SCRATCh

SECURE AND AGILE CONNECTED THNGS

Deliverable 2.4

0

Deliverable 2.4: Security feedback components and tools

Work package:	WP 2.4
Affected milestone:	M24
Partners involved:	DFKI
	OTARIS
Date:	15/03/2022
Deliverable version:	v1.0
Editor:	Duque Anton, Simon – DFKI
Author(s):	Duque Anton, Simon – DFKI
	Reti, Daniel – DFKI
	Schneider, Daniel – DFKI
	Sohr, Karsten - OTARIS
Responsible Contact:	Duque Anton, Simon
	DFKI GmbH
	Trippstadter Straße 122, DE-67655, Kaiserslautern
	Daniel.reti@dfki.de
	+49 631 20575 3012

Version history

Date	Version	Author	Comment
05/09/2020	0.3	Duque Anton, Simon	First draft
01/10/2020	0.9	Duque Anton, Simon	First final draft

15/03/2022	1.0	Daniel Reti	Final revision

Table of Contents

1. Introduction

The secure operation of information systems is a crucial aspect in any organisation. Distributed and heterogeneous systems in particular, such as Internet of Things (IoT)-networks, contain a large attack surface. In the past, IoT-devices have been prone to attacks that are based on poorly implemented security measures, such as standard credentials, hardcoded passwords or developer access. As Iot-devices are highly interconnected by definition, an attacker can potentially extend the hold on a system once an initial IoT-device is compromised. Consumers and commercial organisations alike have little means to increase their security measures. Ideally, vendors would provide devices that follow best security practices and are implemented accordingly. Since this is hard to achieve, increasing the system security is the next best thing: By providing intrusion detection and prevention mechanisms, a user is capable of detecting and isolating an attack and thus prevent it from spreading and causing harm. The research project SCRATCh acknowledges the increasing introduction of IoT-devices into networks, in commercial and private settings, while addressing the need to ensure secure operation. Even though there is no perfect security, tools to increase the operation security and thus minimise negative impact from the most common attacks are developed and evaluated on real use cases.

In Section 2, a state of the art in intrusion detection is discussed, with a strong focus on IoT-networks. Section 3 presents the tools that have been developed in the context of SCRATCh. Section 4 summarises the document.

2. State of the Art Intrusion Detection

2.1 IoT challenges

Intrusion detection systems (IDSs) are an established and well-researched tool in network security. However, they cannot be applied as-is in the context of IoT, due to several reasons. First, IoT devices often have tight resource constraints, which makes it difficult to find a node able to operate an IDS. Second, IoT end systems are often responsible for their own packet-forwarding instead of using switches or routers. Third, IoT networks use different network protocols, such as IPv6 over Low-power Wire-less Personal Area Network (6LoWPAN) or Constrained Application Protocol (CoAP) (Zarpelão et al., 2017).

2.2 IDS Taxonomy

In a systematic literature review, Hajiheidari et al. (2019) largely confirm the taxonomy presented in Zarpelão et al. (2017), but include additional security threats. Since systematic literature reviews focus on rigor and reliability, they provide more reliable evidence than non-systematic surveys. The resulting taxonomy is presented in table 1.

Table 1: Taxonomy of IDSs presented in scientific articles.

Placement Strategy	Detection Method	Security Thread	Validation Strategy
Distributed/Network-based (NIDS)	Signature-based	DoS	Hypothetical
Centralized/Host-based (HIDS)	Anomaly-based	Replay	Empirical
Hybrid	Hybrid	Selective forwarding	Simulation
		Sybil	Theoretical
		Wormhole	
		Black Hole	
		Sinkhole	
		Jamming	
		False Data	

2.3 Strengths and weaknesses

Building on the the review of Hajiheidari et al. (2019), table 2 shows the major advantages and disadvantages of different IDS concepts. A limitation worth noting is that the review is restricted to peer-reviewed scientific articles published in the IoT field.

One issue in the adaption of IDSs for IoT stems from the low resources available to IoT devices, for which conventional IDSs are too computationally expensive and memory intensive, to the point where they can cause network failure. Among the proposed IDSs which successfully addressed this issue, most are anomaly-based.

A further issue is that the communication overhead generated by IDSs is costly for sensor nodes. Anomaly detection is best suited to reduce communication overhead and network load, as well as prolonging network lifetime.

As IoT networks can become very large, scalability and robustness are important. Both are mainly adressed by anomaly-based IDSs, but only in a small number of works.

A major gap in existing research is that many IDSs can only detect a very limited number of attacks, with some only detecting one specific attack. This is a theoretical weakness of

signature-based IDSs, but also apparent in practical implementations of both anomaly- and specification-based IDSs.

Network topology in IoT contexts is often fast-changing and dynamic. This necessitates a certain adaptability, which is adressed by a small number of anomaly-based IDSs as well as specification-based works.

Generally, lot of research suffers from a lack of real-world implementation or even simulation. This makes it difficult to evaluate the real-life merit of proposed concepts.

2.4 Conclusion

Based on the findings regarding IDSs for IoT-environment, the need for tailored security solutions becomes apparent. Anomaly-based approaches lend themselves readily due to their capability to detect attacks in heterogeneous and changing environments, especially attacks which are yet unknown. Furthermore, defining rules to discover which data is used and transmitted where is crucial in order to preserve privacy of critical information. In the following section, the tools for increasing security feedback that have been developed in SCRATCh are introduced.

Table 2: Strenghts and weaknesses of IDSs presented in scientific articles, categorized by detection method and ordered by occurence.

Detection Method [#]	Strengths	#	Weaknesses	#
Anomaly Based [22]	High detection accuracy	16	High resource consumption	7
	Low resource consumption	12	High false positive rate	5
	Low false positive rate	9	Few attack types	4
	Real-time IDS	8	High time overhead	4
	Low communication overhead	4	Long training stage	2
	Robustness (updates, node failures)	4	Low detection accuracy	2
	High adaptability	3	Resource-intensive training stage	2
	Low latency	3		
	Improved network lifetime	2		
Signature Based [3]	High detection accuracy	4	Few attack types	3
	Low resource consumption	2	High network load	1
	Low false positive rate	1	Not real-time	1
	High scalability	1	High memory consumption	1
	Low latency	1	High false positive rate	1
Specification based [10]	High detection accuracy	7	High complexity	4
	Low resource consumption	3	High false positive rate	4
	Low false positive rate	3	High resource consumption	4
	Low latency	2	Low detection accuracy	3
	High adaptability	2	Not real-time	2
	Real-time IDS	2	High latency	2
	Increased data delivery rate	1	Few attack types	1
	Low complexity	1	High communication overhead	1
	High scalability	1	Low scalability	1

3. Security Feedback Tools Developed in SCRATCh

This section addresses security feedback solutions that have been developed in the course of project SCRATCh. First, anomaly detection is presented as a concept. After that, tools to detect botnet activities in IoT networks and a tool that helps to analyze network data of (IoT-)devices with respect to privacy and security
are discussed.

3.1 Anomaly Detection

Anomaly-based intrusion detection employs a pre-learned system model. Normal instances of events, e.g. normal TCP/IP-packets, are conformant to the learned model, while attacks, e.g. a TCP-Syn-flood, deviate from it. Thus, no explicit rules are required. The advantage of anomaly-based approaches lies in the automated generation of models. Furthermore, formerly unknown attacks, meaning no rules for their detection exist, performed against the system can be detected as deviations from the model. This feature provides protection against so-called 0day-attacks. The quality of an anomaly detection tool depends on the data used for training. First, a sufficient amount of data is commonly necessary in order for the tool to learn the model. This data is called training data. Second, the quality of the training data needs to be high. That means it needs to contain examples of any event that might occur during normal operation. Else, new occurrences of normal events can be flagged as an anomaly. Third, the extent of the data is relevant, i.e. which information is contained in the data. The data should contain all relevant information for the system in order for the learned model to be expressive. In the course of project SCRATCh, anomaly-based intrusion detection was applied to the detection of botnets in IoT environments. The next section describes this task and provides information about the types of data used and what attacks can be derived from them.

3.2 Botnet Protection

IoT devices have been targeted by botnets since IoT became established (source: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/into-the-battlefield-a-security-guide-to-iot-botnets). From attacking routers, botnets developed to infecting any internet-connected device. The intentions of attackers are usually focussed on financial gain: price and content scraping as well as account aggregation and hacking or stealing of privileged information are main goals of botnets. Additionally, Distributed Denial of Service (DDoS) attacks can be carried out by botnets, rendering the target resources incapable because of millions of malicious requests. Furthermore, bots can be added to crypto-mining pools to generate cryptocurrency for an attacker. Such attacks are a nuisance for home appliances. However, in a commercial environment, falling prey to a botnet can lead to the organisation being held liable, in addition to reputational damage. Consequently, detection of and defence against botnets is a crucial aspect of IoT security, placed in the operate-phase of the DevOps life-cycle.

MQTT is a protocol, commonly employed in IoT environments that works in a publisher-subscriber fashion. Several clients are connected to a broker, they publish information

and subscribe to the publication of information. If an attacker successfully infects an IoT-device that is part of a MQTT-network, this device can:

- Extract information by subscribing to information types provided by the broker,
- inject information by publishing it to the broker.

Injecting information can be used to exhaust the brokers resources or to create false information for monitoring devices or applications that require information. Furthermore, the infected client can be used for malicious activities, such as mining of crypto-currency.

In the course of project SCRATCh, methods for detecting attacks in MQTT-networks were planned to be researched and developed. While experiments have been done with the aim to detect anomalies in the firmware update process of encrypted pin pads with different machine learning algorithms for supervised anomaly based IDS, such as Long Short Term Memory (LSTM) or Multi-Layer Perceptron (MLP), Decision Trees such as Random Forests, k-means clustering or Support Vector Machines, the resulted accuracy has not been satisfyingly high.

3.2.1 Detection Methods

Malicious activity in IoT botnets located in IoT environments can detected by analysing different kinds of information that will be integrated into the security feedback tool. Sources of information to detect attacks are:

- **Power usage**: this information can be proved by Almende's Crownstones. Power usage at unlikely times or more usage than similar devices is an indicator of increased device activity.
- **Channels subscribed**: The channels a client is subscribed or publishing to can indicate malicious behaviour, especially in comparison with devices of the same type.
- **Communication**: Communication with the Internet and external entities can indicate malicious behaviour, e.g. Command and Control (C2C) communication or data exfiltration, as shown in Figure 1.

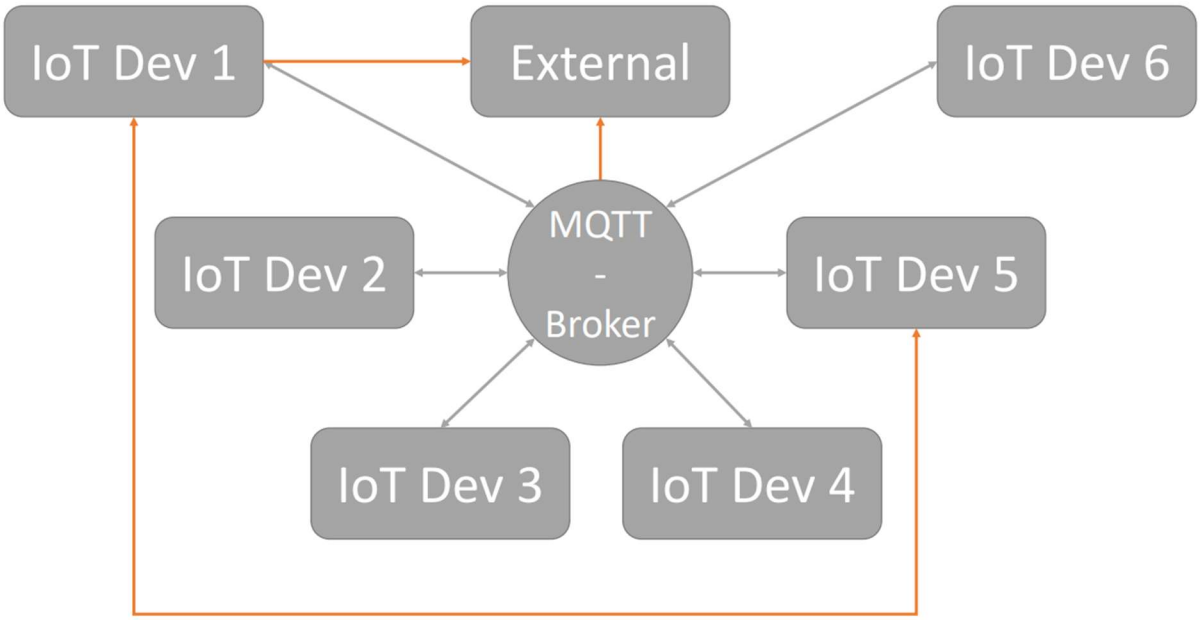

FIGURE 1: EXEMPLARY COMMUNICATION PATTERNS

3.3 OTAlyzer

In the following, the OTAlyzer is introduced, which serves the purpose of an advanced network analysis tool for IoT devices. Its results can be used by developers to improve their code based on the analysis results (in a feedback loop).

3.3.1 Overview

Currently, IoT devices handle a lot of personal data, while becoming more and more omnipresent. At the same time, software is getting more complex and has to be deployed more quickly and developers often rely on third-party libraries to solve specific problems, while not being aware that these might leak sensitive data to application servers. The OTARIS Traffic Analyzer (in short OTAlyzer) is a tool for security testers and developers alike that helps to analyze network data of (IoT-)devices with respect to privacy and security.

The OTAlyzer consists of two applications, serving different use cases: on the one hand, the standalone binary can be used in DevSecOps pipelines to ensure that a change in code does not cause sensitive data to leak (e.g. by including another library or by bad code design). Developers can then detect these flaws before they get incorporated into the product, resulting in a more secure product. The OTAlyzer Test Suite on the other hand includes data visualizations, project & device management and a centralized database for test reports, enabling professional security testers to handle a complex privacy audit of several devices, programs and test cases.

Using a recorded packet capture file (in the form of a .pcapng or .mitm file)[1] of the tested device, the OTAlyzer can detect different kinds of data that are being leaked in the recorded traffic. Search keywords are generated based upon the supplied

[1] PCAP files may be produced by the Wireshark tool and mitm files come from man-in-the-middle test tools.

specifications of the device that was tested (e.g. its screen size, operating system, used credentials or the used WiFi's BSSID). Adding a custom testing device with custom properties can be done via configuration files.

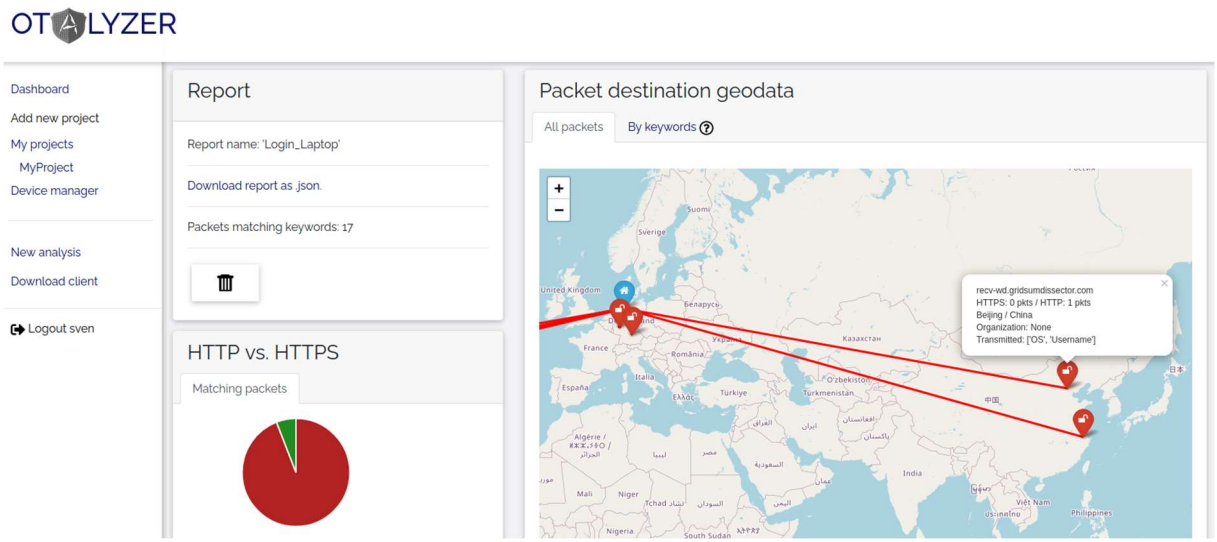

FIGURE 2: THE OTALYZER GUI

Each finding's impact on privacy is then graded using a scoring system from 0-10 for a quick risk assessment overview. Packet flows and details are visualized for each report on the web-UI (see Figure 2).

3.3.2 Features

Devices are managed using a modular approach (a tested smartphone has the specifications of a SIM card, a phone and personal data used in the test, etc.). This promotes re-usability of configurations and makes customization easy. Of course, custom keywords (literal or as regular expressions) can be added as well.

The OTAlyzer can detect leaked data in plain text and in obfuscated form, by probing the recorded traffic for these keywords in various kinds of hashings and encodings. In addition, it can also analyze traffic that is encrypted via TLS/SSL, by decryption of HTTPS packets, if a TLS/SSL key log file is supplied, as well as MQTT traffic.

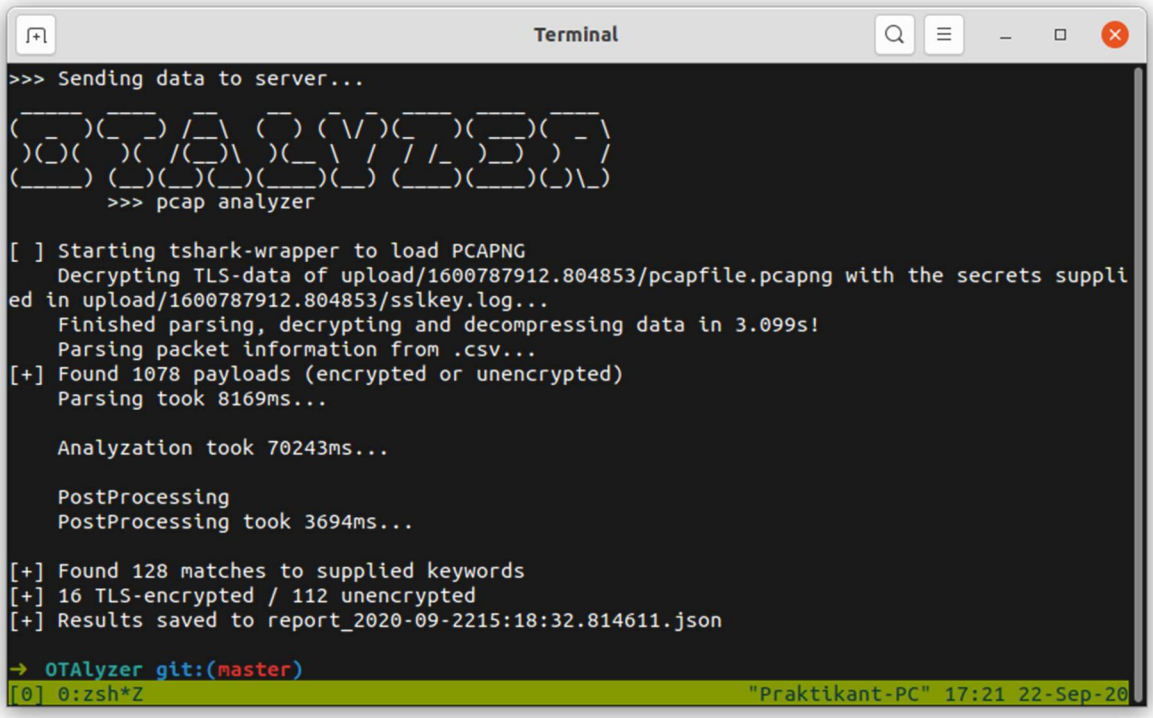

FIGURE 3: RPC CLIENT

The generation of reports can be automated with scripts by either using the RPC client (see Figure 3) to call the API of a remote OTAlyzer Test Suite instance, e.g., running in a container on a local server or by using the command line binary for local or pipeline tests.

3.4 Deception Toolkit (ESCA)

The Deception Toolkit developed in SCRATCh will use various deception techniques on different networking layers combined into one framework named ESCA, derived from the Latin word for 'bait'. The ESCA framework is meant to be placed as a reverse proxy, meaning that every traffic from and to the system it shields will go through the proxy.

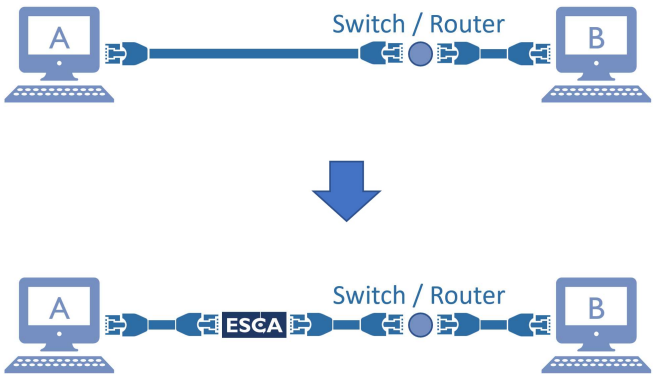

FIGURE 4: ESCA DEVICE FUNCTIONING AS A REVERSE PROXY

This way the ESCA device can apply deception methods on the network traffic in a plug-and-play fashion and the shielded system does not need to be altered. ESCA aims to enable security-related services based on deception without influencing neither

workflow nor performance of existing processes. This configuration allows the ESCA platform to be generalizable, as it is platform-independent and Ethernet-based.

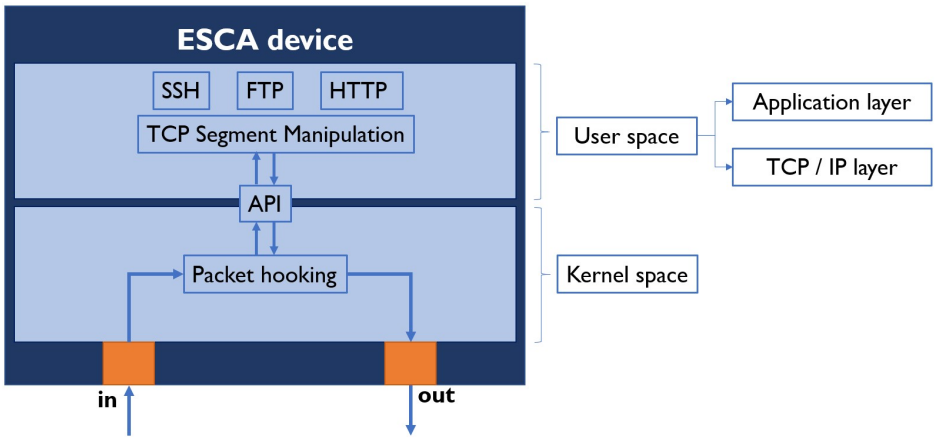

FIGURE 5: ESCA ARCHITECTURE

Within SCRATCh the focus lied on developing three modules, of which two operate on the application layer and one operates on the TCP/IP layer, as seen in figure 2. Incoming packets are hooked using the netfilter queue API, and sent to the modules in user space where information is manipulated by the respective modules and the packets are sent back to the kernel space for further processing.

One module operating on the TCP/IP layer is the OS (Operating System) spoofing module. The goal was to change the TCP fingerprint, which is unique to each Operating System, so that OS fingerprinting software will detect a false operating system. Two main strategies for OS fingerprinting exist, active OS fingerprinting where specially crafted probing packets are sent to the target host to derive the OS from the unique behavior, and passive OS fingerprinting where captured network traffic is used to derive the OS. Within SCRATCh the OS spoofing has been developed to a point where passive fingerprinting tool p0f has been successfully deceived. Approaches to achieve the same on active probing with nmap have not been successful.

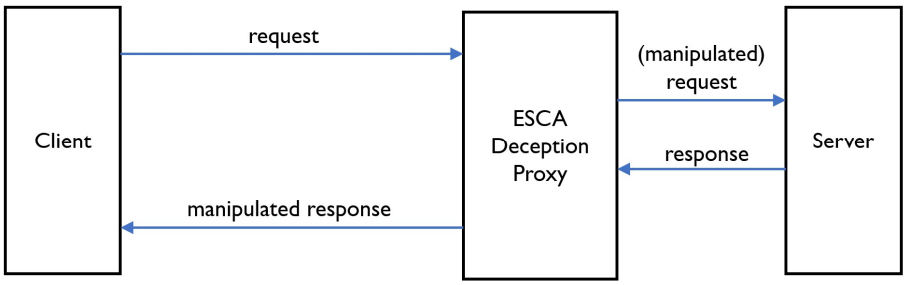

FIGURE 6: TRAFFIC FLOW DIAGRAM

In addition to the OS spoofing module, two deception modules operating on the application layer have been developed. One is intercepting SSH (Secure Shell) packets and the other intercepting http(s) traffic [4,5]. The SSH deception module scans the traffic for specific shell commands that are typical for adversarial behavior and injects false traffic to the response. One example is that when files are listed with the ls

command, in the response certain files can be hidden and other files can be shown that don't actually exist on the host. As another example a fake Linux distribution and kernel version can be spoofed.

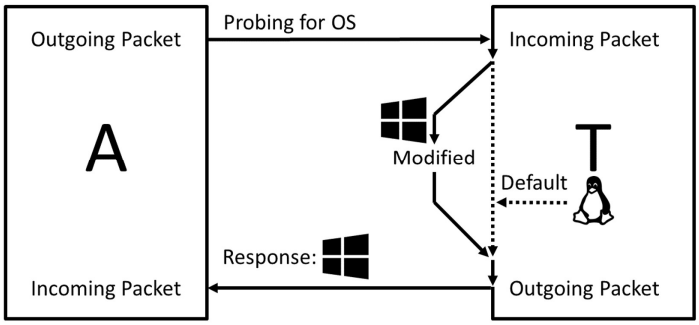

FIGURE 7: OS SPOOFING

The HTTP (Hypertext Transfer Protocol) deception module shields a host that is running a web base application against various types of web-based reconnaissance attacks, by injecting false hints and hidden files at various locations. As an example, when a non-existing resource that would normally respond with a 404 error is requested, instead a fake login-page is be presented. This way the attacker is distracted from real resources, can be detected due to the interaction and stolen credentials might be detected. To help adversaries find the fake resource, the proxy might inject references in for of HTML comments on other locations of the web site or in special locations like the sitemap or the robots exclusion file.

4. Summary

In the context of SCRATCh, three approaches for security feedback tools have been developed and evaluated: An anomaly detection-based tool that automatically learns the normal behaviour of a system and thus detects deviations. Furthermore, a data flow protection tool has been developed, that detects the misuse of information which a user can define by analysing the traffic flow. Finally, a deception toolkit has been developed to detect attackers. As no valid user would interact with a deception resource, it is a binary sensor for a successful attack on the one hand. On the other, it is used to bind and exhaust attackers' resources, so that an attacker will invest time and resources in targets that contain no value. Additionally, it provides insight about goals and methods of an attacker and can be used to provide threat intelligence.

5. References

[1] Hajiheidari, S., Wakil, K., Badri, M., & Navimipour, N. J. (2019). Intrusion detection systems in the internet of things: A comprehensive investigation. Comput. Networks, 160, 165–191.

[2] Zarpelão, B. B., Sanches Miani, R., Toshio Kawakani, C., & Carlisto de Alvarenga, S. (2017). A survey of intrusion detection in internet of things. J. Netw. Comput. Appl., 84, 25–37.

[3] SCRATCh (2020): Creating it from SCRATCh: A Practical Approach for Enhancing the Security of IoT-Systems in a DevOps-enabled Software Development Environment. In: Proceedings of the 1st International Workshop on Underpinnings for Safe Distributed AI. International Workshop on Underpinnings for Safe Distributed AI (USDAI-2020), SAFECOMP2020 September 15-15 Online Springer 2020

[4] D. Reti, D. Klaassen, S. D. Duque Anton, H. D. Schotten. (2021) "Secure (S)Hell: Introducing an SSH Deception Proxy Framework" in IEEE International Conference on Cyber Situational Awareness, Data Analytics and Assessment 2021

[5] Online: https://github.com/SCRATCh-ITEA3/SCRATCh-Tools-Repo/tree/master/C3_Control/07_Operate/Deception_Toolkit

