of this kind are primarily based on the device´s physical features and cannot be programmatically detected from a blackbox perspective. An example for such issue could be a device that provides a USB socket, through which an attacker could locally supply malicious code.

3.2 Network Visualizations

In the following sections possibilities are mentioned for visualizing a computer network including security relevant information. Furthermore the selected visualization for the app´s graphical mode is discussed.

3.2.1 Visualization Possibilities and Related Work

The most natural visualization types for displaying computer networks generally seem to be force-directed graph layouts. This is concluded by multiple publications dedicated to visualizing computer network topologies and network traffic which are making use of this graph layout. Examples include the master thesis of Pauline Gomér and Jon-Erik Johnzon [25] and papers by Matthew Dean and Lucas Vespa [26], Roberto Tamassia et al. [27] and Florian Mansmann et al. [28]. Yarden Livnat et al. [29] suggest a sophisticated, alternative approach. It combines a force-directed graph layout with radial visualizations in order to display what security events were triggered for which device at a certain time. All visualizations displayed in the aforementioned papers are either too complex for novice users or lack visual appeal. This essentially makes them unsuitable for a solution that is primarily targeted towards novice end-users with limited technical knowledge.

Finally, work of Philip A. Legg [30] describes network visualizations that are simple, appealing and specifically targeted towards novice users. Due to the considerable relevance of those requirements in the network representation of the own app´s graphical mode, the paper is described in reasonable detail. This work describes a system which makes use of easily understandable graphical visualizations for network traffic. One of the main goals of the project is to increase the awareness of Non Expert Users (“NEUs”) towards suspicious traffic in their home networks. The visualization makes use of a graph layout, created with the D3 JavaScript library.
It depicts devices as nodes in circular shape and connections between devices as directed edges. Ports are displayed as smaller circles which are concentrically positioned around devices. Additional visual channels like color, shape and size are used for depicting attributes such as volume of activity (amount of sent data), communication between nodes and types of ports used by a host, as seen in Figure 5.

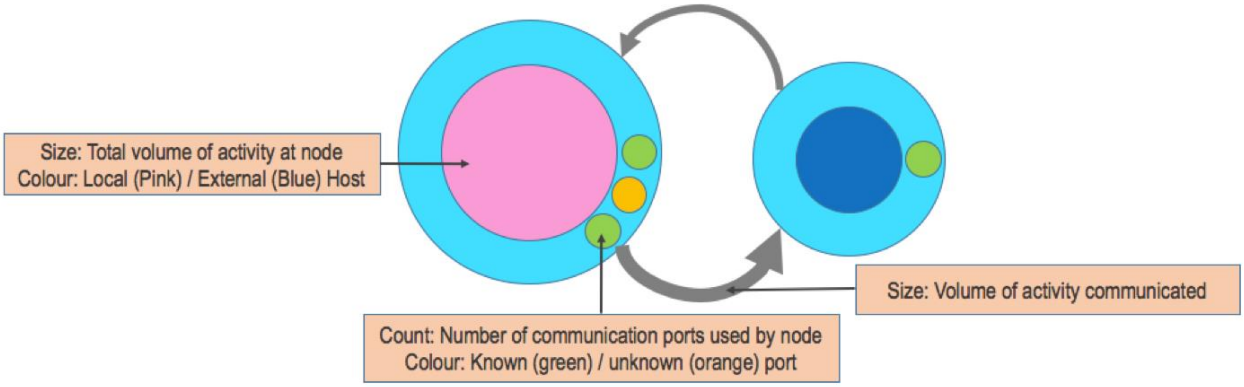


Figure 5 – Device, port and communication visualization [30].

The paper describes one case study where the system is utilized for visualizing the quantity of secure and insecure communication between devices in a large home network (pink-colored nodes) and external hosts (blue-colored nodes). Figure 6 shows a screenshot of the visualization directly from the tool. The used home network consists of two iOS devices, two Android devices, one Windows device, one Apple Mac and a router. For being able to visualize the security of certain communication protocols, port colors are extended by a traffic light scheme. This scheme makes use of the colors red, green and orange. The HTTP protocol (port 80), which is unencrypted and therefore considered insecure, is represented by a red-colored circle. The Secure Hypertext Transfer Protocol (HTTPS) (port 443), which is encrypted and therefore considered secure, is represented by a green-colored circle. All other ports are colored orange, signaling uncertainty regarding the level of communication-security.

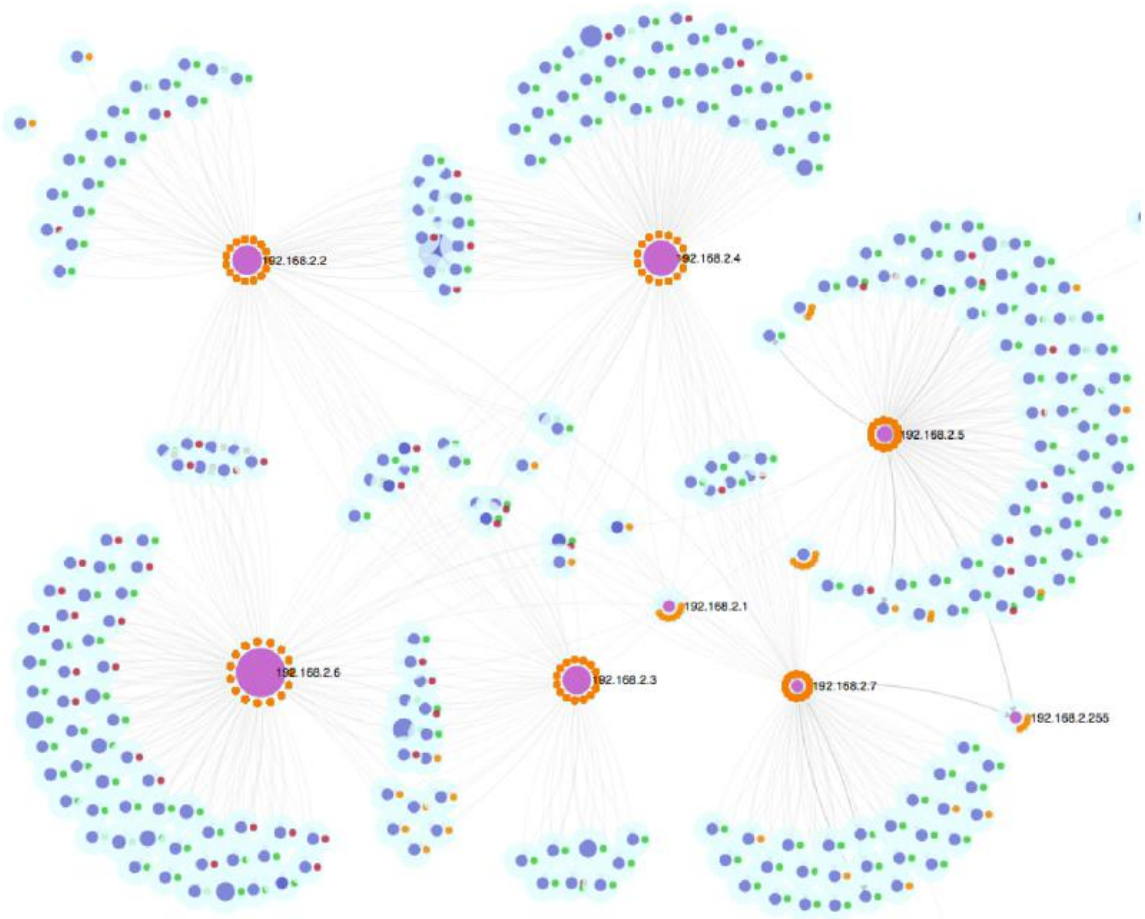


Figure 6 – Visualization of secure and insecure communication between local (pink-colored nodes) and external devices (blue-colored nodes) [30].

Finally the paper mentions that five test-users were selected to interact with the example from the afore-mentioned case-study. The feedback received from this informal consultation was generally positive, with users describing the visual interface to be bright, colorful and inviting. They also understood the security implications behind the traffic-light scheme, finding it simple and intuitive.

3.2.2 Selected Visualization

Due to the similar topic, the positive user feedback and general visual appeal of the solution described in Legg´s paper, some of the aspects mentioned before are also used in the app´s graphical mode. Those include the circular shape of device-nodes and the circular and concentrically positioned port-visualizations. Instead of depicting network traffic via directed edges, all device-nodes are simply connected to the network´s wireless router-node via undirected edges. The router-node is located in the middle of the screen and is slightly larger than the device nodes which are concentrically positioned around it. This should visualize a simplified version of the basic concept behind a computer network: A routing-capable device connects all other devices with each other, enabling communication within the network. Screenshots of the implemented visualization can be found in Section 4.2.5.

3.3 Traffic-Light Scheme and Security States

For appropriately depicting the security state of devices and ports for graphical network visualizations, an adaption of the traffic-light scheme described in Section 3.2.1 is used. This traffic-light scheme is not only put to use in the graphical application mode, but also in other application areas. The use of different colors and symbols is also always combined with a security state and a simple prioritization mechanism. The security state with the highest priority that applies to at least one device during a scan will in the end be applied to the entire network. The complete scheme is listed in Table 2 and the specific use of it in certain parts of the application is described in Section 4.2.2.

State name	Priority	Color and symbol encoding (graphical mode)	Color encoding (textual modes)	Applying condition
Neutral	0	Light blue for device-nodes and black for ports.	Grey background for list elements.	No problems were detected for the device. This does not mean that the device is “secure” though, hence the state “Neutral”.
Validation needed	1	Orange for device nodes and ports. An additional icon is connected to each orange port, visualizing an attack possibility.	Orange background for list elements.	One or more potentially problematic UPnP profiles were detected.
Insecure	2	Red for device nodes and ports. An additional icon is connected to each red port, visualizing an attack possibility.	Red background for list elements.	The use of standard credentials for one or more of the following services was detected: HTTP, SSH, Telnet or FTP.

Table 2 – Traffic-light scheme in combination with security states.

Examples for the Assignment of Security States to Devices and Networks

1. During the scan of a network which contains three devices A, B and C, the app detects that the device B is using a potentially problematic UPnP profile. The security state “Validation needed” (priority 1) is therefore assigned to the device B. No problems are detected for the devices A and C, which means that the security state “Neutral” (priority 0) is assigned to both of them. Out of all devices within the network, the “Validation needed” security state of device B has the highest priority. This means that the security state of the entire network is set to “Validation needed” at that point.
2. At a later time the same network is scanned again. The app detects that device B is using the same UPnP profile, which again results in a “Validation needed” (priority 1) security state for the device. Also again, no problems are detected for the device C which results in the same “Neutral” (priority 0) security state. However at this scan the app detects that device A is using standard credentials for the Telnet service. The state “Insecure” (priority 2) is therefore assigned to A. Out of all devices within the network, the “Insecure” security state of A has the highest priority. This means that the security state of the entire network is now changed to “Insecure”.

Assigning the “Insecure” security state to the entire network, even if just one out of potentially hundreds of devices is found to be actually insecure, might appear exaggerated. However, this decision is justified if taken into consideration that just one insecure device leads to a potential foothold within the network and might enable attackers to compromise more devices.

3.4 Historical Data and Visualization

In the following sections, frequently used chart types for displaying network security related data over time are described. Furthermore the history chart-types which are chosen for the app are explained.

3.4.1 Commonly Used Chart Types in Network Security Related Applications

One of the app´s main features is the possibility to view the results of scans for an entire network via three different interactive history charts. The charts should mainly provide an overview on the overall development of the network (missing-, new-, previously-detected devices) and the network´s security state over time. The displayed chart types should be well known but most importantly also be easily understandable. Line charts, bar charts and scatter plots generally seem to be good choices for visualizing the quantity of one or more properties of a dataset over time. To examine this statement – especially regarding network security related topics – the presentation of timeline-based data within Graphical User Interfaces (GUIs) of several common Intrusion Detection (IDS)-, Intrusion Prevention (IPS)-, and Security Information and Event Management (SIEM)-Systems is considered (Table 3).

Many of those systems allow users to customize how data are presented. This is especially true for systems that are commonly coupled with the “Elastic Stack”, such as the “Suricata” IPS/IDS [31]. The Elastic Stack describes the Logstash, Elasticsearch and Kibana technologies for the collection, analysis and presentation of data [32]. Kibana (the presentation component) allows a very high degree of customization [33]. The information in Table 3 therefore merely provides a rough outline on how time-based data is commonly visualized in popular network security related systems. This is mostly concluded by information from vendor websites (screenshots of the systems).

Product name	Type	Bar charts	Line charts	Scatter plots
SolarWinds Log & Event Manager [34]	SIEM	Yes	Yes	No
OSSIM [35]	SIEM	Yes	Yes	No
ArcSight Enterprise Security Manager [36]	SIEM	Yes	Yes	Yes
Splunk Enterprise Security [37]	SIEM	Yes	Yes	Yes
IBM QRadar [38]	SIEM	Yes	Yes	No
Suricata in combination with Logstash / Kibana [31].	IDS / IPS	Yes	Yes	Yes
SELKS (Suricata based) [39]	IDS / IPS	Yes	Yes	No
Cisco Firepower [40]	IDS / IPS	Yes	Yes	No
Snorby (Snort based) [41]	IDS / IPS	Yes	Yes	No
Cisco Meraki IDS / IPS [42]	IDS / IPS	Yes	Yes	Yes

Table 3 – Overview on common chart types for displaying time-based data within popular SIEM, IDS / IPS systems.

In addition to this overview, the master thesis of Alexander Zahariev [43] which is examining graphical user interfaces for intrusion detection systems in telecommunication systems is coming to a similar conclusion regarding line charts. Five out of six examined intrusion detection systems were utilizing line charts in their graphical user interface.

Another interesting approach of visualizing network-data over time is proposed by John R. Goodall et al. [44]. Their work describes visual network traffic analysis by making use of a tool called “Time-based Network traffic Visualizer” (TNV). The tool implements a visualization matrix which positions time-intervals on the x-axis and all hosts on the y-axis. The number of network packets sent by a host in a time interval is encoded by a colored box.

3.4.2 Selected Chart Types

From Section 3.4.1 it can be concluded that bar charts and line charts are commonly used for visualizing the quantity of one or more properties of a dataset over time. Therefore the decision is made to use those two chart types for two of the three different charts that are implemented for the app´s history functionality. Both of them use the quantity of certain devices as values for the y-axis and points in time as values for the x-axis (when a scan took place).

The third history chart is based on the more sophisticated visualization used by the TNV tool which is described in the paper by John R. Goodall et al., as mentioned in Section 3.4.1. The app employs an adapted version of their visualization approach. It uses scan-times on the x-axis, lists all devices on the y-axis and encodes device-properties (such as “secure” or “insecure”) at a certain network-scan with colored boxes. Compared to charts that discuss the quantity of devices with certain properties, this visualization has the advantage of providing more information on the security state of individual devices.

4. Implementation

In this chapter´s first subsection insights are given into the application´s structure and general relevant technical topics. In the second subsection further elaborations on implementation details and results are provided.

4.1 System Architecture: General Structure of the Application

In this section general technical and structural topics are discussed. Those include the platform, a brief overview on the UI components, application modes, storage and history functionality, multi-threading and the libraries used for this project.

4.1.1 Platform

The project is implemented as native Android application in Android Studio 3.1.3 and is compiled against the minimum Software Development Kit (SDK) version 16 (Jelly Bean) and the target SDK version 24 (Nougat). Due to extensive use of multithreading, it is preferred that the app is used with hardware which at least makes use of quad-core processors. Scans can then take place within reasonable time which increases with the size of the network.

4.1.2 User Interface

General UI Component and Navigation Overview

Figure 7 illustrates the application´s most basic components and navigation possibilities when the app is set to the “Expert”-application mode. In the two other modes, the appearance is different, yet the program flow is similar.

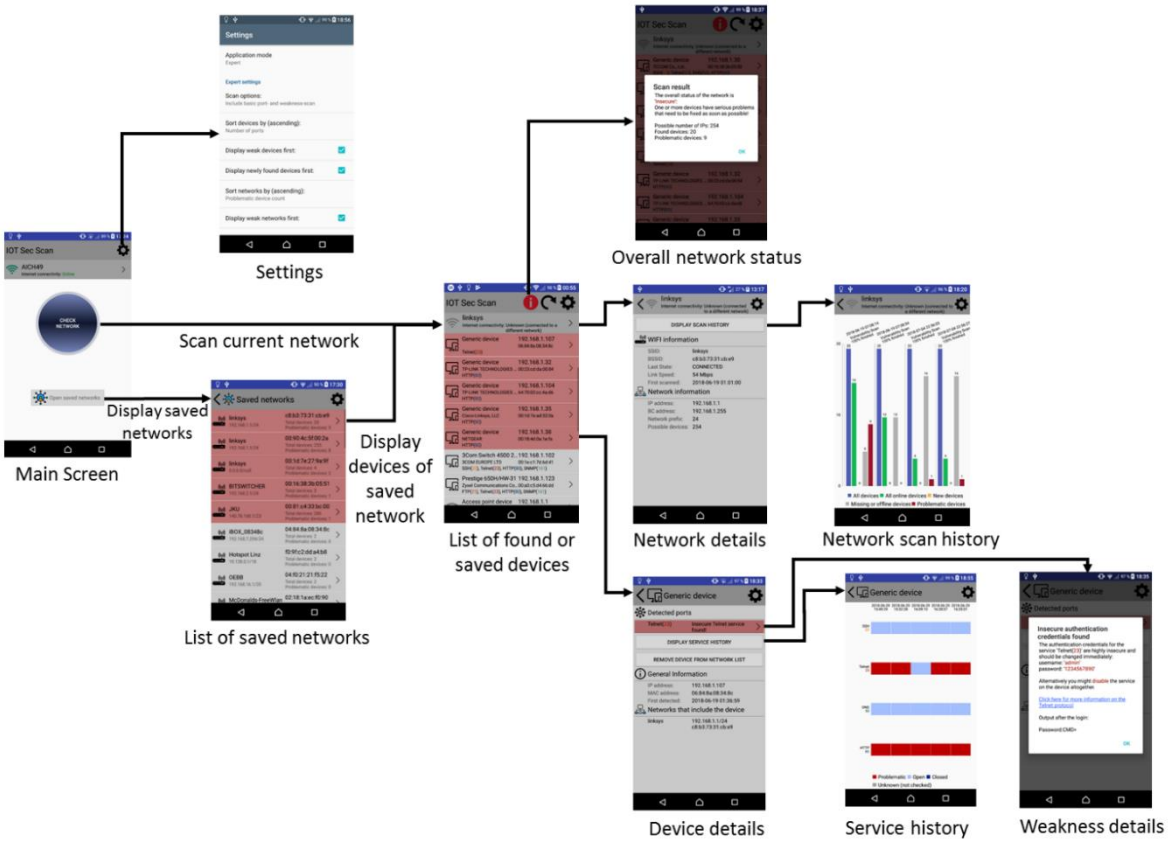


Figure 7 – General UI component and navigation overview

Application Modes

One of the major features are the three application modes which are implemented in order to make the application easily usable for audiences with different degrees of technical knowledge. One of the project´s main objectives is to provide application modes that present security-relevant information as comprehensible as possible – especially for novice users. This goal is achieved by reducing information that is considered to be too complex and by providing a graphical representation of network topologies and inherent vulnerabilities.

The three different application modes are:

- Expert mode: This mode is targeted towards users having an understanding of fundamental networking concepts such as IP-addresses, MAC-addresses, network services and ports. With this mode the user has unrestricted access to the application´s entire functionality and is able to view any information gathered on networks / devices.

- Normal 1 mode (reduced information): The Normal 1 mode mostly consists of functionality similar to that of the Expert mode. However, the data presented in this mode are greatly reduced by hiding information that might not be sufficiently comprehensible for novice users, such as IP-addresses, MAC-addresses, port numbers or network services.
- Normal 2 mode (graphical): The Normal 2 mode differs from the Normal 1 mode in the way devices within a network are being presented. Instead of merely textually listing devices, this mode follows a graphical approach visualizing the topology and possible vulnerabilities of a network.

Furthermore each application mode provides a unique chart, visualizing a network´s scanning history. The application also provides a chart for visualizing a device´s service history – however this feature is only available in the application´s "Expert" mode, as network services have generally found to be a too complex topic for novice users.

It is important to note that there is no functionality in place which helps users in choosing an application mode that would fit best for them. However, finding a fitting application mode can be considered a rather easy task. If the user considers him/herself as technically experienced then the Expert mode is probably a good choice. If that is not the case, then the user might try both Normal modes. Technically inexperienced users that rather prefer textual information will then probably chose the Normal 1 mode. For users who like to explore information visually, the Normal 2 mode might be a good choice. It also has to be noted that users do not have to restrict themselves to use the app in just one particular mode. Via the settings menu it is possible to freely switch between all modes in any part of the app (except when a network scan is taking place).

4.1.3 Storing Network and Device Information

Scanned networks and their devices are automatically saved during initial scans. During re-scans respective database records are updated with information retrieved from the device detection, information gathering, port and vulnerability scanning functionality.

The application makes use of the following data objects:

- Device
- Network
- OUI – Organizational Unique Identifier for identifying a device´s Network Interface Controller (NIC) vendor name via its MAC address
- ServiceEntry (used for services detected during portscans)

It has to be noted that the device and network data objects are being reused for holding data from device and network snapshots (as there are only minor differences between saved devices / networks and saved device / network snapshots).

Database Model

The database model consists of the following tables:

- Device table: This table contains all device specific information, such as the MAC address, open ports, the device´s host name, SNMP and UPnP details, and security specific information such as credentials found during HTTP, Telnet, SSH or FTP brute-force attempts.

- Network table: This table contains all network specific information, such as the SSID, the Basic Service Set Identifier (BSSID), the gateway IP address and Internet Service Provider (ISP) information in cases where the network is connected to the Internet.
- Network_to_device table: This table is required for implementing the N:M relationship between networks and devices. It contains information which is unique to a device´s relationship to a specific network, such as the IP address that the device received, when it has been connected to the network the last time.
- Device_snapshot table: This table contains the same information as the device table, with minor differences. For snapshots, the device´s IP address is directly stored in the device_snapshot table. This is because there is an 1:N relationship between the network_snapshot and the device_snapshot table, instead of a N:M relationship as it is the case with the network and device tables.
- Network_snapshot table: This table contains the same fields as the network table and two additional ones. The scan type, which indicates what kind of network-scan took place when the network snapshot has been created and the scan percentage which indicates whether the scan has been completed or not.
- OUI table: This table contains datasets which consist of the three OUI (Organizational Unique Identifier) bytes and the corresponding NIC vendor names. For initially filling this table, a formatted version of the OUI-list on the Institute of Electrical and Electronics Engineers (IEEE) website is used [45].
- Service table: This table contains service-specific information such as a service´s name, port number, open frequency and description. For initially filling this table, a formatted version of the well-known Nmap project´s service list is being used [46].

Figure 8 illustrates the relationships between the database tables mentioned above. For the sake of brevity not all attributes are listed in this figure.

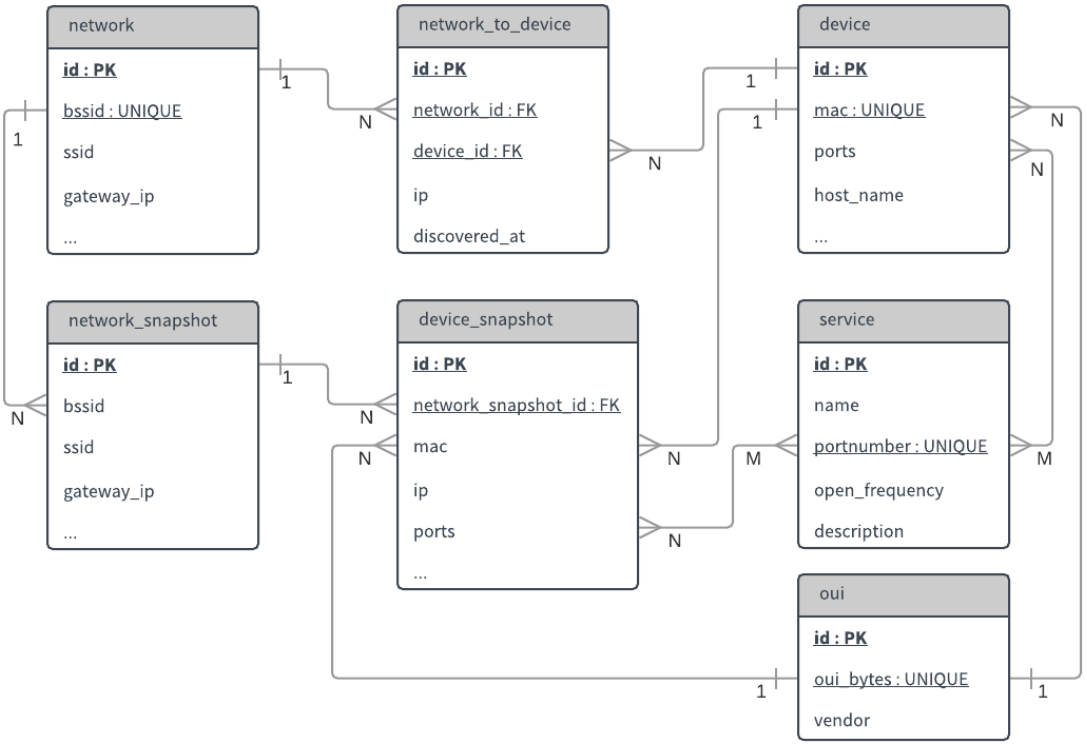


Figure 8 – Database model

Notes on Database Design Decisions

- It is important to clarify the difference between the network and network_snapshot and the device and device_snapshot tables. The network and device tables are used for saving an object´s current state, which means that the corresponding data is updated when changes are detected. Snapshots however are used for storing final versions of objects (networks or devices) at a certain point time. When changes of the object are detected, previous snapshots are not updated – instead a new snapshot is created, which guarantees the traceability of the object over time.
- Not making the unique BSSID and MAC attributes the primary keys of the network and device tables is a deliberate decision to remain consistent with the incremental integer-type primary-key creation for each table. Furthermore, for the network_snapshot and device_snapshot tables, using the BSSID and MAC properties as primary keys would not work, because those properties are not unique for records of those tables. For instance, multiple snapshots of the same device might obviously contain the same MAC value.
- It is a deliberate decision to not resolve the device to service and the device_snapshot to service N:M relationships via a dedicated junction-table – as seen with the network to device N:M relationship. Firstly, unlike the network_to_device table which contains the IP- and discovered_at-properties, additional attributes (besides the foreign keys) for possible device_to_service and device_snapshot_to_service tables would not be required. Secondly, introducing more junction-tables would increase the model´s complexity, which would result in more complex Create-Read-Update-Delete (CRUD) Structured Query Language (SQL) statements. Therefore the decision is made to save a list of open ports directly for each device and each device snapshot and to use this list to load corresponding information from the service table via a service´s port number (unique), only if necessary.
- For resolving the 1:N relationships between the device and device_snapshot and the network and network-snapshot tables, the database model refrains from using dedicated device and network foreign keys in the network_snapshot and device_snapshot tables. If necessary, a device´s snapshots might be loaded by simply looking up device snapshot records that contain the same MAC address as the device. Likewise, if a network´s snapshots should be loaded, a database lookup for network snapshots containing the corresponding BSSID, takes place.

4.1.4 History

For being able to observe the development of networks and individual devices over time, history features are implemented. For every full or partial network scan, a snapshot of every detected device is saved. The entire network-history can be displayed via three different chart visualizations (based on the currently selected application mode) as described in Section 4.2.4.

Device snapshots are not only created during full or partial network scans but also during vulnerability scans (including basic-port scans) and extended port scans, which both can be triggered for individual devices. A device-service-history chart might be displayed via a device´s detail page as briefly described in Section 4.2.5.

4.1.5 Multi-Threading

The application´s AsyncTask components responsible for performing detection, information gathering, port- and vulnerability scanning operations make heavy use of multithreading. For efficient handling of threads the Android ExecutorService class is used – mostly with thread pools of a fixed size.

Synchronization after executing any asynchronous code is achieved by making use of the Android CountDownLatch class. Likewise asynchronous operations are cancelled in time by regularly checking the corresponding AsyncTask´s state during longer running operations.

For scanning activities the app uses thread pools with a large number of threads. Thread pools of about 100 threads have been successfully tested. This is because individual actions mostly consist of operations that are not very CPU-intensive, such as waiting for a ping-request / authentication attempt / etc. to complete.

4.1.6 Used Libraries

The application makes use of a range of different external libraries which are listed in Table 4.

Library name	Vers.	General usage scenario	License
OK HTTP [47]	3.4.1	Used for various HTTP-Requesting functionality, including the HTTP-authentication brute force.	Apache 2.0
Sadun Telnet Client library [48]	1.13	Used for Telnet brute force functionality.	GNU Lesser General Public License (LGPL)v2
JSch – Java Secure Channel Library [49]	0.1.54	Used for SSH brute force functionality.	"Berkeley Software Distribution (BSD) style license"
Apache commons net [50]	3.6	Used for FTP brute force- and networking-functionality, such as the extraction of all possible IP addresses from a network.	Apache 2.0
Jsoup [51]	1.10.2	Used for processing the Extensible Markup Language (XML) inside of UPnP Service Control Point Definition (SCDP) documents.	Massachusetts Institute of Technology (MIT) License
The Java Common Internet File System Protocol (CIFS) Client Library [52]	1.3.18	Used for extracting Netbios information.	LGPL 2.1
SNMP4J [53]	2.5.8	Used for extracting SNMP information.	Apache 2.0
Android MP charts library [54]	3.0.3	Used for creating network- and device-history charts.	Apache 2.0
D3.JS (JavaScript charting library) [55]	3	Used for creating a graphical network-topology visualization.	BSD 3-Clause “New” or “Revised” License

Table 4 – Used libraries

4.2 User Interface

In this chapter´s first subsection, several UI areas are grouped by making use of a “UI components and navigation” visualization. In the next subsection, elaborations are given on the use of the traffic-light and security state scheme in the user interface. In the remaining sections descriptions on the implementation of all UI areas are provided in more detail.

4.2.1 UI Components and Navigation

The user interface is implemented by a set of Activities in combination with appropriate XML layout files. However in some Activities UI-elements are also being created dynamically, directly via Java code provided by numerous helper classes. Easily understandable icons for the different areas of the application are selected from the Website https://www.flaticon.com/; some are adapted.

In addition to the general UI component and navigation overview shown in Figure 7, Figure 9 provides a different UI visualization. It takes the UI changes into account that might occur, based on the selected application-mode. Furthermore the figure arranges the user interface´s components into three different areas (general, network- and device-specific). Doing so helps to categorize the UI components described in the subsequent chapters.

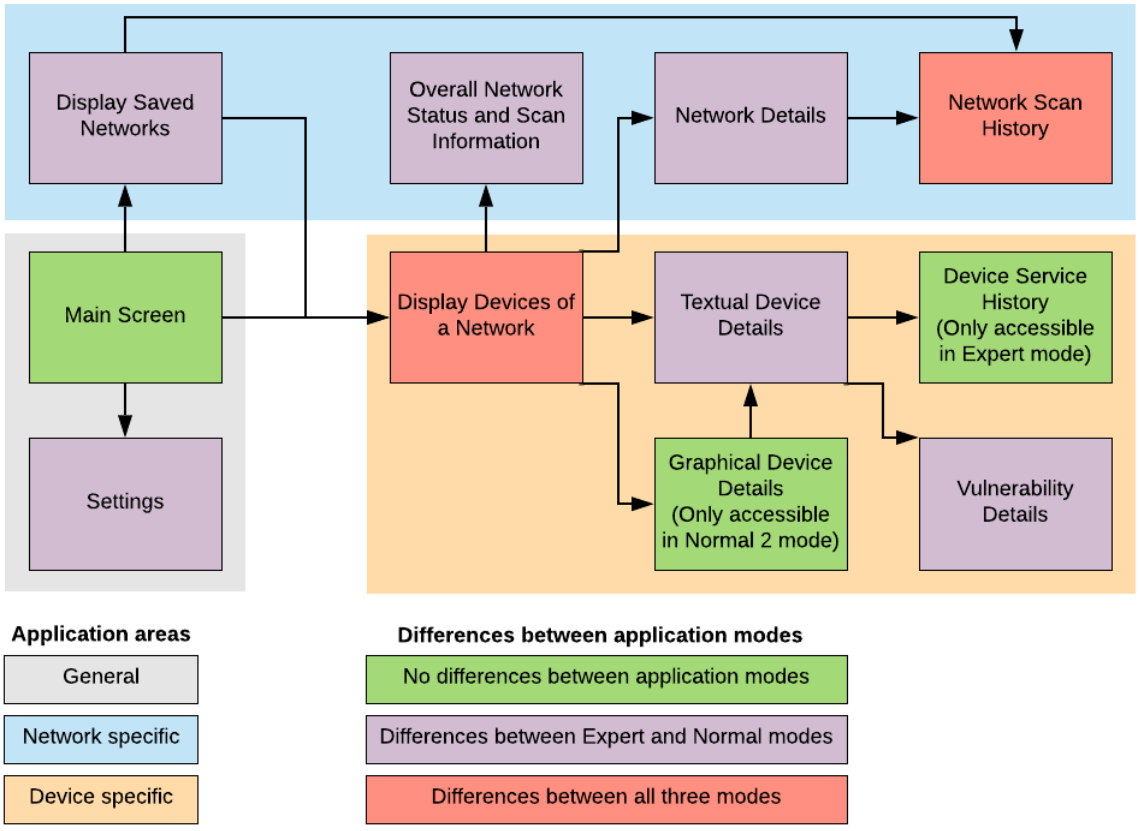


Figure 9 – Schematic view of UI components and navigation regarding application areas and differences between application modes.

4.2.2 Uses of the Traffic-Light and Security State Scheme

Table 5 lists all the UI components that contain areas which make use of the traffic-light and security state scheme described in Section 3.3.

UI component	Affected areas
Network related (Section 4.2.4)	
Display saved networks	Colorization of network list-items
Overall network state and scan information	Colorization and description of the security state
Device related (Section 4.2.5)	
Display devices (list-based)	Colorization of device list-items
Textual device details	Colorization of port list-items
Display devices (graphically)	Colorization of device nodes and port-circles and use of corresponding vulnerability icons.
Graphical device details	Colorization of device node and port-circles and use of corresponding vulnerability icons.

Table 5 – UI components that make use of the coloring scheme.

4.2.3 General UI Areas

In the following sections, information is provided on the main-screen and the settings. Both of those UI-areas can be considered general as they provide functionality which is important for features regarding both networks and devices.

Application Icon and Main Screen

The first UI screen that gets displayed when opening the application via its icon (Figure 10) is the main screen (Figure 11). It lets the user open the settings menu via an icon on the top right, scan the wireless network that the mobile device is connected to or lets the user open the previously scanned, saved networks. The main Activity performs a network connectivity check when the Activity is started and also when the user tries to perform a network scan. If the connectivity check fails, the main screen provides corresponding feedback (Figure 12). Upon the first start of the application, the main Activity also triggers the initialization of the Service- and OUI-database tables. This is done automatically and temporarily prevents the user from performing any actions by blocking the user interface with an appropriate “loading”-overlay. The database initialization routine makes use of multithreading and therefore usually finishes within seconds.

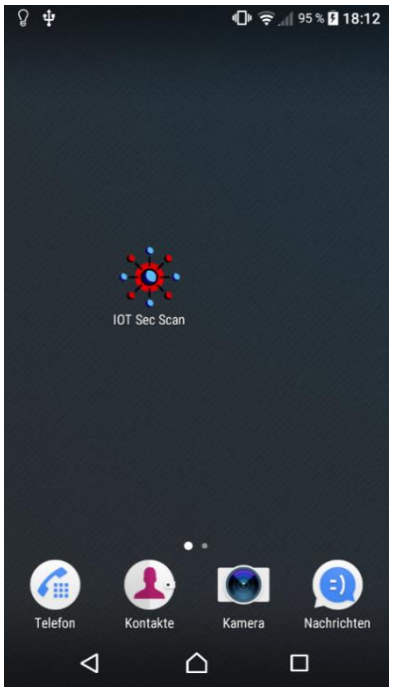


Figure 10 – Application icon

Figure 11 – Main screen when the device is connected to a network

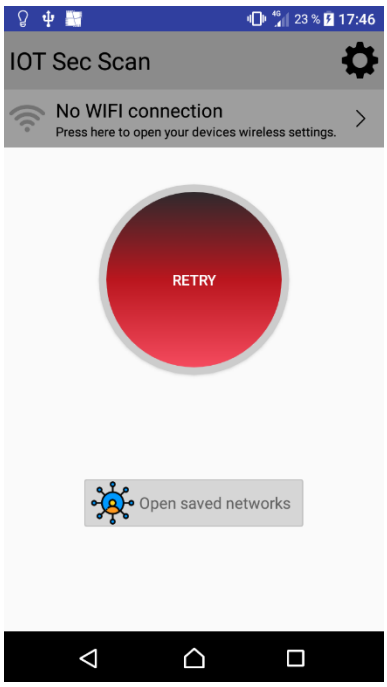


Figure 12 – Main screen when the device is not connected to a network

Settings

It is possible to open the settings menu from all application-screens via the “gear”-icon in the top right corner. However, the settings menu is not accessible while a network-scan is active (hidden settings-icon) or when the vulnerability details-, graphical device details- or overall network state-popups are opened (areas outside of the popup are not clickable). Which kind of settings are presented to the user, depends on the selected application mode.

When the Expert mode is selected (Figure 46) the user can change the way how device- and network-lists are sorted via several different attributes (Figures 47 and 48). For device-lists it is possible to order devices – in addition to the primarily selected sorting attribute – by two additional attributes:

- By the device´s security state (“weak devices first”).
- By information that indicates whether the device has been found in that network before (“newly found devices first”).

Likewise it is possible for networks to additionally order them by their security state (“weak networks first”). In the Expert mode it is also possible to change the application´s scan mode (Figure 49). The default-mode here is: “Device detection and basic port scan and weakness scan”. When the Normal 1 or Normal 2 modes are selected, the only available setting is the one for changing the application mode (Figures 50 and 51). All screenshots for this section can be found in the Appendix.

4.2.4 Network Specific UI Areas

In the following sections information is provided on UI areas that are more specific to networks.

Display Saved Networks

This Activity provides functionality for displaying saved networks, based on different application modes. Figure 13 depicts the Activity in the Expert mode and Figure 14 shows the Activity in the Normal modes. Table 6 illustrates the differences in terms of displayed information between the application modes. For enabling users to quickly reach other relevant application areas from this screen, an intermediate navigation step is introduced. If the user taps on one of the networks a popup menu appears (Figure 15). The first option allows displaying the network´s last state (listing saved devices) and the second option allows navigating directly to the network´s scan history.

Application mode	**Expert**	**Normal 1 & 2 (reduced information)**
Displayed information	SSID	SSID
	BSSID	-
	Access Point IP	-
	Number of total devices	Number of total devices
	Number of problematic devices	Number of problematic devices

Table 6 – Displayed information in the Saved Networks view

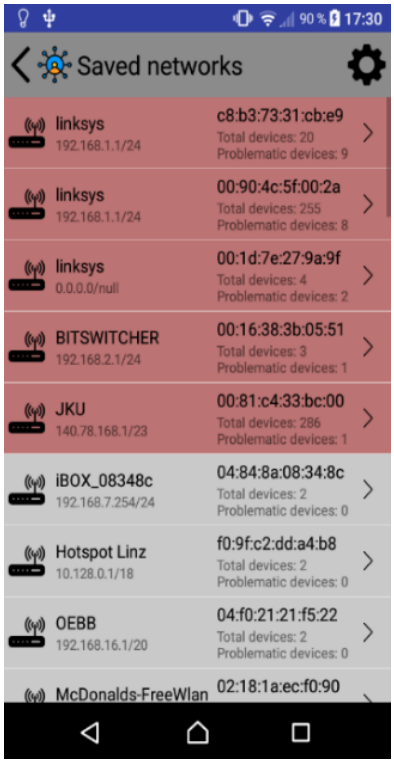


Figure 13 – Saved networks in the Expert mode

Figure 14 – Saved networks in the Normal 1 & 2 modes (reduced information)

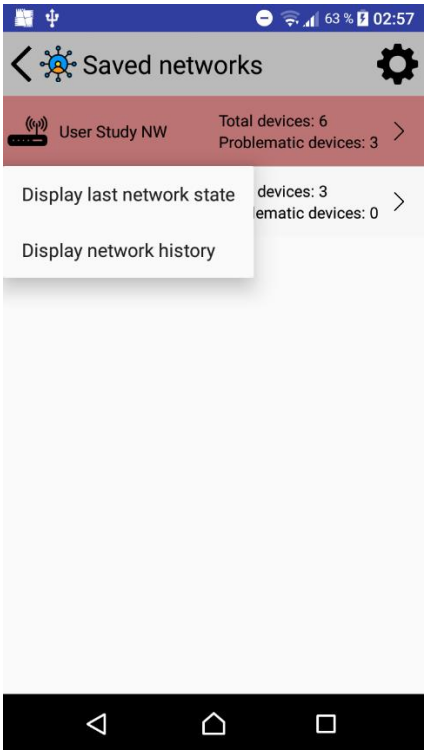


Figure 15 – Additional navigation possibilities

Overall Network State and Scan Information

The information popups displayed in Figures 16, 17 and 18 inform the user about the overall network state and result of a preceding scan. The only difference between the Expert and both Normal application modes is the description of the number of scanned IP addresses. In the Normal 1 and 2 modes the description says “Scanned device addresses” instead of “Scanned IPs”.

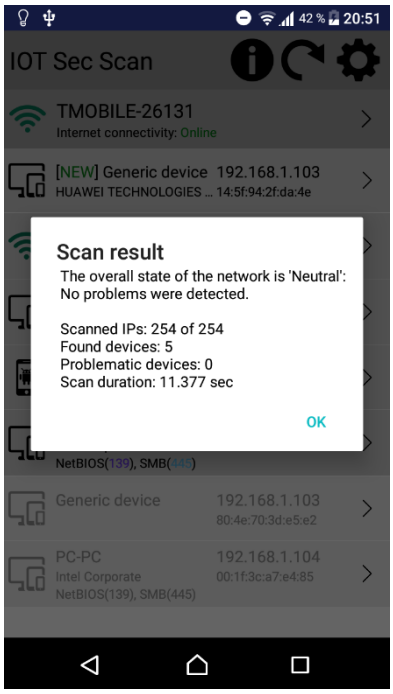


Figure 16 – Overall state of a “Neutral” network

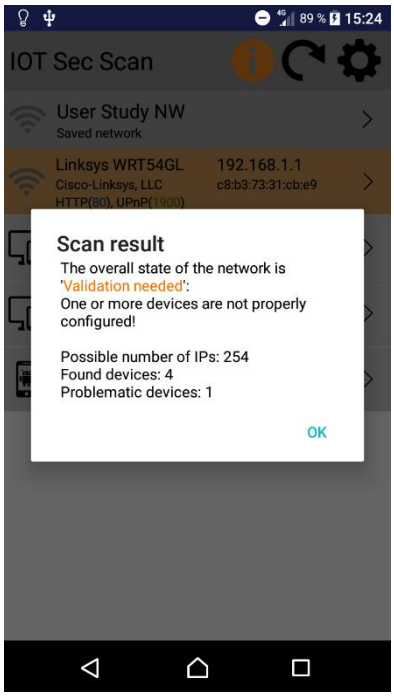


Figure 17 – Overall state of a network that requires validation

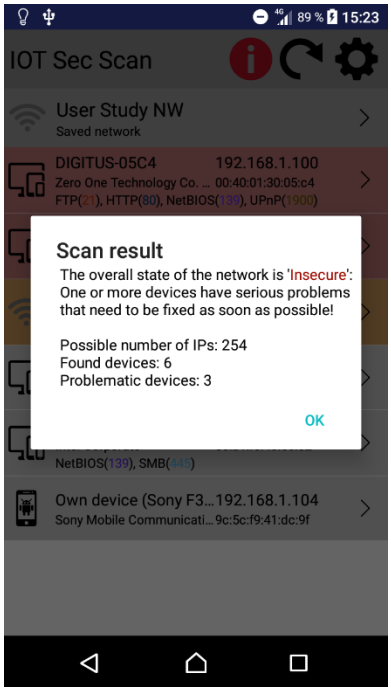


Figure 18 – Overall state of an “Insecure” network

Network Details

This Activity provides functionality for displaying network-information collected during the information-gathering phase of a network scan, based on the selected application mode (Figures 19 and 20). From this Activity the user also might access a network´s scanning history (application-mode independent). Table 7 illustrates the information that is being displayed in the Normal 1 and 2 application-modes in comparison to the Expert mode.

Application mode	Expert	Normal 1 & 2 (reduced information)
Displayed information	**Wi-Fi information**	
	SSID	SSID
	BSSID	-
	Network state	-
	Link speed in Megabits Per Second (Mbps)	Link speed in Megabits Per Second (Mbps)
	First scanned	First scanned
	Network information	
	Gateway IP	-
	Broadcast address	-
	Network prefix	-
	Total possible devices	-
	Provider information	
	ISP name	ISP name
	Organization	Organization
	External IP	-
	Country	Country
	Country code	-
	Region	-
	City	City
	Zip code	Zip code
	Latitude, longitude	-
	Time zone	Time zone
	Autonomous System (AS) number	-

Table 7 – Displayed information in the Network Details view

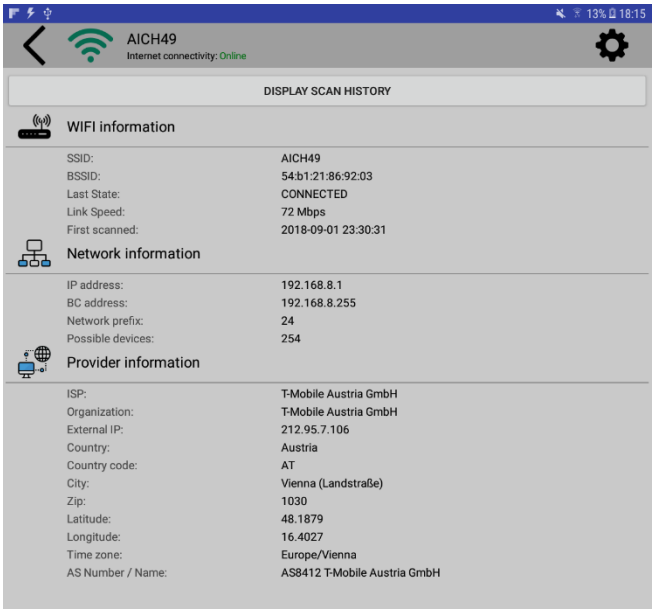


Figure 19 – Network details in Expert mode

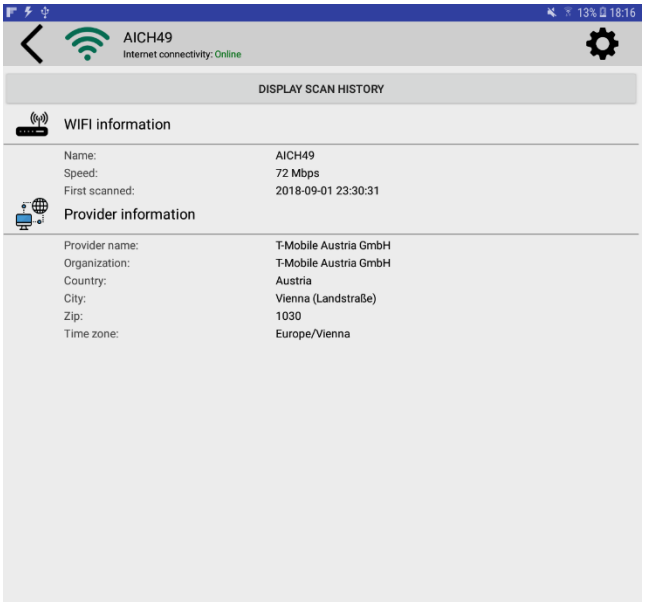


Figure 20 – Network details in Normal 1 & 2 modes

Network Scan History

This Activity provides functionality for displaying all the snapshots of a network´s previous scans. Snapshots are visualized via appropriate charts by making use of the “MPCharts” library [54]. The chart type that gets presented to the user, when viewing the network history, depends on the currently selected application mode. In addition to the reasons from Section 3.4, the types are chosen to match the general purpose of the respective application mode, which is described in the next sections.

Expert Mode

The Expert mode generally aims at providing detailed information. As a consequence a chart type is required which makes it possible to easily visualize the number of devices with different properties, for multiple network scans over time. Grouped bar charts are used where each of the bars indicates the number of devices in a group with a certain property. Groups that are used in this chart are:

- All devices
- All online devices
- New devices
- Missing or offline devices
- Problematic devices

This chart-type is quite common and has the advantage of letting users easily compare the quantity of devices from one scan to others. The bar chart makes it easy to detect an increase in the number of problematic devices from one scan to the next, for instance. Additionally, information about the exact time, type and completion state is added for each scan. Figure 21 depicts a screenshot of the described chart.

Normal 1 Mode

This mode aims at omitting possibly complex information, so the decision is made to only display the quantity of online devices over time, via a simple line chart. Additionally the chart visualizes another important aspect, which is whether at least one of those online devices were found to be vulnerable during a scan. This is done by simply coloring the respective data-point in red. A screenshot of the chart can be seen on Figure 22.

Normal 2 Mode

This mode is introduced to mainly provide some interesting, alternative visualization compared to the other modes. In accordance to that, a history-chart is chosen that would have a major advantage over the other charts. The chosen solution lists all device names on its y axis and the time when a scan took place on its x axis. The devices´ states after a certain scan took place are displayed by differently colored sections along the x-axis (time) on the respective y-position (device names). The visualization distinguishes between the following states:

- Problematic
- Found new
- Found existing
- Offline

This solution allows users to quickly spot the exact names of devices that caused problems during a certain scan. In order to find this information with the other charts, user interaction for directly opening individual snapshots, would be necessary. Figure 23 shows a screenshot of the described chart.

All of the charts are interactive and support horizontal scrolling (vertical scrolling only for the chart in the Normal 2 mode) and horizontal zooming for enlarging a certain chart-area. It is also possible to click on certain chart areas, which has different effects for the different chart types. When a user clicks on one of the bars from the Expert- or on one of the data-points from the Normal 1- charts, one or more device-snapshots for the respective selection are displayed via the “Display Devices” Activity which is described in Section 4.2.5. However, one bar in the Normal 2 – chart always represents one snapshot of a particular device. This means that if the user clicks on one of those bars, the respective device-snapshot is opened in the “Textual device detail” Activity, described in Section 4.2.5.

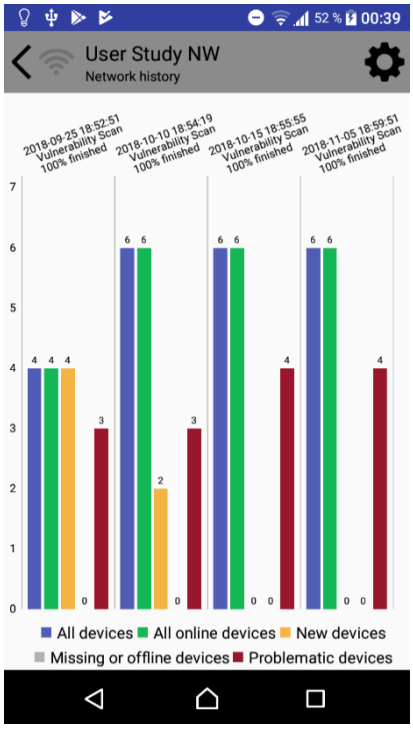


Figure 21 – Network history in Expert mode

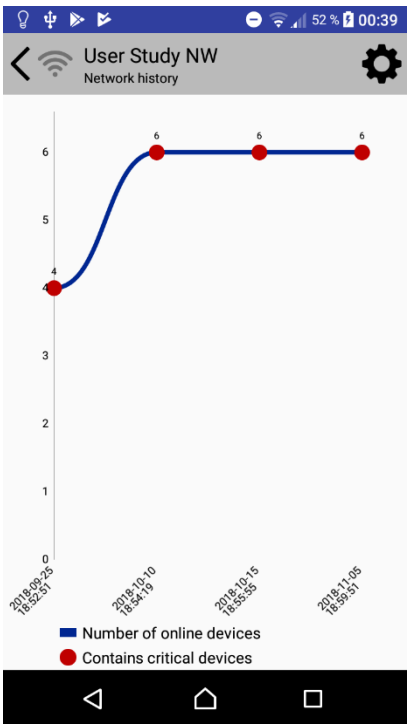


Figure 22 – Network history in Normal 1 mode

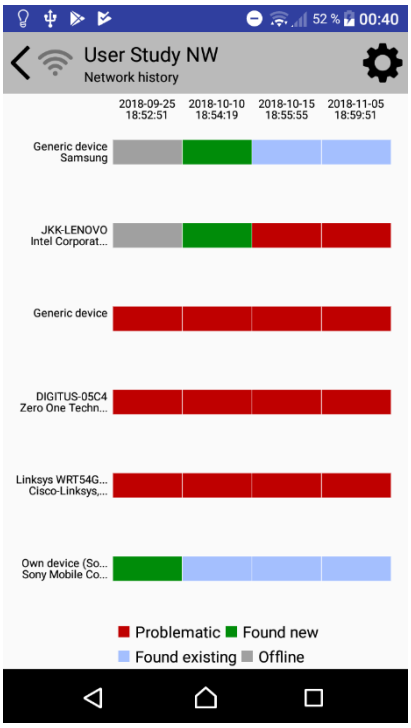


Figure 23 – Network history in Normal 2 mode

4.2.5 Device Specific UI Areas

In the next sections elaborations are provided on UI areas that are rather specific to functionality regarding devices.

Display Devices

This Activity is by far the most complex one and provides functionality for cancelling scans and triggering re-scans (application mode-independent). It most importantly displays information on devices during active scans, as well as saved devices or device snapshots. Figures 24, 25 and 26 show devices within a smaller home network in different application modes. Table 8 shows application mode dependent differences in the displayed information.

Application mode	**Expert**	**Normal 1 (reduced information)**	**Normal 2 (graphical)**
Displayed information	Device name	Device name	Device as node with a name
	MAC-vendor	MAC-vendor	-
	IP-address	-	-
	MAC-address	-	-
	Services including port numbers	-	Ports as smaller circles, positioned around the device node
	Basic device type indicated by icon	Basic device type indicated by icon	-
Action on device selection	Open textual device detail page in Expert mode	Open textual device detail page in Normal 1 mode	Open graphical device detail page

Table 8 – Displayed information and possible actions in the Device List view

In order to provide an alternative to merely displaying networks based on ListViews, a custom graphical network visualization for devices/vulnerabilities is created (Normal 2 mode). For the implementation it is necessary to use the “D3.js” JavaScript library [55]. This is because no Java charting library (like the one used for the history charts) exists, which allows the implementation of the desired visualization. The following list mentions the most important components which are employed to create the final visualizations:

- WebView components are used to integrate Web-based content – such as Hypertext Markup Language (HTML) pages – into the app.
- Two basic HTML pages (embedded within WebView components) are used to display an entire network or an individual node.
- Several JavaScript files containing D3 code and helper functions are loaded within the HTML files.
- A WebViewInterface is required for calling Java code from within JavaScript. In this case that is needed for opening the graphical device detail popup, after a user taps on an individual device in the network visualization.
- A DialogFragment component is necessary for displaying a graphical device detail popup containing a visualization and vulnerability-information for an individual device, as shown in the next section.
- Several images are used for displaying different vulnerability classes within the visualization.

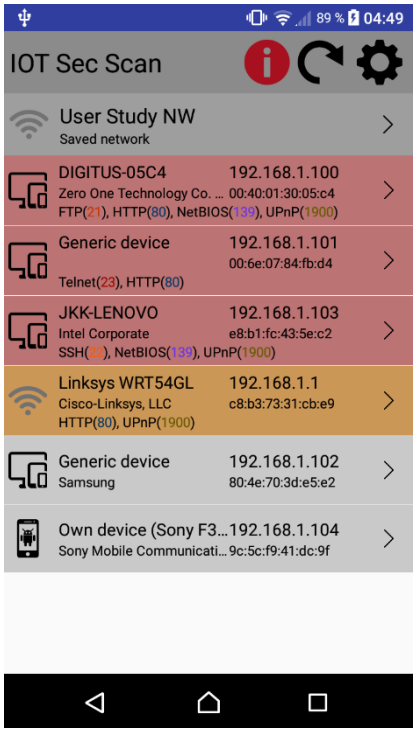


Figure 24 – Device list in Expert-Mode

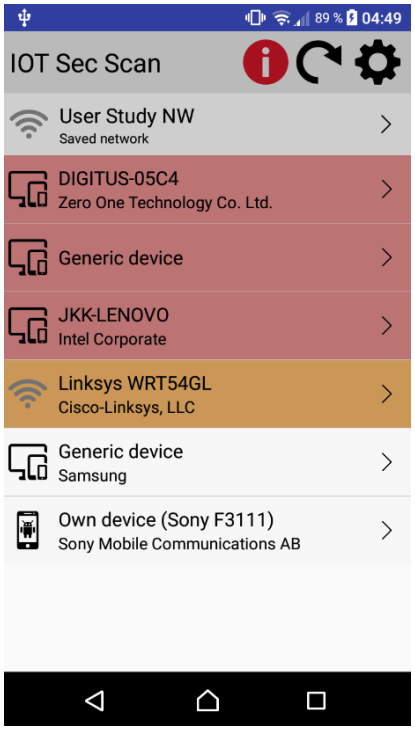


Figure 25 – Device list in Normal 1 mode

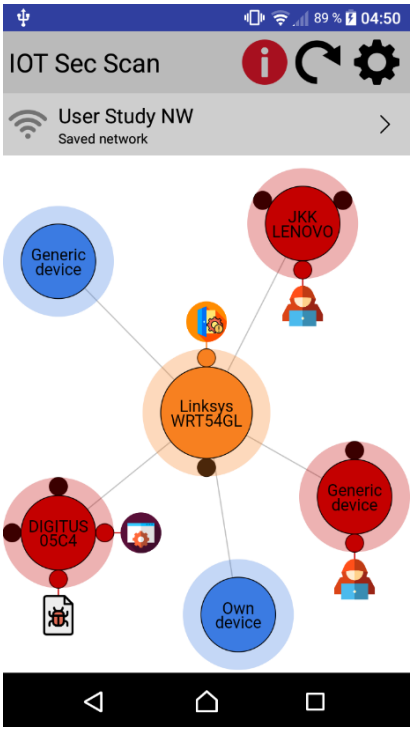


Figure 26 – Devices in graphical mode

Graphical Device Details

Graphical device details are presented in the form of a dialog. This dialog can only be accessed when clicking on one of the devices from the graphical device visualization and is therefore only available in the Normal 2 mode. The popup enlarges the selected device-circle and provides some additional explanations. Figures 27, 28, 29 and 30 show detailed information about the devices of the network topology depicted in Figure 26.

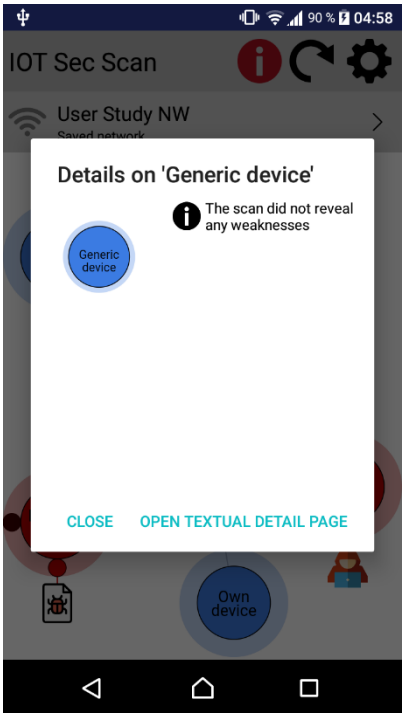


Figure 27 – Graphical details on a neutral device

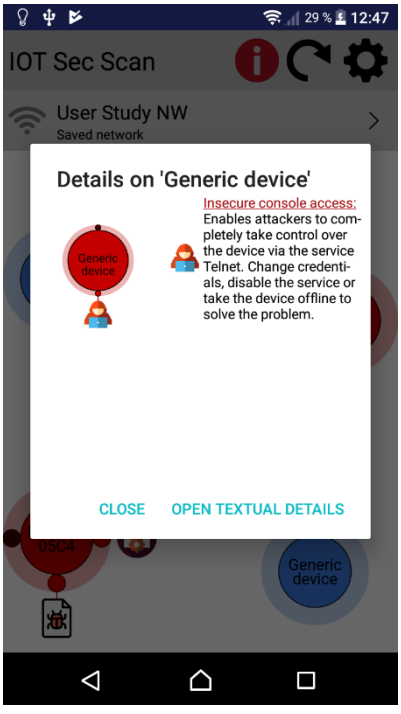


Figure 28 – Graphical details on a device with insecure console access

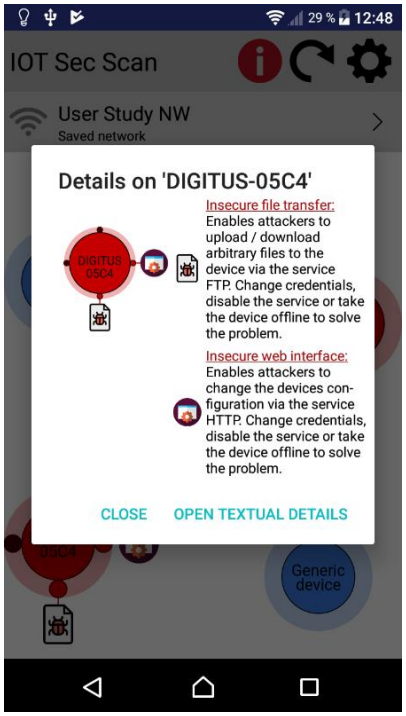


Figure 29 – Graphical details on a device with insecure FTP access and insecure web interface

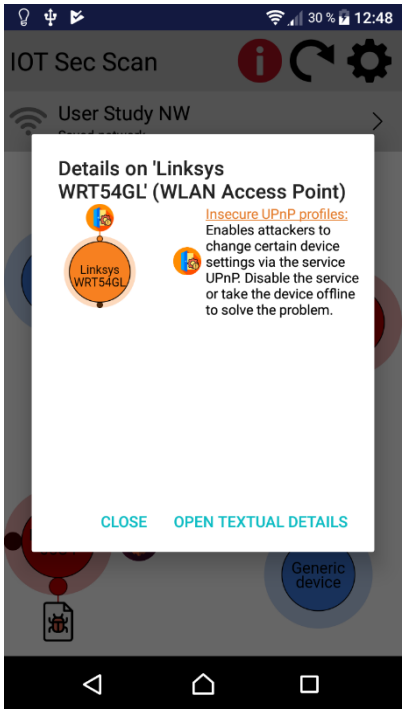


Figure 30 – Graphical details on a device with insecure UPnP profiles

Textual Device Details

This Activity provides functionality for displaying all the device-information collected during the device analysis, and security-testing phases of a network scan (see chapter 4.3). Figures 31 and 32 display a device in the Expert mode (comprehensive textual information). The app detected that several services are running on this device such as SSH, Telnet, DNS and HTTP. The Telnet service is marked in red because it was found to be vulnerable in a preceding scan. The user can find out what exactly is wrong by clicking on a service. As described in the next section, a popup then opens which provides details on the found vulnerability. In the “General Information” section of this view the user can see – in addition to other general data such as device´s IP or MAC address – that this device acts as the network´s access point. The last section of this view lists all the networks where this device showed up during scans. Figure 33 displays the same device in the Normal 1 and 2 modes (reduced information). In those modes information and functionality is hidden that is considered to be too complex for novice users. Similarly, Figures 34 and 35 show a different, potentially vulnerable device. The most obvious difference to the previous example is that this device makes use of an UPnP service which contains one or more potentially problematic profiles. For devices that use UPnP, the device details view provides a dedicated section that contains additional UPnP information. Figure 36 shows the same device in the Normal 1 and 2 modes with reduced information. The information reduction is also applied to the “UPnP Information” section.

This part of the application also enables users to scan an individual device for weaknesses (any application mode) and to perform an extended port-scan (only in the Expert mode), if the corresponding device was online during a preceding scan. The Expert mode also allows the user to display a device´s service history and to remove a device from the device-list of a saved network. Removing a device is only possible if the device does not act as the network´s access point. This is because the graphical network representation of the Normal 2 mode uses the access point as root node. A detailed comparison of the presented information and the available functionality between the Expert and Normal modes is listed in Table 9.

Application mode	Expert	Normal 1 & 2 (reduced information)
Displayed information	**If the device is the device on which the app is running**	
	Model	-
	Operating System (OS) Version	-
	OS API level	-
	Hardware	-
	Host	-
	Type	-
	Display	-
	If the device is the network´s access point	
	Indicate that this device is the network´s access point	Indicate that this device is the network´s access point
	Detected ports / problems	
	Short service names	Only for insecure services: Short service names
	Always: Full service names For insecure services: Additional problem notifications	Only for insecure services: Full service names and problem notifications
	Port numbers	-
	General information	
	IP address	-
	Host name	Host name
	Netbios name	Netbios name
	MAC address	-
	MAC vendor	MAC vendor
	UPnP information	
	Friendly name	Friendly name
	Device type	Device type
	Manufacturer	Manufacturer
	Model name	-
	Model description	-
	Services	-
	SCDP URL	-
	SNMP information	
	Description	Description
	Name	Name
	Contact	Contact
	Location	Location
	Uptime	Uptime
	Object ID	-
	Services	-
	Networks that include the device	
	SSID	SSID
	IP address / network prefix	-
	BSSID	-
Possible actions	**General**	
	Display service history	-
	Remove device from device list	-
	If the device is contained in the currently connected network	
	Scan for problems	Scan for problems
	Extended scan for open ports	-

Table 9 – Displayed information and possible actions in the Textual Device Details view

- **Example Device 1**

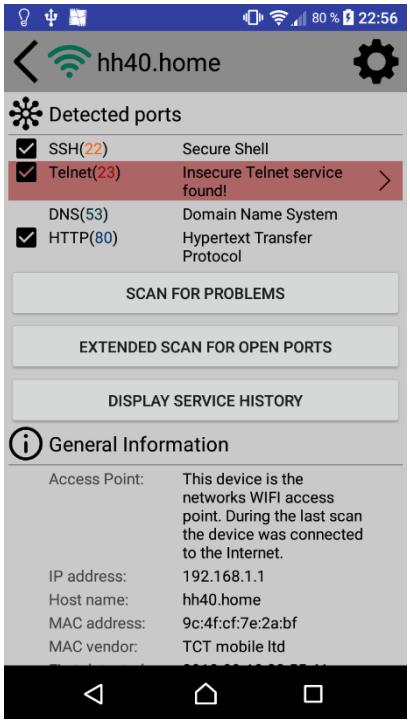


Figure 31 – Textual details of a critical device in Expert mode (part 1)

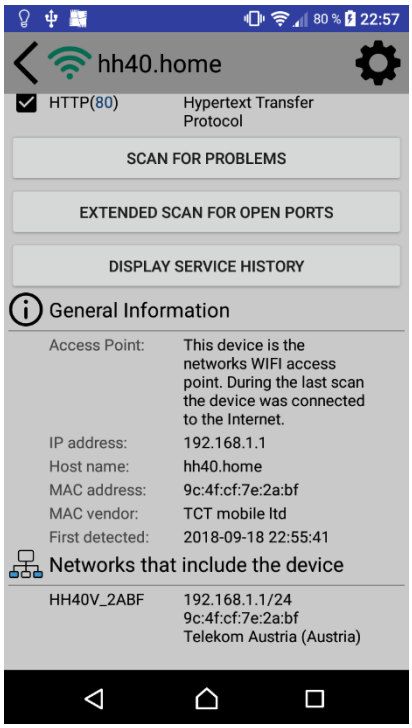


Figure 32 – Textual details of a critical device in Expert mode (part 2)

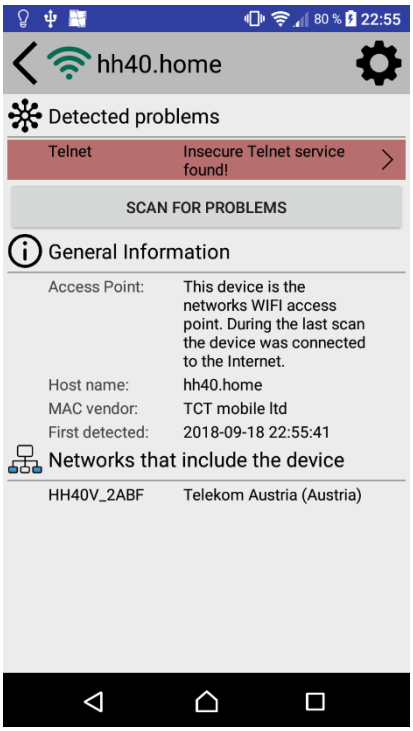


Figure 33 – Textual details of a critical device in Normal 1 & 2 modes (reduced information)

- **Example Device 2**

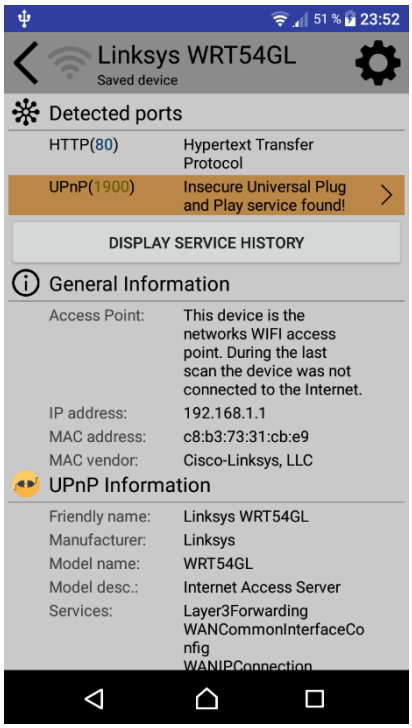


Figure 34 – Textual details of a device with problematic UPnP profiles in Expert mode (part 1)

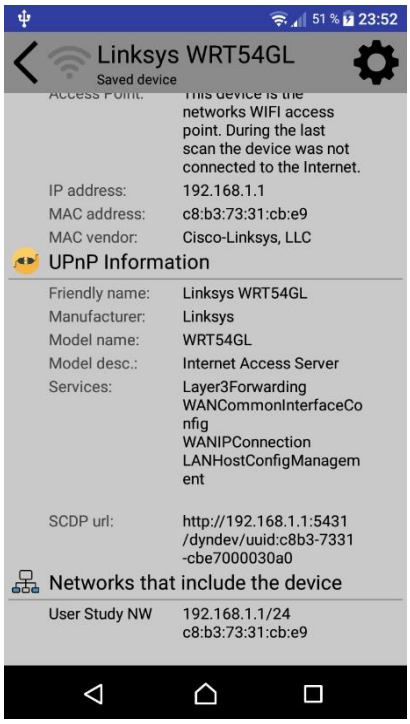


Figure 35 – Textual details of a device with problematic UPnP profiles in Expert mode (part 2)

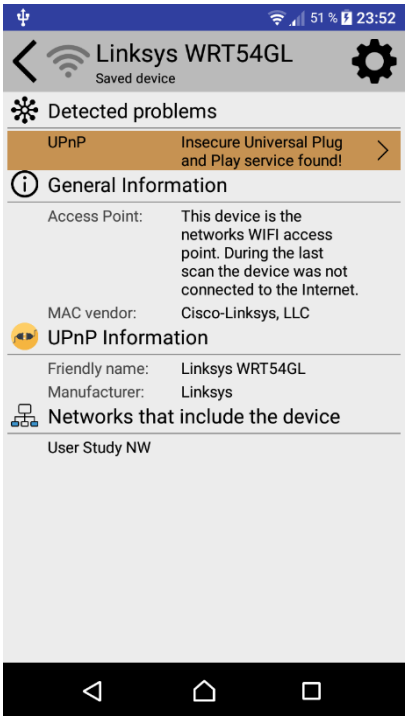


Figure 36 – Textual details of a device with problematic UPnP profiles in Normal 1 & 2 modes (reduced information)

Vulnerability Details

This Activity displays vulnerability-specific information and provides general suggestions for mitigating the found problems. Those general mitigation suggestions are to either change insecure authentication credentials (in case of an authentication vulnerability), to disable the corresponding service or to take the device offline altogether. The popup also provides a link to an online resource where the affected service is explained in more detail:

- HTTP: https://study-ccna.com/http-https/
- FTP: https://study-ccna.com/ftp-tftp/
- Telnet / SSH: https://study-ccna.com/telnet-ssh/
- UPnP: http://www.upnp-hacks.org/upnp.html

Figure 37 shows information on a vulnerable Telnet service in the Expert mode. Figure 39 displays the popup containing information on a problematic UPnP service – also in the Expert mode. Analogously Figures 38 and 40 depict the same problematic services – but with reduced information – for both the Normal 1 and Normal 2 modes. Detailed differences in terms of application modes can be found in Table 10.

Application mode	**Expert**	**Normal 1 & 2 (reduced information)**
Displayed information	**Any vulnerability type**	
	Short service name	Short service name
	Full service name	Full service name
	Port	-
	Vulnerability description	Vulnerability description
	Remediation advice	Remediation advice
	Information link	Information link
	Authentication vulnerability (Default credentials for Telnet, SSH, HTTP or FTP)	
	Found username and password	Found username and password
	Only for Telnet or SSH: Console output after a successful attack	-
	Potential UPnP vulnerabilities	
	Names of problematic UPnP profiles	Names of problematic UPnP profiles
	Detailed UPnP profile explanations	-

Table 10 – Displayed information in the vulnerability details popup

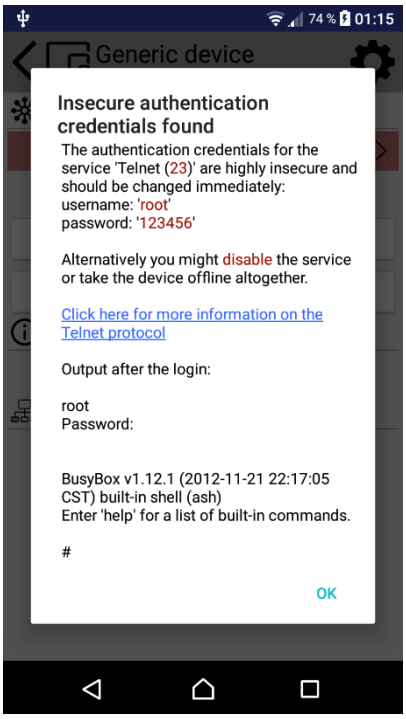


Figure 37 – Details on weak Telnet authentication in Expert mode

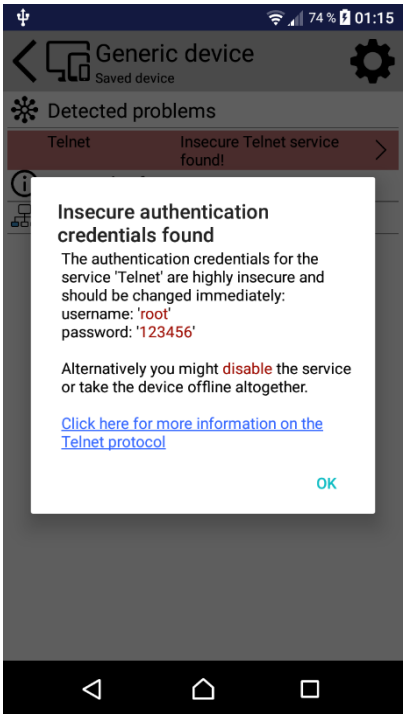


Figure 38 – Details on weak Telnet authentication in Normal 1 & 2 modes

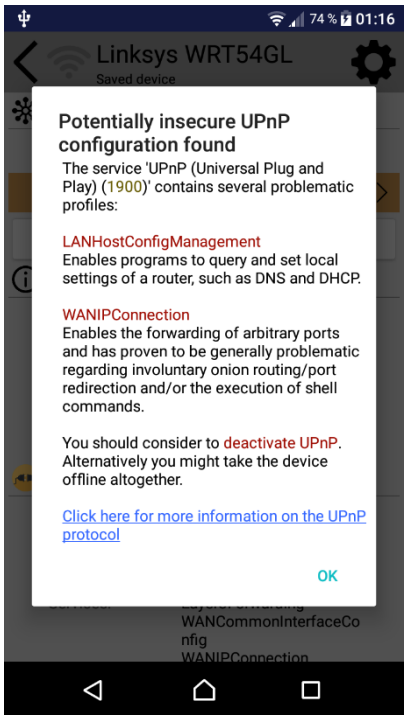


Figure 39 – Details on potentially insecure UPnP profile in Expert mode

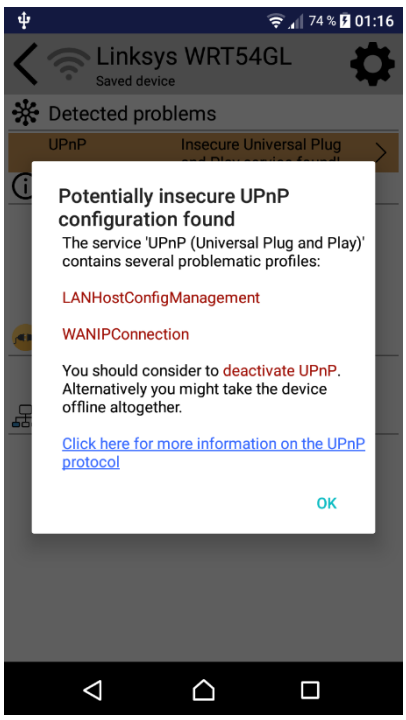


Figure 40 – Details on potentially insecure UPnP profile in Normal 1 & 2 modes

Device Service History

This Activity displays a chart for an individual device which is similar to the Normal 2-mode network-snapshot chart. However, this chart indicates the change of service states for a specific device over time. The different service states are:

- Problematic
- Open
- Closed
- Unknown (port not checked)

The visualization should make it especially obvious which services turned out to be problematic during the device´s scan history. This mode is only available via the Expert application-mode, since network services and ports are generally found to be too complicated topics for novice-users. Figure 41 displays the service history chart for a laptop.

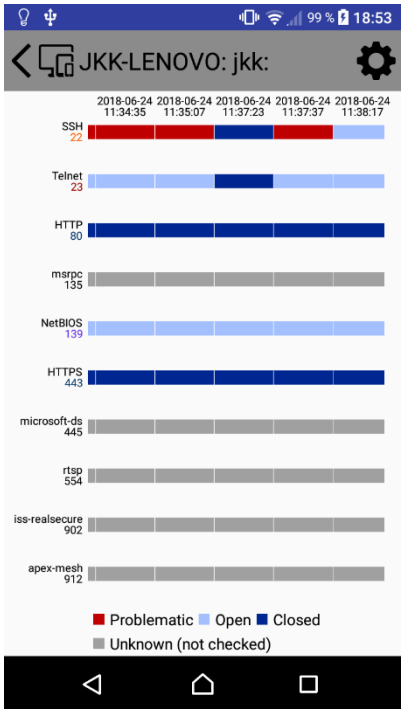


Figure 41 – Device service history chart for a laptop (Expert mode only)

4.2.6 Usage of Fragments in the UI Context

Fragments are components that can be used to enable other components to outlive the Android Activity-lifecycle. In the UI context, Fragments have proven to be especially useful for saving and retaining data, as well as the creation of dialogs.

Data-Fragments

During configuration changes, such as screen rotations, the current Activity – including any data that is associated with it – gets destroyed. During the Activities´ subsequent recreation, associated data is not automatically retained. In the app Data-Fragments are used in the UI context for saving and retaining large amounts of data. The data comes in the form of collections, containing a large amount of device-objects which might get created during scans of larger networks.

Large device-collections might also be created in the context of functionality which requires to load many saved devices from the database, such as network-scan history features. After a configuration change occurred, device-collections can be retained and used to again display devices either via lists (in the Expert and Normal 1 application-modes) or graphically (in the Normal 2 application mode). Traditionally, data can also be saved by writing it into a parcel in the Activity´s SavedInstanceState method. However, this method has one major drawback in comparison to the Fragment-solution described above. By writing too much data into a parcel, a "TransactionTooLargeException" can occur [56].

Dialog-Fragments

Dialog-Fragments are used for creating information popups with custom layouts that also might contain more complex content than just static text, such as WebViews. Dialog-Fragments are used for displaying detail information for the following areas:

- AuthenticationDialogFragment: Used for displaying insecure authentication.
- UPnPDialogFragment: Used for displaying insecure UPnP configuration.
- NetworkStatusDialogFragment: Used for displaying the overall network state after a scan.
- GraphDeviceDialogFragment: Used for displaying devices / vulnerabilities in the graphical application-mode.

4.3 Scanning Functionality

In the following sections approaches are described which are used for the detection, information gathering and security scanning of devices.

4.3.1 Scan Modes

The application´s settings let users select one of three scan modes which are based upon each other and alter the behavior of the main network scan. This feature is implemented in order to give advanced users more control over what is actually happening during a network scan. The scan modes can be briefly described as follows:

- Mode 1 – Device detection and information gathering: Devices will be detected and for each device the app will try to extract general information.
- Mode 2 – Device detection and information gathering and basic port scan: This mode – in addition to the functionality from the first mode – includes a scan on the standard ports of ten common network services.
- Mode 3 – Device detection and information gathering and basic port scan and vulnerability scan: This mode – in addition to the functionality of the first and second modes – includes vulnerability scans for the authentication of certain services via brute force.

4.3.2 Device Detection

The initial detection of devices during the scan of a network mainly takes place in three stages:

1. A Simple Service Discovery Protocol (SSDP) broadcast is performed to detect devices which have the UPnP protocol enabled.
2. Every possible IP address of the current network is checked with the ping utility.
3. Lookups in the Address Resolution Protocol (ARP) cache are performed for devices that haven´t been detected in the previous two steps. This makes sense as the ARP resolution might still have taken place, even for devices that did not respond to SSDP broadcasts or ping requests.

4.3.3 Device and Network Analysis

During the device and network analysis phase of the network scan, the application tries to extract the information listed in Table 9 for each device. This is accomplished by making use of numerous smaller helper-classes, which in turn are using libraries for fetching SNMP-, UPnP-, Netbios- and OUI-data. For each network, the application tries to gather the information listed in Table 7. For gathering Internet provider information the app makes use of the API provided by http://ip-api.com. Most of the found network- and device-data is gathered and displayed for mainly informational purposes. A major goal of the device analysis phase is to make individual devices as easily identifiable as possible. Security relevant information that can be extracted directly during this phase are the detection of potentially problematic UPnP profiles, as briefly described in Section 4.3.4.

4.3.4 Security Scanning

In the following sections elaborations are provided on the techniques used for extracting security relevant information, taking place after the device-detection and device-analysis phases.

Port Scanning

Port scanning takes place in several different application contexts and different extent in terms of how many ports are scanned. A basic port scan consists of a scan on the standard ports of ten common services and takes place in the context of a main scan on an entire network (depending on the selected scan-mode). It is also automatically executed for a vulnerability scan that is triggered on the device detail Activity of an individual device. The used services are listed in Table 11. The selection of those services is based on the service list that is used within the popular Nmap port scanning tool. Each service in this list is attributed with an “open-frequency” which indicates the rate with which the service is opened. The top ten services of this list based on their open-frequency were selected to be used in the app´s basic port scan.

Abbreviation	Full service name	Standard Port
FTP	File Transfer Protocol	21
SSH	Secure Shell	22
Telnet	Telnet	23
SMTP	Simple Mail Transfer Protocol	25
DNS	Domain Name System	53
HTTP	Hypertext Transfer Protocol	80
POP3	Post Office Protocol v3	110
HTTPS	Hypertext Transfer Protocol over SSL/TLS	443
SMB	Server Message Block	445
HTTP Alternative	Hypertext Transfer Protocol (alternative)	8080

Table 11 – Used services for the basic port scan

An extended port scan can only be triggered on the device detail Activity of individual devices and consists of a scan on the standard ports of 6323 services. The used service list is also fetched from the Nmap port scanning tool.

Vulnerability Scanning

Similar to the basic port scan, a vulnerability scan might take place in the context of a main scan on an entire network (depending on the selected scan-mode) or in the context of a scan on an individual device. The latter can be triggered in a device´s detail Activity. During a vulnerability scan the app tries to find weak authentication credentials via brute-force approaches by making use of a standard username/password list consisting of ~70 credential pairs. Most of the used credentials are directly taken from the Mirai source code [57]. However only the following services can be checked for weak authentication – which makes it necessary that one or more of those services are actually found to be running on the target device during the preceding basic port scan:

- HTTP
- FTP
- Telnet
- SSH

- **HTTP Message Digest Authentication**

First the application verifies whether HTTP Message Digest Authentication is available on the target device (HTTP-Response-Code 401). Upon successful verification the application determines whether the device is vulnerable, by observing response codes after brute-force authentication attempts (by making use of the credential-list mentioned above). The HTTP-Response-Code 200 means that the authentication attempt has been successful. The library for performing HTTP-requests is called “OK-HTTP” and is used in different parts of the application [47].

- **HTTP Authentication via Simple HTML Forms**

Some devices provide authentication via HTML forms. However, to detect whether a device is vulnerable or to successfully verify that a device is in fact even using this kind of authentication is far from trivial. One of the reasons for this is, that many modern web applications often make excessive use of JavaScript which is integrated into HTML form authentication mechanisms. This makes it very hard to programmatically find a way to successfully verify the existence or even brute-force this kind of authentication, without parsing and intelligently interpreting JavaScript code.

However, for very basic HTML form authentication mechanisms that refrain from making use of JavaScript, simple verification and brute-force functionality was implemented. To verify if there is an actual HTML form authentication mechanism in place, the app first follows any possible HTTP header-redirects and then tries to find a form element from the underlying HTML source code. If the search for a form element is unsuccessful, the app tries to anticipate possible JavaScript redirects by extracting and visiting the URLs found in JavaScript redirect-calls listed beneath:

- window.location
- top.location
- document.location
- parent.location

After performing manual source-code checks, the listed redirect methods are found to be common with devices that use HTML authentication. After a corresponding URL is found and followed, the underlying HTML source code is again searched for possible form elements. Once a form element is found, the following information is extracted:

- The value of the form´s “method” parameter (mostly GET or POST)
- The value of the form´s “action” parameter
- The name of a possible “username” input-child-element (possible candidates are input-elements that contain the string “user” in the name-attribute)
- The name of a possible “password” input-child-element (possible candidates are input-elements that contain the string “pass” in the name-attribute)
- The names and values of any other input-child-elements
- Any cookies

As an example, Figure 42 shows the HTML login form of a Tenda N3 Wireless router. The page even indicates that the default password for accessing the router´s configuration page is “admin”. For this login-form, all information which is necessary for being able to programmatically send login HTTP requests, can be gathered from the page´s HTML source code which is shown in Listing 1. The following important parts are marked in red in the Listing and are extracted by the app:

- The value of the form´s “method” parameter is “GET”.
- The value of the form´s “action” parameter is “/LoginCheck”.
- A possible “username” input-element is identified, including the default value “admin”.
- A possible “password” input-element is identified.
- An additional input-element with the name “checkEn” and the value “0” is identified.

Figure 42 – Login form for a Tenda N3 Wireless router

HTML source code

```
<form name="Login" method="post" action="/LoginCheck" _lpchecked="1">
       <input type="hidden" name="Username" value="admin">

       <input type="hidden" name="checkEn" value="0">

       <h1 class="login-title">Login</h1>

       <div class="container">

              <p class="login-massage"><span>Default:</span> admin</p>

              <div class="control-group">

                     <div class="control-label">Password:</div>

                     <div class="controls">

                     <input type="password" name="Password" maxlength="12" class="text
                     input-medium" onkeydown="enterDown(document.Login,event);"
                     style="...">

                     </div>

              </div>

              ...

       </div>

</form>
```

Listing 1 – HTML code of the login form for a Tenda N3 Wireless router

Via the extracted information mentioned above, a false-login-request containing bogus login-information is sent, in order to receive the HTML output of an obviously invalid authentication attempt. During the actual brute-force attack, requests containing possibly valid authentication credentials are sent. After each request the resulting HTML output is compared line-by-line to the HTML output of the false-login-request. If at any point the HTML output of an authentication request differs from the HTML output of the false-login-request, the attack might have been successful.

Yet again it has to be noted that the entire approach is rather unreliable and might not work in cases where an application´s authentication mechanism makes major use of JavaScript. As mentioned above it is very hard to programmatically find a way to successfully study the authentication mechanism (e.g. through a brute force-attempt), without parsing and intelligently interpreting JavaScript code. Also dynamic content like the display of the current time makes the direct comparison of HTML responses error-prone and might lead to false-positives.

- **FTP Authentication**

The code created for the FTP authentication brute-force makes use of the apache.commons.net FTPClient [50] functionality and is very straight-forward. If an FTP connection attempt in combination with a certain credential-pair is working, then the brute-force attack was successful.

- **SSH Authentication**

Similarly to the FTP authentication, the functionality created for the SSH authentication brute-force is also pretty straight forward. The code uses the “Java Secure Channel” [49] library and attempts to establish working SSH sessions in combination with possibly valid credential pairs.

- **Telnet Authentication**

The Telnet brute-force functionality makes use of the “Sadun Telnet client library” [48] and is slightly more complex than the previous two brute-force mechanisms. With Telnet it generally is not possible to reliably detect successful or unsuccessful login attempts. This is because the actual authentication functionality is not part of Telnet but part of the underlying server application. Devices may differ in their underlying operating systems and command line interfaces. This means that login procedures do differ as well. Even with the help of the used Telnet client library it is possible to merely read-from or write-to the stream of an established network socket and to determine by the server´s response whether a login succeeded or not. However due to the aforementioned differences this is not so obvious to detect. In fact the author of the infamous “Mirai” IoT malware was facing the same issue. However, Mirai´s source code was published in late 2016, which made it possible to take a look at the malware author´s approach. We see that the author detects successful login-attempts by certain characters that are likely to be part of a shell prompt, such as ‘>’, ‘#’ or ‘$’. The corresponding code is located in the lines 768 to 844 of the malware´s scanner component [57]. Therefore a similar approach is used in the application, which has proven to work well for most devices that are accessible via Telnet.

- **Problematic UPnP Profiles**

The following UPnP profiles are directly detected during the device-analysis phase and have proven to introduce potentially problematic functionality [58]:

- LANHostConfigManagement: This profile enables programs to query and set local settings of a router, such as DNS and Dynamic Host Configuration Protocol (DHCP) changes.
- WANIPConnection / WANPPPConnection: Those profiles enable the forwarding of arbitrary ports and have been proven to be generally problematic regarding involuntary onion routing/port redirection and/or the execution of shell commands.

4.3.5 Usage of Task-Fragments

In the application, Task-Fragments are essential components that are used for all longer-running background tasks – a topic which is therefore especially important for the app´s scanning functionality. Task-Fragments are needed for being able to re-establish the reference from a running task to an Activity after configuration changes - such as screen rotations. The main idea behind TaskFragments is to encapsulate a potentially longer running AsyncTask into a Fragment that might be reliably restored after configuration changes. The employed approach has proven to be practical and robust from the very start of the project [59]. By not using this method, the reference from a running task to a newly created Activity would be lost, ultimately leading to undesired behavior. The first problem is that the UI cannot be accessed/updated anymore from within the running task. Secondly, Memory leaks occur as the task still holds a reference to the old Activity, preventing the garbage collector from doing its work.

The following list briefly describes the functionality implemented within the most important TaskFragment classes in combination with their corresponding inner AsyncTask classes and numerous helper classes:

- Performing the application´s main scan on an entire network, including functionality for device-detection, information gathering, vulnerability-scans and basic port scans.
- Performing extended port scans and vulnerability scans for individual devices.
- Initialization of the SQLite database with MAC-vendor and network-service information during the initial startup of the application.
- Asynchronous loading of device snapshots, network snapshots and saved networks as fast as possible.

4.3.6 Typical Scan Sequence

Listing 2 contains a very simplified algorithm in pseudo-code (Java-based) which includes all major steps that take place during a scan. The algorithm is independent from any UI functionality / interaction and considers all phases described in the previous sections.

Pseudo-code (Java-based)

```
public void scanCurrentNetwork(){
    Network currentNetwork = getCurrentNetwork();
    gatherNetworkInfo(currentNetwork);
    //network and snapshot database actions
    Long networkId = DBHelper.createOrUpdateNetwork(currentNetwork);
    Long networkSnapshotId = DBHelper.createNetworkSnapshot(currentNetwork);

    new ScanTaskFragment<>(){        //execution within a Fragment
        new AsyncTask<>() {          //execute asynchronously
            List<Device> foundDevices = new ArrayList<>();
            //start device detection
            addDevicesFromUPnPBroadcast(foundDevices,currentNetwork));
            addDevicesFromPing(foundDevices,currentNetwork);
            addDevicesFromARP(foundDevices);

            for(Device device : foundDevices){
                //start device analysis
                addUPnPInfo(device);
                addHostName(device);
                addNetbiosInfo(device);
                addSNMPInfo(device);

                //start security scanning
                performBasicPortscan(device);
                if(device.isUsingHTTPMsgDigestAuth()){
                    attackHTTPMsgDigestAuth(device);
                } else if(device.isUsingHTMLAuth()){
                    attackHTMLAuth(device);
                }
                if(device.isUsingTelnet()){
                    attackTelnet(device);
                }
                if(device.isUsingSSH()){
                    attackSSH(device);
                }
                if(device.isUsingFTP()){
                    attackFTP(device);
                }
                //device and snapshot database actions
                DBHelper.createOrUpdateDevice(device,networkId);
                DBHelper.createDeviceSnapshot(device,networkSnapshotId);
            }
        }.execute();
    }.commit();
}
```

Listing 2 – Scan sequence algorithm in (Java-based) Pseudo-Code

5. Evaluation

The prototype is evaluated along two important criteria. Firstly, the functionality is tested on several Android test devices and different IoT devices to demonstrate that the scanner application works as intended (Section 5.1). Secondly, tests were conducted together with users for being able to objectively assess the app´s usability when used in different application modes. The metrics for the main user-study are the completion times of several tasks that users had to complete with the help of the app and the quantitative ratings of several user-experience related aspects (Section 5.2).

5.1 Functional Tests

Functional tests took place in two stages. Firstly tests were performed on a range of several Android devices to ensure that the app is actually running properly on devices with different Android versions, and different hardware specifications (screen-size, CPU type, RAM). Secondly the app was tested regarding the detection of devices and inherent vulnerabilities with a variety of different IoT devices. This is to ensure that the functionality described in Section 4.3 works as intended. Additionally, the outcome of those tests results in a list that contains the security state of a range of different IoT devices in terms of authentication security.

5.1.1 Android Test Devices

At no point during development an emulator has been used. New functionality was always tested by making use of real devices (mainly phones) which run on the supported Android versions. This is because the app contains numerous networking features – many of which are hard or impossible to test on emulators. Table 12 lists the used Android devices on which the app was successfully tested.

Device name	Android version	CPU type	RAM
Samsung Galaxy S8	Android 8	4x2.3GHz and 4x1.7GHz Octa-Core	4GB
Sony Xperia XA	Android 7.0	4x1.0GHz and 4x2.0GHz Octa-Core	2GB
Huawei P9 Lite	Android 7.0	4x1.7GHz and 4x2.0GHz Octa-Core	2GB
Samsung Galaxy Tab A	Android 7.0	8x1.6GHz Octa-Core	2GB
LG G4 Stylus	Android 6.0	1.2GHz Quad-Core	1GB
Lenovo Tab 3 Essential	Android 5.1	1,3GHz Quad-Core	1GB

Table 12 – Android devices used for testing

5.1.2 Smart Home Devices

For testing the app´s scanning and authentication-brute-forcing functionality, the devices in Table 13 (a) and (b) were utilized after resetting them to their factory defaults. The devices mostly consist of Small Office, Home Office (SOHO) routers that were tested individually or in a network, together with other devices. For routers this was possible by turning off DHCP and by configuring them to use static IP addresses.

Furthermore, the Android devices that have been used for running the app, were also included in such networks. In addition to that, tests for detection and port-scanning functionality were regularly made via publicly accessible Wi-Fi networks.

For testing the brute-forcing functionality for FTP authentication, the “Digitus Mini Network Attached Storage (NAS) server”, the “Zyxel Prestige 600” and the “Zubo Helicute H821HW” devices (the first three in Table 13 (a)) were primarily used. The app was able to successfully log in to the FTP service of the NAS and the “Zubo Helicute” devices with the credentials “anonymous” : “”. The FTP service of the Zyxel router could be entered with the credentials “test” : ”1234”.

Furthermore it is noticeable that only two of the devices support authentication via SSH. Those are the “Thomson Speed Touch 5 tg 585 v7” router and the “3Com Switch 4500” (fourth and fifth in Table 13 (a)). The router could be accessed with the credentials “Administrator” : “” and the switch can be accessed with the credentials “admin” : “”. Additionally, the SSH and Telnet authentication brute-force functionality was also successfully tested via the “FreeSSHd” tool on a Windows 7 laptop. This tool makes it possible to easily set up SSH or Telnet servers that grant console access after successful authentication [60]. For the sake of brevity, the availability of FTP or SSH and the afore-mentioned tests on those services are not included in the Tables 13 (a) and (b).

Device name	**Type**	**HTML form or HTTP auth.**	**Found HTML/HTTP std. pass**	**Telnet auth.**	**Found Telnet std. pass**
Digitus Mini NAS Server	NAS-Server	Http auth	Yes, with “admin” : “admin”	No	-
Zyxel Prestige 600	Router	Http auth	Yes, with "admin" : "1234"	Yes	No (can´t connect multiple times)
Zubo Helicute H821HW	Drone	No	-	No	-
Thomson Speed Touch tg 585 v7	Router	Http auth	Yes, with “Administrator” : “”	Yes	Yes, with “Administrator” : “”
3Com Switch 4500	Managed switch	No	-	Yes	Yes, with “admin” : “”
Ibox Wi-Fi repeater WR01	Wi-Fi repeater	Html auth	No (extensive use of JS)	Yes	Yes, with “admin” : “1234567890”
Linksys Wireless-G Broadband Router WRT54GL v1.1	Router	Http auth	Yes, with “admin” : “admin”	Yes	Yes, with “admin”
Maginon IP-Camera IPC-100AC	IP Camera	Html auth	No (extensive use of JS)	Yes	Yes with “root” : “123456”
Cisco E3000	Router	Http auth	Yes, with “admin” : ”admin”	No	-
Zyxel NBG-417N	Router	Html auth	Yes, with “” : "1234"	No	-
Linksys Wrt54G2 V1	Router	Http auth	Yes, with “admin” : “admin”	No	-
Linksys Wrt54GC	Router	Http auth	Yes, with “admin” : “admin”	No	-

Table 13 (a) – Test devices used

Device name	Type	HTML form or HTTP auth.	Found HTML/HTTP std. pass	Telnet auth.	Found Telnet std. pass
TP-Link 54M	Router	Http auth	Yes, with “admin” : “admin”	No	-
Netgear WGR614 v6	Router	Http auth	Yes, with “admin”:”password”	Yes	No (connecting to the port was not possible)
Siemens Gigaset SE505	Router	Html auth	Yes, with “”:””	No	-
Netgear WNR834B	Router	Http auth	Yes, with “admin”: “password”	No	-
Thomson Speed Touch 510i	Router	No	-	Yes	Yes, with “” : “”
Edimax Broadband Router	Router	Http auth	Yes, with "admin" : "1234"	No	-
Linksys E1200	Router	Http auth	No (multiple steps in HTML page necessary)	No	-
Speedport W 500V	Router	Http auth	Yes, with "root" : "0000"	Yes	Yes, with "root" : "0000"
Fujitsu Siemens AP-600RP	Router	Http auth	Yes, with "" : "connect"	No	-
Belkin N300	Router	Html auth	No (extensive use of JS)	No	-
Tenda N3 Wireless	Router	Html auth	Yes, with "admin" : "admin"	No	-
Lenovo ThinkPad T540p - Win 7	Laptop	No	-	No	-
Lenovo ThinkPad T61 - Kali Linux	Laptop	No	-	No	-
Asus Workstation - Win 7	Desktop	No	-	No	-
Sony PS4	Gaming Console	No	-	No	-
Medion MAX! Cube LAN Gateway	Smart Home device	No	-	No	-
Panasonic TXL42ETW60	TV	No	-	No	-
Samsung Galaxy S8	Mobile Phone	No	-	No	-
Sony Xperia XA	Mobile Phone	No	-	No	-
Huawei P9 Lite	Mobile Phone	No	-	No	-
Samsung Galaxy Tab A	Tablet	No	-	No	-
LG G4 Stylus	Mobile Phone	No	-	No	-
Lenovo Tab 3 Essential	Tablet	No	-	No	-

Table 13 (b) – Test devices used

Vulnerabilities Detected in Smart Home Devices

The results of the tests can be summarized as follows:

- Routers: Most of the test devices are routers and most of them are either susceptible to Telnet or HTTP authentication vulnerabilities. It also has to be noted that only one of the routers supports SSH, which should be the preferred way of accessing a device´s shell. This is because in contrast to Telnet, SSH provides encrypted communication.
- NAS-Server: The NAS-Server that was used for testing allows unauthenticated FTP access and uses standard credentials for HTTP authentication.
- Wi-Fi Drone: The drone that was covered during testing allows unauthenticated FTP access to its internal system, which could allow a remote attacker to manipulate the drone´s behavior during operation. This can be considered a security vulnerability that could even be used to physically harm people.
- Managed Switch: The managed switch that was tested uses Telnet and SSH services by default – both of which were using standard authentication credentials.
- Wi-Fi Repeater: The tested Wi-Fi repeater also allows Telnet access via standard credentials.
- IP Camera: The IP camera that was tested contains a Telnet backdoor. In contrast to networking-devices such as routers, it can be considered rather unrealistic that Telnet access to a camera provides any benefit for end-consumers. This service was most likely used by developers for debugging purposes and forgotten to be removed before the product was shipped.
- Smart Home Device: The Medion MAX! Cube LAN Gateway device acts as a central heating-regulation system. None of the vulnerabilities that were covered in this project could be found with this device.
- Laptops and Desktops: None of the covered vulnerabilities could be found when scanning Windows 7 laptops and desktops with the app.
- Gaming Console: The PS4 gaming console that was tested with the app did not display any of the vulnerabilities that are covered in the project.
- TV: The Panasonic TV that was tested also did not contain any of the covered vulnerabilities.
- Mobile Phones and Tablets: Lastly, the phones and tablets that were scanned during testing also did not show any of the covered vulnerabilities.

To summarize, a considerable number of the tested devices is displaying authentication vulnerabilities and most of those devices could be successfully identified as vulnerable by the app described in this thesis.

5.2 Usability Tests

Two user studies and their evaluations are presented in order to objectively assess the app´s usability. The preliminary user study consisted of semi-formal interviews where the participants were guided through the app. Based on the participants´ comments regarding the usability of UI components and the understandability of presented information, a “comprehensibility-score” was assigned to components or pieces of information. The results of this first test were primarily used to increase the usability and overall quality of the app. This was done by hiding information, renaming description texts and fixing bugs.

The first objective of the main user-study – which was conducted with a questionnaire – was to test how well users manage to perform basic tasks with the app when using it in a certain application mode. This was achieved by measuring the time that participants took to complete three different user tasks with the help of the app (the faster the better). The second objective of the main user study was to assess how well participants perceive the user experience with the app after using it in a certain application mode. This was achieved with a short evaluation where participants could numerically rate the app in terms of several user experience related topics. This approach made it possible to compare the participants´ results with special regard to the three application modes.

5.2.1 Preliminary User Study

The main objective of the preliminary user study was to gather first impressions on how people would use the app, how well the presented information was understood and to detect inconsistencies and bugs. The preliminary user study was conducted via individual semi-formal interviews with four technically inexperienced participants, consisting of one female (age 21) and three males (with the ages 24, 24 and 28). They were questioned about their opinion on all of the UI components and the understandability of all information described in Section 4.2, which resulted in interviews that approximately took 30 to 40 minutes. Based on the participant´s answers the interviewer assigned a “comprehensibility-score” to individual UI components or pieces of information. The score could either be 1 (difficult to comprehend), 2 (somewhat comprehensible) or 3 (fully comprehensible). To ensure that all UI areas were encountered, the participants were guided through the app while also having the opportunity to interact with it at will. The app was presented in the Expert mode to get an impression on what kind of information could be too complex for technically inexperienced users.
The received feedback was directly utilized, mainly for changing the app in the following ways:

- Hide information which is difficult to comprehend in the Normal 1 and 2 modes
- Rename description texts based on modes
- Fix bugs

5.2.2 Main User Study

In the following sections, elaborations are provided on details of the main user study. Those include the goal of the study, the used metrics, the used questionnaire, information on user tasks, the procedure which was employed when conducting the study and the study´s results.

Goal of the Study

One major goal of the study is to objectively assess how efficiently users are able to complete common tasks with the app in a certain application mode. Another goal of the study is to determine how participants perceive the app´s user experience after using the app in a certain application mode. This makes it possible to compare the three different application modes in terms of the users´ completion times for different tasks and in terms of the user experience ratings.

Metrics used for the Evaluation

The main user study is conducted with a questionnaire that for one part consists of three different tasks that users have to complete with the help of the app. The time that it takes for users to complete those tasks is used as metric to determine how efficient they are in using the app (the faster the better). The second part of the user study targets subjective ratings of the users´ experience. Users can rate the app along different categories with an integer score ranging from -2 to +2.

Questionnaire

The main user study was conducted via a simple questionnaire which consists of the following parts:

1. General Information: This section provides information on how much time the survey should approximately take and gathers information on the participant´s gender, technical experience and age.
2. User tasks: This section contains three different, increasingly difficult tasks that the participant has to solve by making use of the app in one pre-defined application mode. The user tasks are based on a pre-defined saved network. Before the tasks two notes state that participants should pay special attention to underlined parts and to return to the “Saved Networks” view each time a task was completed. By introducing a fixed starting-point the attempt was made to ensure that the user tasks are solved independently from each other.
3. User experience evaluation: The questionnaire is concluded by a section that lets participants rate the app in terms of several user experience aspects, by assigning values from -2 to +2. The used evaluation schema and the various user experience aspects are based on Maria Rauschenberger et al. [61].

The different aspects that are used in the user experience evaluation and their explanations are:

- Attractiveness: The app is appealing – I like it (overall rating).
- Efficiency / Understandability: The app is well structured and easy to understand.
- Dependability: The behavior of the app is comprehensible (the app does not confuse).
- Novelty: The app comprises novel aspects and is original.
- Stimulation: The app is interesting, I will likely use it in future.

Figure 52 (in the Appendix) shows the questionnaire that was used for the study.

User Tasks

All user tasks were designed to ensure that participants interact with parts of the app that contain interesting information, such as details on vulnerabilities and historical data. The tasks were also chosen to introduce parts of the UI that differ between application modes. This makes it possible to compare the modes in terms of completion times and correctness of answers for the user tasks and also regarding user experience ratings.

- **Task 1**

This task is about looking up a certain device, describing inherent vulnerabilities and providing solutions for dealing with them. Even though there is no major difference for the vulnerability descriptions between application modes, this task is still very important. It is emphasizing one of the app´s main application areas which is the presentation of vulnerabilities and appropriate remediation suggestions. It is of vital importance that the app presents this kind of information in a way so users can understand it, thus the necessity for evaluating their ability to do so.

- **Task 2**

The second task requires users to find the device which acts as the network´s Wi-Fi access point. In the device list (Expert and Normal 1 modes) the corresponding device is marked with a preceding Wi-Fi symbol. If in doubt the user can always open a device´s detail view – which provides a textual indication if the device is the network´s access point. With the graphical network representation (Normal 2 mode) this information might be easier recognizable, due to the access point´s distinct visual properties. The device node is larger, positioned in the middle of the screen and connects all devices with each other. This task aims at revealing possible differences in how visual network representations are perceived compared to networks illustrated as lists of devices.

- **Task 3**

The last task expects users to identify the date and time when the scan took place that first revealed a problem with a certain device. Users are instructed to do so by inspecting the network´s scan-history chart. This task has the primary objective to identify possible differences between the three different history charts in terms of completion times and correctness of the answers. Here the participants that used the app in the Normal 2 application mode are generally expected to have an advantage over participants that use the app in other modes. This is due to the fact that the chart in the Normal 2 mode does not require any user interaction to find the desired information. The answer can be found by simply looking up the time and date (x-axis) when the corresponding device (y-axis) was first marked with a red rectangle (“problematic”). To find the same information with charts of other modes, user-interaction becomes necessary. By opening device-groups, looking at device-lists and comparing them to those of other device-groups, users might identify when a specific device first turned out to be problematic.

Procedure

The user study was conducted with a total of 30 participants consisting of students of JKU, aged between 18 and 46. They were mostly invited to the study by short presentations that were held at the beginning of various computer science lectures. One major goal of the main user study was to compare the app´s three different application modes. This was accomplished in the following way: For each of the three modes ten persons would fill out the questionnaire by making use of the app in the respective mode. For providing a pre-defined saved network which is required for the user tasks, functionality has been implemented which makes it possible to easily clear the app´s database and re-initialize it with an appropriate test-network. All records are inserted with hard-coded information which guarantees that every participant is working with the same data-set. The saved network consists of multiple devices and contains several history entries. The described database functionality can be triggered by an additional button on the home screen which can be hidden via the app´s settings menu.

At the beginning of an evaluation session, each participant receives a smartphone on which the app was installed beforehand. Furthermore the app was set to the appropriate application mode and the database was initialized with the functionality described above. At first the participants are asked to fill out the “General Information” section of the questionnaire. Then the two notes in the “User Task” section are mentioned with special emphasis on the necessity to start each task from the “Saved Networks” view, as described in the previous sections. After that the participants are introduced to the time-measurement approach which is required in order to determine how long it took for the user to complete individual tasks. A task has been finished when the participant has solved the task and returned to the “Saved Networks” view. For each task n (n=1,..3), the CompletionTimeStamp(n) is noted as the point in time when task n has been finished, where CompletionTimeStamp(0) = 0. In a post processing step the time to solve a task is calculated as:

TimeToSolve(n) = CompletionTimeStamp(n) – CompletionTimeStamp(n-1);

Upon completion of the tasks, the participants may fill out the “user experience evaluation” part. Even though the questionnaire mentions an approximate duration of 15 minutes for the entire procedure, no fixed time limit was set for any parts of the study.

Collected Data

Table 18 (see Appendix) displays the raw data that was gathered from the user study. It lists the used application mode, the participant´s gender, age and whether the participant would consider him/herself as technically experienced. The data further contain the times that were required to complete each of the three user tasks and information on whether the given answers were correct, partially correct or incorrect (this was assigned by the author afterwards). The values that were assigned during the third part of the questionnaire which is the “user experience evaluation” are also part of the raw data.

Results

Tables 15, 16 and 17 summarize the evaluation results. From Table 18 it becomes apparent, that some participants solved some tasks incorrectly or only partially correct. However, the solve-times for those (incorrectly or partially solved) tasks were still included in the overall evaluation without any dedicated labelling or changes. Even though the number of incorrectly or partially solved tasks is rather small, it is still important to note that those datasets exist and that they might introduce some marginal errors. The numbers of correctly, partially correct and incorrectly solved tasks by application mode are listed in Table 14.

It also is important to note that the application modes, on which the user study is largely based, were primarily designed for two different target audiences. Both Normal modes were designed for technically inexperienced audiences whereas the Expert mode is targeted towards users with more technical knowledge. Since the participants of the study were invited in the course of various computer science lectures it is safe to assume that they rather fall into the latter category. This is further underlined by the results of the questionnaire where all participants stated that they would consider themselves technically experienced. The lack of technically inexperienced participants is not considered in the evaluation of the results. The circumstance that there are no participants of such an audience is best explained by fewer opportunities to introduce the user study outside of technical lectures.

App Mode	Solved	Task 1	Task 2	Task 3
Expert	Correctly	8	10	9
	Partially correct	2	0	0
	Incorrectly	0	0	1
Normal 1	Correctly	9	10	10
	Partially correct	0	0	0
	Incorrectly	1	0	0
Normal 2	Correctly	8	10	9
	Partially correct	2	0	0
	Incorrectly	0	0	1
Total	Correctly	25	30	28
	Partially correct	4	0	0
	Incorrectly	1	0	2

Table 14 – Number of correctly, partially correct and incorrectly solved tasks by application mode

Table 15 contains the calculation of the average, median and standard deviation of the participants´ ages, completion times for the three user tasks and the user experience ratings. The numbers are grouped by the application mode that the participants were using. For each task colors are indicating the ranking of the application modes (green: best, yellow: middle, red: worst) for the average and the median, in terms of completion times (the faster the better). The same coloring scheme is also used for the average and median calculations of the user experience ratings (the higher the better).

When taking a look at the average and median values of the second user task one can conclude the following: In contrast to prior assumptions participants that used the app´s Normal 2 mode (which uses a graphical network visualization) were not faster in solving the task (identifying the network´s access point). However, for the third task the average and median values show that the persons using the Normal 2 mode´s chart (the matrix containing devices and scan times) were slightly faster than those who made use of other charts (the grouped bar chart and the line chart).

The user experience ratings show that for the most part the app was positively perceived. It also becomes apparent that the app´s Normal 2 mode was generally favored over the other application modes. However, for the dependability and stimulation aspects the Normal 2 mode was on average ranked slightly lower than the Expert mode, which takes the second place regarding the overall rating. The least favored mode was the Normal 2 mode which was ranked lowest for most aspects. The reason for this could be that the participants were mostly technically experienced and therefore least impressed of the Normal 2 mode – which reduces technical information.

	App Mode	Age	T1 time	T2 time	T3 time	Attractive-ness	Efficiency / Understand ability	Depend-ability	Novelty	Stimulation
Average	Expert	23	03:32	01:28	02:39	1,20	0,90	1,70	1,50	1,60
	Normal 1	24	03:56	01:15	01:59	1,10	1,40	1,30	1,40	0,80
	Normal 2	22	03:50	01:20	01:56	1,40	1,80	1,50	1,60	1,40
	Total	23	03:46	01:21	02:11	1,23	1,37	1,50	1,50	1,27
Median	Expert	21	03:29	01:19	02:22	1,00	1,00	2,00	1,50	2,00
	Normal 1	21	03:56	01:10	01:55	1,00	1,00	1,00	2,00	1,00
	Normal 2	22	03:18	01:17	01:51	1,00	2,00	2,00	2,00	1,50
	Total	21	03:24	01:14	01:55	1,00	1,00	2,00	2,00	1,00
Std. Deviation	Expert	5	01:19	00:37	01:12	0,87	0,70	0,46	0,50	0,49
	Normal 1	8	01:39	00:18	00:33	0,70	0,49	0,64	0,80	0,87
	Normal 2	2	01:01	00:20	00:37	0,49	0,40	0,67	0,49	0,66
	Total	6	01:22	00:27	00:54	0,72	0,66	0,62	0,62	0,77

Table 15 – Data evaluation: Average, Median and Standard Deviation. Colors are indicating the ranking of the application modes (green: best, yellow: middle, red: worst).

Table 16 contains the calculation of the standard error and the coefficient of variation (CV) for each task, per application mode. The latter is expressing the ratio between the standard deviation and the mean. A lesser value means less variance within the sample. A mode which shows a small CV means that the participants solved the task with a similar completion time.

When considering the average and median together with the CV, the following conclusions can be drawn:

1. The second and third tasks were generally completed faster by the participants that used the Normal 1 and 2 modes.
2. Participants that used the app in the Normal 1 and 2 modes were able to solve the second and third user task with similar completion times as most other participants who used one of those modes for those tasks.

	App Mode	T1	T2	T3
Coefficient of Variation (CV) (Average / Standard Deviation)	Expert	37,49%	42,49%	45,51%
	Normal 1	42,22%	24,92%	28,09%
	Normal 2	26,67%	26,08%	32,16%
	Total	36,43%	33,80%	41,41%
Standard Error (Standard Deviation / sqrt(N))	Expert	1,75%	0,82%	1,59%
	Normal 1	2,19%	0,42%	0,73%
	Normal 2	1,35%	0,46%	0,82%
	Total	1,05%	0,35%	0,69%

Table 16 – Data evaluation: Coefficient of Variation (CV) and Standard Error. N in the Standard Error is the number of participants (10 for each mode and 30 in total). Colors are indicating the ranking of the application modes (green: best, yellow: middle, red: worst).

Table 17 summarizes the quartile points of the evaluation of each task. The quartile points consist of the first quartile, the median value and the third quartile. The regions between the first and third quartiles are most likely to contain prospective data samples. The data of Table 17 are being visualized with three box-plot charts – one for each user task (Figures 43, 44 and 45). Each figure contains four box-plots (the three app modes and their aggregation) that describe the minimum, maximum and the calculated quartile points.

Task	App mode	Minimum value	First quartile	Median value	Third quartile	Maximum value
1	Expert	01:44	02:27	03:29	04:49	05:28
	Normal 1	01:33	02:32	03:56	05:23	06:41
	Normal 2	02:31	03:09	03:18	04:32	05:52
	Total	01:33	02:49	03:24	04:55	06:41
2	Expert	00:50	00:56	01:19	01:42	02:46
	Normal 1	00:55	01:02	01:10	01:25	01:58
	Normal 2	00:50	01:08	01:17	01:31	02:02
	Total	00:50	01:00	01:14	01:32	02:46
3	Expert	01:00	01:45	02:22	03:05	05:00
	Normal 1	01:13	01:29	01:55	02:26	02:55
	Normal 2	01:14	01:28	01:51	02:01	03:18
	Total	01:00	01:34	01:55	02:39	05:00

Table 17 – Completion time evaluation: Calculation of quartiles (data for box-plots).

The box plots visualize that the user tasks that were solved with the Normal 1 and 2 application modes were generally solved quicker. They also show that the first and third quartile points are generally closer to the median, for the Normal modes, indicating that most of the participants solved the tasks in a similar time. Finally, as visualized by the box plots, no mode has turned out to be clearly superior with respect to solving the user tasks. One of the reasons may be that the selected tasks are too simple to benefit from more comprehensible network and timeline representations.

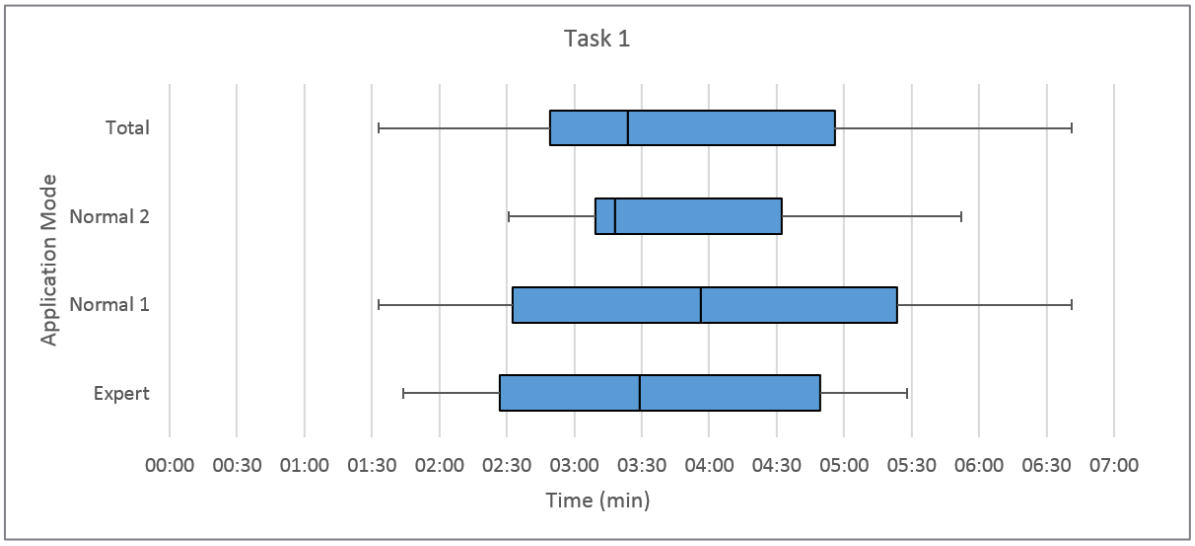


Figure 43 – Box-plot chart for the first user task.

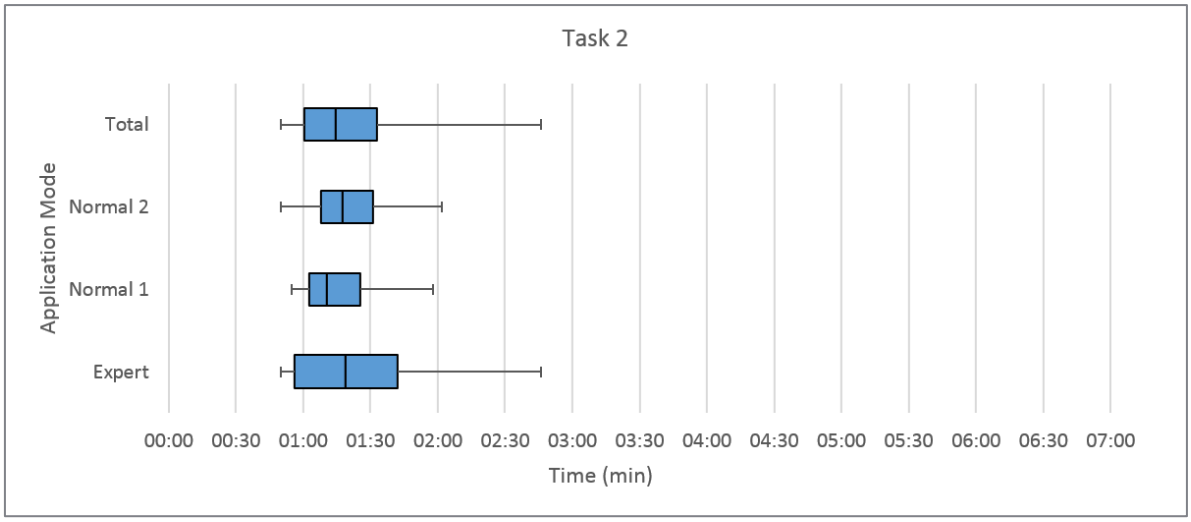


Figure 44 – Box-plot chart for the second user task.

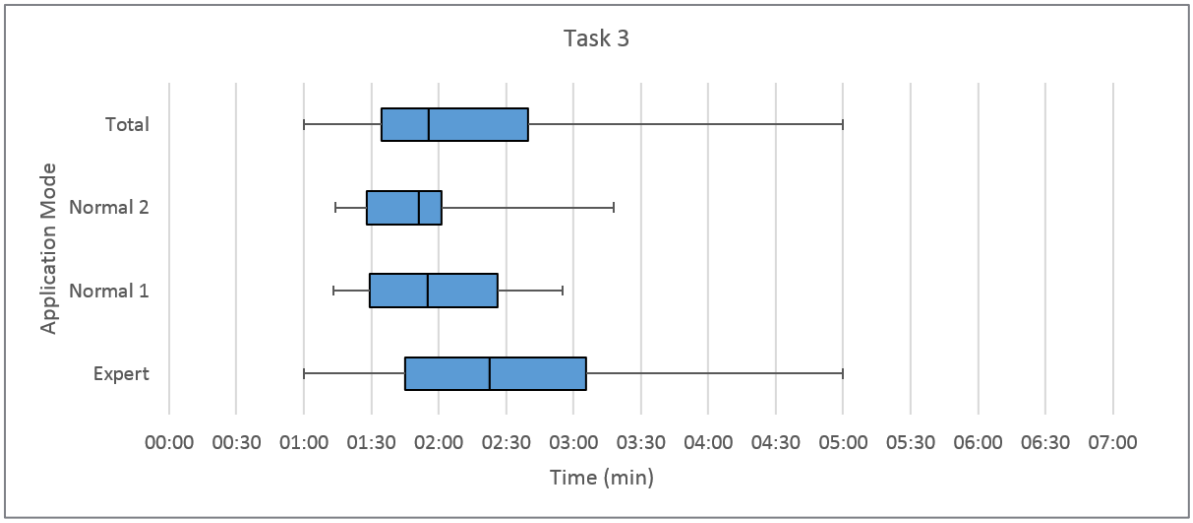


Figure 45 – Box-plot chart for the third user task.

6. Summary

This thesis describes the design, implementation and evaluation of a modern, end-user friendly mobile application that enables non-experts to detect currently relevant and highly problematic security issues in IoT devices. The resulting implementation contains various features that comparable apps are not providing on a sufficient level. Such features include dedicated history functionality which easily lets users track the results of multiple network scans over time by making use of three different, interactive history charts. This implementation also provides improved information gathering by trying to extract human-readable data from various network services which helps users to identify individual devices more easily.

Implemented features that are not found within comparable applications include the three different application modes that allow users to adapt the presented information and available functionality to their level of technical knowledge. Further features include the graphical visualization of networks and inherent vulnerabilities which provides a novel alternative to list-based network representations on smartphones. One of the app´s core features is the vulnerability scanning functionality which enables users to detect the use of standard authentication credentials in common network services. Additionally, the app provides three different scan modes that differ in what kind of tests are performed on devices after they were detected, in order to give the user better control over network scans.

The results of functional tests that were performed by using the app to scan various types of network devices underline the relevance of the vulnerabilities that are covered in this project. Most of the tested devices were susceptible to one or more of the covered vulnerabilities and could be successfully identified as such by the app.

The thesis concludes with the results of a user study which on the one hand was conducted in order to objectively assess how efficiently users could solve tasks with the app in a certain application mode. On the other hand the user study determined how users perceive the app in terms of several user experience aspects after using it in a certain application mode. The results of the study show that participants who used the app in the Normal 1 (reduced information) and Normal 2 (graphical network visualization) modes solved the given tasks slightly faster than participants who used the app in the Expert mode. Furthermore the results of participants that used the app in the Normal modes show less variation, which means that those users finished the tasks with more similar completion times. In terms of usability ratings the app was generally positively perceived. However, the graphical mode was favored over other application modes.

7. Appendix

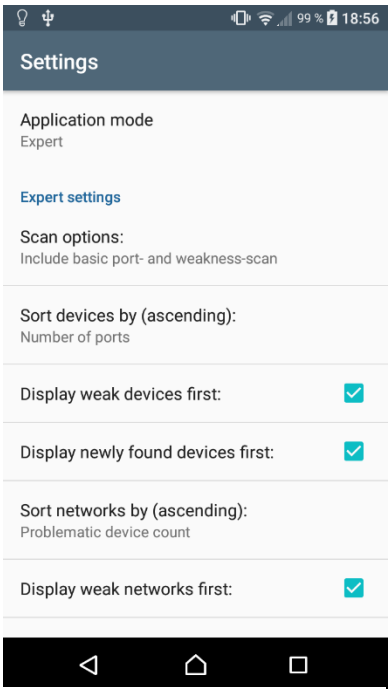


Figure 46 – Settings in the expert mode (all settings are available)

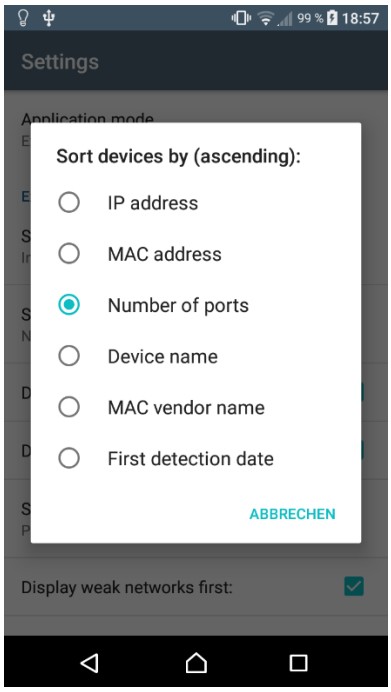


Figure 47 – Selection of one of six device-sorting options

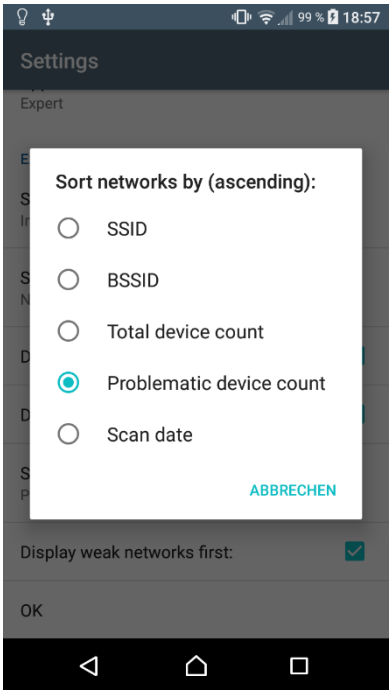


Figure 48 – Selection of one of five network sorting options

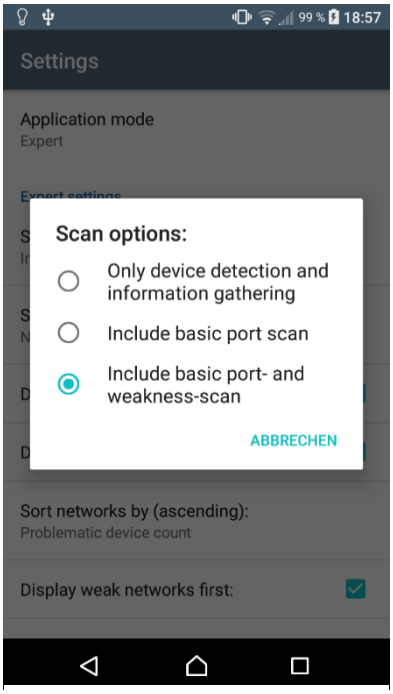


Figure 49 – Selection of one of the three scan options

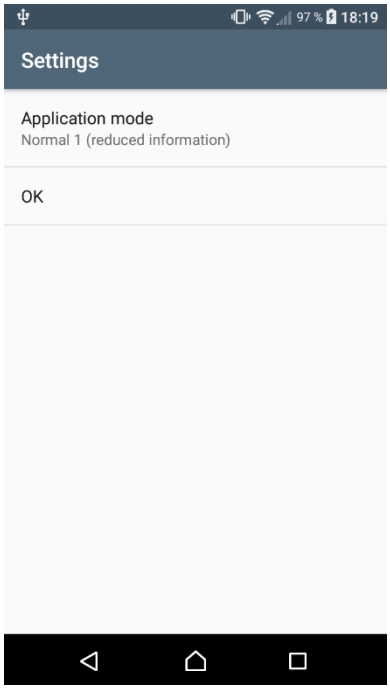


Figure 50 – Settings in the Normal 1 & 2 modes (reduced settings)

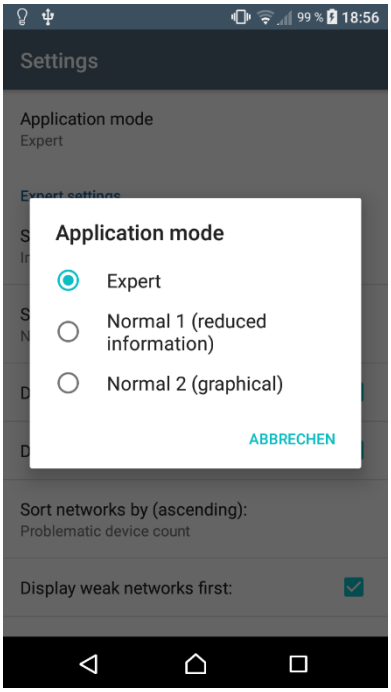


Figure 51 –Selection of one of the three application modes

IoT Security Scanner – Main User Study

General information

Time to complete the survey: 15 minutes (~10 min for the tasks and ~5min for the survey)

Gender: □ Male □ Female

Technical study / experience: □ Yes □ No

Age: _____

User Tasks

Note 1: Please closely read the questions for each task and pay special attention to the underlined parts!

Note 2: The starting point for each task is the applications "Saved networks" view. After each completed task please return to this view (with your devices back-button) and then begin with the next task!

Task 1

Start the task by displaying the last network state of the saved network called "User Study NW". Then take a look at the Mini NAS Server (called "DIGITUS-05C4" in the app): **1.** Briefly describe the problems that have been found with this device. **2.** Briefly describe how the problems could be solved.

Answer 1 (Description of found problems):

Answer 2 (Description of possible solutions):

Completed at: _____

Task 2:

After completing task 1, return to the "Saved networks" view. Then again display the last network state of the saved network called "User Study NW" (as already done in task 1). After that, find the networks WIFI access point (the device that connects all other devices with each other) and note its name.

Answer: _____

Completed at: _____

Task 3

After completing task 2, return to the "Saved networks" view. This time however, display the network history (chart visualization) of the saved network called "User Study NW". Use the chart to find out when (note at which date and time) the scan took place that first revealed a problem with the Lenovo-Laptop (called "JKK-Lenovo" in the app).

Answer: _____ (date / time)

Completed at: _____

User Experience evaluation

Please answer the questions beneath by choosing values from 2 to -2:

2 - I fully agree (excellent)
1 - I agree (very good)
0 - Ok, but ... (good)
-1 - I disagree (bad)
-2 - I fully disagree (very bad)

Aspect	Explanation	Rating
Attractiveness	The app is appealing - I like it (overall rating).	
Efficiency, Understandability	The app is well structured and easy to understand.	
Dependability	The behavior of the app is comprehensible (the app does not confuse).	
Novelty	The app comprises novel aspects and is original.	
Stimulation	The app is interesting, I will likely use it in future.	

Taken from: Efficient Measurement of the User Experience of Interactive Products. M. Rauschenberger, M. Schrepp, M. Perez Cota, S. Olschner, and J. Thomaschewski. International Journal of Artificial Intelligence and Interactive Multimedia, Vol.2, N 1, 2013
(http://www.ijimai.org/journal/sites/default/files/files/2013/03/ijimai20132_15_pdf_35685.pdf)

Figure 52 – User study questionnaire

ID	App Mode	M/F	Tech. exp.	Age	T1 time	T2 time	T3 time	T1 correct	T2 correct	T3 correct	Attractive ness	Efficiency / Understandability	Depend ability	Novelty	Stimulation
1	Expert	M	Yes	19	05:02	01:28	02:50	Yes	Yes	Yes	2	1	2	1	2
2	Expert	M	Yes	20	02:48	01:10	01:42	Yes	Yes	Yes	1	1	1	1	2
3	Expert	M	Yes	18	03:58	00:55	02:36	Yes	Yes	Yes	1	1	2	2	1
4	Expert	M	Yes	22	04:18	02:46	03:11	Yes	Yes	Yes	2	2	2	1	2
5	Expert	M	Yes	23	05:28	01:47	01:55	Yes	Yes	Yes	2	1	2	1	1
6	Expert	M	Yes	34	03:00	02:21	02:09	Yes	Yes	No	1	0	2	2	1
7	Expert	M	Yes	19	05:00	01:00	05:00	Yes	Yes	Yes	1	0	2	2	1
8	Expert	F	Yes	20	02:20	00:50	01:40	Yes	Yes	Yes	1	0	2	2	2
9	Expert	M	Yes	22	01:45	00:55	01:00	Partially	Yes	Yes	2	2	1	1	2
10	Expert	M	Yes	30	01:44	01:28	04:29	Partially	Yes	Yes	-1	1	1	2	2
11	Normal 1	M	Yes	30	03:25	01:58	01:13	Yes	Yes	Yes	1	2	1	1	1
12	Normal 1	M	Yes	21	05:45	01:05	02:30	Yes	Yes	Yes	1	1	0	2	0
13	Normal 1	M	Yes	19	05:38	01:15	01:33	Yes	Yes	Yes	1	2	1	2	2
14	Normal 1	M	Yes	19	01:58	01:37	02:15	Yes	Yes	Yes	1	1	2	0	1
15	Normal 1	M	Yes	19	04:40	00:55	01:45	Yes	Yes	Yes	1	1	1	2	1
16	Normal 1	M	Yes	20	01:33	00:57	01:26	Yes	Yes	Yes	0	2	2	2	1
17	Normal 1	M	Yes	46	02:26	01:07	02:41	Yes	Yes	Yes	2	1	2	1	0
18	Normal 1	M	Yes	25	06:41	01:02	01:28	No	Yes	Yes	2	1	1	2	1
19	Normal 1	M	Yes	21	02:53	01:29	02:05	Yes	Yes	Yes	0	1	2	0	-1
20	Normal 1	M	Yes	21	04:28	01:14	02:55	Yes	Yes	Yes	2	2	1	2	2
21	Normal 2	M	Yes	22	04:00	00:50	02:50	Yes	Yes	Yes	1	2	2	2	1
22	Normal 2	M	Yes	20	02:31	01:19	01:24	Yes	Yes	Yes	2	2	1	1	2
23	Normal 2	M	Yes	26	03:11	01:08	01:53	Partially	Yes	Yes	1	2	1	1	2
24	Normal 2	F	Yes	19	05:11	01:46	02:03	Partially	Yes	Yes	2	2	2	2	2
25	Normal 2	M	Yes	24	05:52	01:08	03:18	Yes	Yes	Yes	1	1	2	1	2
26	Normal 2	M	Yes	22	03:13	01:22	01:40	Yes	Yes	Yes	2	2	2	2	1
27	Normal 2	M	Yes	19	03:23	00:57	01:14	Yes	Yes	No	1	2	1	2	2
28	Normal 2	M	Yes	26	03:09	01:34	01:20	Yes	Yes	Yes	1	1	0	2	0
29	Normal 2	F	Yes	22	03:08	01:16	01:56	Yes	Yes	Yes	1	2	2	1	1
30	Normal 2	M	Yes	21	04:43	02:02	01:49	Yes	Yes	Yes	2	2	2	2	1

Table 18 – Data gathered from the user study

8. Abbreviations

Abbreviation	Full text
API	Application Programming Interface
ARP	Address Resolution Protocol
AS	Autonomous System
BSD	Berkeley Software Distribution
BSSID	Basic Service Set Identifier
CIFS	Common Internet File System Protocol
CRUD	Create Read Update Delete
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FTP	File Transfer Protocol
GB	Gigabyte
GHz	Gigahertz
GNU	GNU's Not Unix
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPS	Intrusion Prevention System
ISP	Internet Service Provider
IoT	Internet of Things
JS	JavaScript
LAN	Local Area Network
LGPL	GNU Lesser General Public License
MAC	Media Access Control
MIT	Massachusetts Institute of Technology
Mbps	Megabits Per Second
NAS	Network Attached Storage
NAT	Network Address Translation
NEU	Non Expert User
NIC	Network Interface Controller
Nmap	Network Mapper
OS	Operating System
OUI	Organizational Unique Identifier
OWASP	Open Web Application Security Project

PPP	Point-to-Point Protocol
SCDP	Service Control Point Definition
SDK	Software Development Kit
SDN	Software Defined Networking
SIEM	Security Information and Event Management
SMB	Server Message Block Protocol
SNMP	Simple Network Management Protocol
SOHO	Small Office, Home Office
SQL	Structured Query Language
SSDP	Simple Service Discovery Protocol
SSH	Secure Shell Protocol
SSID	Service Set Identifier
TCP	Transmission Control Protocol
TNV	Time-based Network Traffic Visualizer
UI	User Interface
UPnP	Universal Plug and Play
WAN	Wide Area Network
XML	Extensible Markup Language

9. References

[1] P. Paganini, "Linux/Mirai ELF, When Malware is Recycled Could be Still Dangerous," Security Affairs, 5 September 2016. [Online]. Available: http://securityaffairs.co/wordpress/50929/malware/linux-mirai-elf.html. [Accessed June 2020].

[2] S. Hilton, "Dyn Analysis Summary of Friday October 21 Attack," Dyn, 26 October 2016. [Online]. Available: https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/. [Accessed June 2020].

[3] D. McMillen and M. Alvarez, "Mirai IoT Botnet: Mining for Bitcoins?," SecurityIntelligence, 10 April 2017. [Online]. Available: https://securityintelligence.com/mirai-iot-botnet-mining-for-bitcoins/. [Accessed June 2020].

[4] A. Gilchrist, "Chapter 3 - Flawed, Insecure Devices," in *IoT Security Issues*, De|G Press, pp. 35-36, 2017.

[5] L. O'Donnell, "At CES, Focus is on 'Cool Factor' not IoT Security," Threatpost, 10 January 2019. [Online]. Available: https://threatpost.com/at-ces-focus-is-on-cool-factor-not-iot-security/140767. [Accessed June 2020].

[6] A. Tannenbaum, "Why do IoT Companies Keep Building Devices with Huge Security Flaws?," Harvard Business Review, 27 April 2017. [Online]. Available: https://hbr.org/2017/04/why-do-iot-companies-keep-building-devices-with-huge-security-flaws. [Accessed June 2020].

[7] J. Nazario, "The Problem with Patching in Addressing IoT Vulnerabilities," fastly.com, 29 August 2017. [Online]. Available: https://www.fastly.com/blog/problem-patching-addressing-iot-vulnerabilities. [Accessed June 2020].

[8] N. Buchka, "Switcher: Android Joins the 'Attack-the-Router' Club," Securelist, 28 December 2016. [Online]. Available: https://securelist.com/switcher-android-joins-the-attack-the-router-club/76969/. [Accessed June 2020].

[9] R. M. Ogunnaike and B. Lagesse, "Toward Consumer-Friendly Security in Smart Environments," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 1-6*, 2017.

[10] G. Jonsdottir, D. Wood and R. Doshi, "IoT Network Monitor," in *2017 IEEE MIT Undergraduate Research Technology Conference (URTC), pp. 1-5*, 2017.

[11] Y.-r. SU, X.-f. LI, S.-f. WANG, J. YI and H.-r. HE, "Vulnerability Scanning System Used in the Internet of Things for Intelligent Devices," in *2nd International Conference on Communications, Information Management and Network Security (CIMNS 2017), pp. 1-6*, 2017.

[12] "Mobile Penetration Testing Toolkit and Risk Assessment," ZIMPERIUM, [Online]. Available: https://www.zimperium.com/zanti-mobile-penetration-testing. [Accessed June 2020].

[13] "fing.com," Fing, [Online]. Available: https://www.fing.com/. [Accessed June 2020].

[14] "ezNetScan," VRSSPL, [Online]. Available: https://play.google.com/store/apps/details?id=com.vrsspl.android.eznetscan.plus. [Accessed June 2020].

[15] N. Circelli, "Net Scan," [Online]. Available: https://play.google.com/store/apps/details?id=com.wwnd.netmapper. [Accessed June 2020].

[16] "Dojo by BullGuard," Bullguard Israel Ltd, [Online]. Available: https://play.google.com/store/apps/details?id=dojo.dojo. [Accessed June 2020].

[17] "Kaspersky Smart Home & IoT Scanner," Kaspersky Lab Switzerland, [Online]. Available: https://play.google.com/store/apps/details?id=com.kaspersky.iot.scanner. [Accessed June 2020].

[18] "IoT Security (Guard Internet of Things Devices)," Cheetah Mobile, [Online]. Available: https://apkpure.com/iot-security-%EF%BC%88guard-internet-of-things-devices%EF%BC%89/com.cheetahmobile.iotsecurity. [Accessed June 2020].

[19] D. Novikov, "IoPT: Network Security Scanner," [Online]. Available: https://play.google.com/store/apps/details?id=pro.dnovikov.iopt. [Accessed June 2020].

[20] "OWASP Internet of Things Project," OWASP, 2018. [Online]. Available: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project. [Accessed June 2020].

[21] M. Kuzin, Y. Shmelev and V. Kuskov, "New Trends in the World of IoT Threats," Securelist, 18 September 2018. [Online]. Available: https://securelist.com/new-trends-in-the-world-of-iot-threats/87991/. [Accessed June 2020].

[22] S. Boddy, J. Shattuck, D. Walkowski and D. Warburton, "The Hunt for IoT: Multi-Purpose Attack Thingbots Threaten Internet Stability and Human Life," F5 Labs, 24 October 2018. [Online]. Available: https://www.f5.com/labs/articles/threat-intelligence/the-hunt-for-iot--multi-purpose-attack-thingbots-threaten-intern. [Accessed June 2020].

[23] "OWASP Internet of Things Project - IoT Top 10," OWASP, 2018. [Online]. Available: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10. [Accessed June 2020].

[24] H. Moore, "Security Flaws in Universal Plug and Play," Rapid 7, 29 January 2013. [Online]. Available: https://information.rapid7.com/rs/411-NAK-970/images/SecurityFlawsUPnP%20%281%29.pdf. [Accessed June 2020].

[25] P. Gomér and J.-E. Johnzon, "Computer Network Analysis by Visualization," Chalmers University of Technology / Department of Computer Science and Engineering (Chalmers), pp. 32-37, 2010.

[26] M. Dean and L. Vespa, "Simplified Network Traffic Visualization for Real-Time Security Analysis," in *The 2013 International Conference on Security and Management, pp. 1-5*, 2013.

[27] R. Tamassia, B. Palazzi and C. Papamanthou, "Graph Drawing for Security Visualization," in *The 16th International Symposium on Graph Drawing (GD 2008), pp. 1-12*, 2008.

[28] F. Mansman, L. Meier and D. A. Keim, "Visualization of Host Behavior for Network Security," in *The 4th International Workshop on Computer Security (VizSec 2007), pp. 6-14*, 2007.

[29] Y. Livnat, J. Agutter, S. Moon, R. F. Erbacher and S. Foresti, "A Visualization Paradigm for Network Intrusion Detection," in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, pp. 1-8*, 2005.

[30] P. A. Legg, "Visual Analytics for Non-Expert Users in Cyber Situation Awareness," in *2016 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (CyberSA), pp. 8-15*, 2016.

[31] "Suricata 2.0 Available!," Suricata, 25 March 2014. [Online]. Available: https://suricata-ids.org/tag/logstash/. [Accessed June 2020].

[32] "The Elastic Stack," Elastic, [Online]. Available: https://www.elastic.co/products. [Accessed June 2020].

[33] "KIBANA The Elastic Stack," Elastic, [Online]. Available: https://www.elastic.co/products/kibana. [Accessed June 2020].

[34] "Intrusion Detection Software with Threat Monitor - IT Ops Edition," Solarwinds, [Online]. Available: https://www.solarwinds.com/topics/intrusion-detection-software. [Accessed June 2020].

[35] "OSSIM Download – Open Source SIEM Tools & Software," Darknet, 20 October 2017. [Online]. Available: https://www.darknet.org.uk/2017/10/ossim-download-open-source-siem-tools-software/. [Accessed June 2020].

[36] "ArcSight Investigate - Security Investigation and Analytics," Micro Focus, [Online]. Available: https://www.microfocus.com/en-us/products/arcsight-investigate/overview. [Accessed June 2020].

[37] "Splunk Enterprise," Splunk, [Online]. Available: https://www.splunk.com/en_us/software/splunk-enterprise.html. [Accessed June 2020].

[38] "IBM QRadar SIEM," IBM, [Online]. Available: https://www.ibm.com/us-en/marketplace/ibm-qradar-siem. [Accessed June 2020].

[39] P. Manev, "Let’s Talk About SELKS 2.0," Stamus Networks, 6 May 2015. [Online]. Available: https://www.stamus-networks.com/2015/05/06/lets-talk-about-selks-2-0/. [Accessed June 2020].

[40] "Cisco Firepower Management Center Demos," Cisco, [Online]. Available: https://www.cisco.com/c/en/us/products/security/firepower-management-center/demos.html. [Accessed June 2020].

[41] "Snorby Introduction," 1 November 2013. [Online]. Available: https://www.aldeid.com/wiki/Snorby#Introduction. [Accessed June 2020].

[42] "Keeping the Wolves at Bay," Meraki Cisco, 25 September 2014. [Online]. Available: https://meraki.cisco.com/blog/2014/09/keeping-the-wolves-at-bay/. [Accessed June 2020].

[43] A. Zahariev, "Graphical User Interface for Intrusion Detection in Telecommunications Networks," Aalto University / School of Science, p. 42, 2011.

[44] J. R. Goodall, W. G. Lutters, P. Rheingans and A. Komlodi, "Preserving the Big Picture: Visual Network Traffic Analysis with TNV," in *IEEE Workshop on Visualization for Computer Security, 2005. (VizSEC 05), pp. 3-4*, 2005.

[45] "OUI List," IEEE, [Online]. Available: http://standards-oui.ieee.org/oui.txt. [Accessed June 2020].

[46] "Nmap," [Online]. Available: https://nmap.org/. [Accessed June 2020].

[47] "OKHttp," Square, [Online]. Available: http://square.github.io/okhttp/. [Accessed June 2020].

[48] C. Sadun, "Telnet Client Library," [Online]. Available: http://sadun-util.sourceforge.net/telnet_library.html. [Accessed June 2020].

[49] "Jsch," Jcraft, [Online]. Available: http://www.jcraft.com/jsch/. [Accessed June 2020].

[50] "Apache Commons Net," The Apache Software Foundation, [Online]. Available: https://commons.apache.org/proper/commons-net/. [Accessed June 2020].

[51] J. Hedley, "Jsoup," [Online]. Available: https://jsoup.org/. [Accessed June 2020].

[52] "JCIFS," The JCIFS Project, [Online]. Available: https://jcifs.samba.org/. [Accessed June 2020].

[53] F. Fock and J. Katz, "SNMP4J," [Online]. Available: https://www.snmp4j.org/. [Accessed June 2020].

[54] P. Jahoda, "MPAndroidChart," [Online]. Available: https://github.com/PhilJay/MPAndroidChart. [Accessed June 2020].

[55] M. Bostock, "D3 JS," [Online]. Available: https://d3js.org/. [Accessed June 2020].

[56] "TransactionTooLargeException," Google, [Online]. Available: https://developer.android.com/reference/android/os/TransactionTooLargeException. [Accessed June 2020].

[57] "Mirai Source Code," 2016. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code/blob/master/mirai/bot/scanner.c. [Accessed June 2020].

[58] A. Hemel, "UPnP IGD Hacking," UPnP Hacks, [Online]. Available: http://www.upnp-hacks.org/igd.html. [Accessed June 2020].

[59] A. Lockwood, "Handling Configuration Changes with Fragments," Android Design Patterns, 29 April 2013. [Online]. Available: https://www.androiddesignpatterns.com/2013/04/retaining-objects-across-config-changes.html. [Accessed June 2020].

[60] K. Petric, "freesshd.com," [Online]. Available: http://www.freesshd.com/. [Accessed June 2020].

[61] M. Rauschenberger, M. Schrepp, M. P. Cota, S. Olschner and J. Thomaschewski, "Efficient Measurement of the User Experience. How to use the User Experience Questionnaire (UEQ)," *International Journal of Interactive Multimedia and Artificial Intelligence, pp. 1-7,* 2013.

JOHANNES KEPLER
UNIVERSITY LINZ

CURRICULUM VITAE

Kai Knabl

kai.knabl@gmail.com

Studies / Education

2014 – today	**JKU Linz** *Master study, Computer Science with the major subject „Networks and Security“*
2010 – 2014	**FH Hagenberg** *Bachelor of Science, Software Engineering*
2004 – 2008	**BORG Ried im Innkreis** *AHS Matura*

Professional experience / Internships / Study projects

04/2019 – today	**CBC-X GmbH, Leonding** *Software Quality & Test Engineer, Application Security Tester*
03/2017 – 09/2017	**Hotel Reichmann, St. Kanzian am Klopeinersee** *Requirements Engineer and Technical Consultant for coordinating major changes in the technical infrastructure (hardware-, software and network solutions) of two hotels and one restaurant (family business)*
10/2013 – 03/2016	**ReqPOOL GmbH, Linz** *Software Developer, Software Tester, Requirements Engineer*
07/2013 – 10/2013	**KTM Sportmotorcycle GmbH, Mattighofen** *Penetration Tester, Creation of the theoretical bachelor thesis in cooperation with the KTM GmbH*
04/2013 – 06/2013	**ReqPOOL GmbH, Linz** *Internship as Software Developer, Creation of the practical bachelor thesis in cooperation with the ReqPOOL GmbH*
03/2012 – 06/2012 10/2012 – 01/2013	**Profactor GmbH, Steyr** *Software Developer, Study project in cooperation with the Profactor GmbH in the summer- and winter-semesters 2012*

All summers
2005 – 2008
2010 – 2018

Hotel Reichmann, St. Kanzian am Klopeinersee
Various activities in the hotel and restaurant business

Certifications

IREB CPRE Foundation Level

ISTQB CT Foundation Level

Community Service

07/2009 – 06/2010

Arbeiter Samariterbund, Alkoven
Paramedic (Nine regular + three voluntary months)

JOHANNES KEPLER
UNIVERSITY LINZ

STATUTORY DECLARATION

I hereby declare that the thesis submitted is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references. This printed thesis is identical with the electronic version submitted.

Place, Date

Signature