Poster: Snout - An Extensible IoT Pen-Testing Tool

John Mikulskis
jkulskis@bu.edu
Boston University
Boston, MA, USA

Johannes K Becker
jkbecker@bu.edu
Boston University
Boston, MA, USA

Stefan Gvozdenovic
tesla@bu.edu
Boston University
Boston, MA, USA

David Starobinski
staro@bu.edu
Boston University
Boston, MA, USA

ABSTRACT

Network mapping tools designed for IP-based networks generally do not provide access to non-IP based wireless protocols used by Internet of Things (IoT) devices, such as Zigbee and Bluetooth LE. We present Snout, a versatile and extensible software defined radio-based tool for IoT network mapping and penetration testing. Snout is geared towards the various IoT protocols that are not accessible with traditional network enumeration tools, such as Nmap. The tool allows for device enumeration, vulnerability assessment, as well as more offensive techniques such as packet replay and spoofing, which we demonstrate for the Zigbee protocol. Snout is built on an open-source stack, and is designed for extensibility towards other IoT protocols and capabilities.

CCS CONCEPTS

• **Security and privacy** → **Penetration testing**; **Vulnerability scanners**; **Mobile and wireless security**; • **Networks** → *Cross-layer protocols*; *Network performance analysis*; *Mobile and wireless security*; *Wireless local area networks*; • **Hardware** → *Analog, mixed-signal and radio frequency test*; • **Applied computing** → Forecasting.

KEYWORDS

Internet of Things, Device Enumeration, Vulnerability Assessment, Fuzzing

ACM Reference Format:
John Mikulskis, Johannes K Becker, Stefan Gvozdenovic, and David Starobinski. 2019. Poster: Snout - An Extensible IoT Pen-Testing Tool. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3319535.3363248

1 INTRODUCTION

Traditional network administration tools are fundamentally limited when it comes to Internet of Things (IoT) devices, as they typically operate on the TCP/IP network stack. As such, visibility into wireless devices that communicate using non-IP IoT protocols is limited to information provided by gateways, such as IoT smart hubs. Such gateways create bridges between the respective wireless networks (e.g., Zigbee) and the IP network.

Yet, from both security and asset management perspectives, the ability to enumerate and further analyze the security of installed IoT devices without having to rely on gateways is highly desirable. Possible applications include detecting mismatches between the device inventory reported by a gateway and the actual inventory observed from IoT wireless traffic (e.g., orphaned devices, forgotten legacy devices or even malicious rogue devices), and gathering intelligence on vulnerabilities of certain devices deployed in the organization (e.g., unpatched devices).

We introduce the *SDR-Based Network Observation Utility Toolkit (Snout)* to address this limitation. Snout leverages Software-Defined Radio (SDR) to passively sniff, and interact with various common IoT protocols. Our contributions in this context are as follows:

- We present an open-source IoT pen-testing tool capable of communicating with a variety of non-IP based wireless devices. We show that it can be used both interactively and through automated tasks.
- We describe the open-source software architecture that enables Snout.
- We demonstrate device enumeration capabilities of Snout for two major wireless IoT protocols (Bluetooth LE and Zigbee).
- We highlight passive and active detection of a recent Zigbee vulnerability.
- We outline future direction of development and research envisioned for this tool.

2 TOOL DESCRIPTION

Snout can be installed as a Python 3 package or as a stand-alone Docker container for easy deployment. It leverages a number of well-established open-source projects for SDR and network stack management as a foundation (see Figure 1) and interoperates with the PyBOMBS [7] package manager for GNU Radio software package management. We demonstrate an initial feature set on our website [2].

Snout is built as an abstraction layer above the low-level signal transcoding processes required for SDR-based communication. In order to provide interoperability with existing tools and facilitate

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11.
https://doi.org/10.1145/3319535.3363248

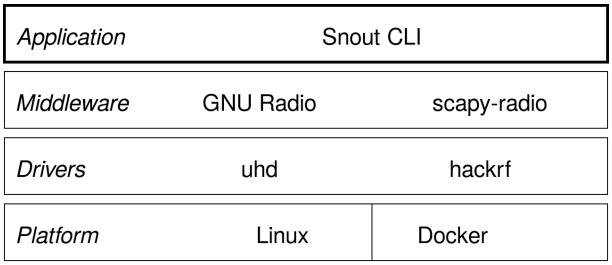

Figure 1: The Snout application leverages an entirely open-source software stack.

advanced packet handling, Snout leverages established software packages:

GNU Radio flowgraphs [6], such as IEEE 802.11 and 802.15.4 transceivers [3, 5] and RF metadata parsers like RFTap [18], which can be controlled directly from within Snout.

scapy-radio, which adds GNU Radio compatibility to the packet manipulation library scapy [1, 19] and can be used as an abstraction layer for packet transcoding.

Special-purpose SDR software, such as Xianjun Jiao's BTLE toolkit [10], which can provide general-purpose controllers and input/output interfaces to low-level processes.

2.1 Software Architecture

Snout is built with extensibility towards the large range of available IoT protocols in mind. Its command-line interface (CLI) dynamically assembles protocol and use case-specific information from the user and the underlying data models. It then orchestrates low-level software that interfaces with the SDR hardware. By design, Snout can interact with any type of child process. This design facilitates extension and integration of other existing or custom-built modules without imposing strict API requirements.

2.2 Usage and Functionality

Snout provides a flexible and interactive framework for transmitting and receiving packets across different wireless protocols, making it simple to start a scan or a transmission through its adaptable command line. The CLI follows the syntax:
`snout [OPTIONS] PROTO COMMAND [ARGS]...`, where

OPTIONS are optional program modifiers such as output format, program verbosity, or Wireshark result export.

PROTO is the wireless protocol (e.g., `zigbee`, `btle`, etc.).

COMMAND is the command to be performed, such as `scan` or `transmit`.

ARGS represent further command-specific arguments, such as timeouts or input files.

If a command requires specific arguments the user did not provide, the CLI dynamically prompts the user for all necessary inputs until these arguments are resolved. To facilitate command replay, Snout prints out a fully parameterized command at the end of each execution, which can be copied or reused in automated experimental setups or shell scripts. The `--help` flag provides further information on how to use the Snout CLI.

Snout's functionality is prioritized based on the most prevalent use cases of network mapping tools, such as Nmap. Its current functionality can be broken down as follows:

Device Enumeration: Depending on the protocol, Snout can passively monitor wireless communication and enumerate devices, or actively query devices for information.

Vulnerability Assessment: Different vulnerabilities can be detected by listening to ongoing communication (passively) or by triggering vulnerable processes (actively). Snout can also find specific vendor, OS, and protocol versions.

Advanced Packet Replay: Snout can replay received packets as-is or with specific modifications, such as dynamic sequence number increments or other packet modifications, making it a useful tool for replay vulnerability detection.

Packet Fuzzing: Snout allows the user to configure smart fuzzing functionality on *both* the preamble and the body of packets, enabling a large range of use cases around wireless communication physical layer fuzzing.

3 RESULTS

Currently, Snout's most advanced capabilities relate to the Zigbee and BTLE wireless IoT protocols [2], which we highlight below.

3.0.1 *Zigbee.*

- *Device Enumeration*: Using passive scanning, Snout can enumerate devices by sniffing traffic on any of the Zigbee protocol's 15 RF channels in the 2.4 GHz band (11-26).
- *Vulnerability assessment*: Through active scanning, Snout can assess the vulnerability of Zigbee devices to certain exploits, such as the recent Zigbee Light Link (ZLL) Commissioning exploit which allows an attacker to take over ZLL enabled smart light devices [4, 14]. While entirely passive detection is possible, it requires waiting for ZLL commissioning to happen in the wild, which may not produce immediate insights. We stress that Snout detects the vulnerability without running the exploit.
- *Command Replay*: Snout can replay any Zigbee packet sequence from either a PCAP file or from a live scan, with transmit times that are true to the packets' original timestamps. This feature can be used to test whether certain devices are vulnerable to replay attacks.
- *Fuzzing*: Snout's transmission mode has the ability to fuzz the preamble and higher layers of Zigbee packets. This feature is useful for security research, i.e., fingerprinting devices based on their response to different preambles [9].

3.0.2 *Bluetooth Low Energy (BTLE).*

- *Passive and Continuous Device Enumeration*: Using its BTLE scanning mode, Snout can track devices which broadcast BTLE packets, collect MAC addresses, and analyze traffic information (such as message frequency and uptime).
- *Device Analysis*: Snout can further analyze messages for information about device vendor, OS, model, and other device details. For Apple devices, Snout implements the majority of OS and activity intelligence gathering using recent techniques developed by Martin et al. [13], based on reversing enginering Apple's Continuity protocol.

4 PRIOR WORK

Nmap [12] and SeeSec [15] scan for IP-based networks vulnerabilities, with limited access to OSI Layer 2. Nmap and Snout share features such as device discovery as well as detection of protocol, device firmware, and vendor version. Both tools also have the ability to generate arbitrary traffic to a device and perform response analysis. However, Nmap only applies to wired Ethernet and wireless 802.11-based IoT devices, excluding lower complexity radio protocols (e.g. Zigbee, ZWave, M-Bus, Bluetooth etc.) which connect IoT endpoints/devices such as sensors and actuators. Snout, on the other hand, is built on a SDR stack that fully includes the physical layer (Layer 1) and the MAC layer (Layer 2), while still providing access higher Layers. This allows it to fingerprint device vendors or protocol versions (e.g., CVE-2016-5058) through lower level attributes such as signal quality or sampling frequency offset.

The proprietary RadioInspector software [17] can monitor radio spectrum and detect devices with multiple communication standards such as IEEE 802.15.4 (Zigbee, ISA100.11a, WirelessHART, MiWi) or Bluetooth. It runs on different hardware platforms including software defined radios or spectrum analyzers. The main application of RadioInspector is in Technical Surveillance CounterMeasure (TSCM), including searching for clandestine radio frequency sources, such as spy audio or video bugs. Snout shares similarities with RadioInspector with respect to Zigbee and Bluetooth device discovery, but is built on a fully open-source technology stack.

RFDump [11] is a software architecture for monitoring multi-technology wireless networks. Similar to RadioInspector, RFDump supports real-time classification of multiple wireless technologies on the USRP platform. RFDump is an early approach to the field of device identification that predates the rise of IoT and does not seem to have been further developed or maintained.

Z3Sec [14] is an open-source penetration testing framework that focuses on Zigbee-certified devices that implement either the Zigbee Light Link (ZLL) or the Zigbee 3.0 standard. Although Z3Sec is primarily for using these exploits, we have ported over the code necessary to scan for ZLL devices from this tool into Snout.

Universal Radio Hacker (URH) [16] is a tool that mainly investigates unknown radio protocols. It shares features with Snout such as device discovery, version detection and fuzzing and is modular and extensible to various wireless technologies. However, its primary focus lies in the identification and demodulation of initially unknown signals, while Snout focuses on network asset management and security assessment of devices running known IoT protocols.

TumbleRF [20] is a fuzzing framework for RF and physical layer protocol analysis. It leverages the fact that chipsets behave differently when subjected to malformed preamble and sync word of Zigbee packets, which enables fingerprinting or tracking of certain device types. Zigdiggity [8] is a Zigbee hacking toolkit intended for Raspberry Pi and RaspBee radio. Its features include passive device discovery, network discovery, finding and unlocking locks and other attacks. In contrast to these tools, Snout is designed on a modular SDR-based hardware and software platform.

5 SUMMARY AND FUTURE DIRECTION

We presented Snout, a versatile open-source toolkit for penetration testing and non-IP IoT network mapping. Snout provides information gathering and device enumeration utilities for a variety of IoT protocols, including Zigbee, Bluetooth, and Wi-Fi. Snout also improves accessibility of SDR-based software for wireless connected asset management and pen-testing. We plan to augment Snout with a graphical user interface and to integrate additional IoT protocols, including Z-Wave and LoRa. Finally, we plan to expand Snout's transmission capabilities, including active scanning and fuzzing, to other IoT protocols beside Zigbee. An initial preview release of the Snout CLI will soon be available on its website [2].

6 ACKNOWLEDGMENTS

The authors thank Byoungsul Lee, Christina Chimienti, Mohammed Uddin, Seung Hee Lee, and Spencer Liu for their 2018-2019 Senior Design Team project "IoT Nmap" at Boston University, which served as an early prototype of Snout. This research was supported in part by NSF grant CNS-1409053.

REFERENCES

[1] Bastille Research. 2016. scapy-radio. https://github.com/BastilleResearch/scapy-radio
[2] Johannes K Becker and John Mikulskis. 2019. Snout.tools. https://snout.tools/
[3] Bastian Bloessl, Christoph Leitner, Falko Dressler, and Christoph Sommer. 2013. A GNU Radio-based IEEE 802.15.4 Testbed. In *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*. Cottbus, Germany, 37–40.
[4] Bastian Bloessl, Michele Segata, Christoph Sommer, and Falko Dressler. 2017. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. *IEEE Symposium on Security and Privacy (SP)* (May 2017), 195–212. https://doi.org/10.1109/SP.2017.14
[5] Bastian Bloessl, Michele Segata, Christoph Sommer, and Falko Dressler. 2018. Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype. *IEEE Transactions on Mobile Computing* 17, 5 (May 2018), 1162–1175. https://doi.org/10.1109/TMC.2017.2751474
[6] Eric Blossom. 2004. GNU Radio: Tools for Exploring the Radio Frequency Spectrum. *Linux J.* 2004, 122 (June 2004), 4–. http://dl.acm.org/citation.cfm?id=993247.993251
[7] Martin Braun and Seth Hitefield. 2019. PyBOMBS. https://github.com/gnuradio/pybombs/
[8] Bishop Fox. 2019. ZigDiggity. https://github.com/BishopFox/zigdiggity
[9] Ira Ray Jenkins, Rebecca Shapiro, Sergey Bratus, Ryan Speers, and Travis Goodspeed. 2014. *Fingerprinting IEEE 802.15.4 Devices with Commodity Radios.* Technical Report TR2014-746. Dartmouth College, Computer Science, Hanover, NH. http://www.cs.dartmouth.edu/reports/TR2014-746-rev2.pdf
[10] Xianjun Jiao. 2019. BTLE. https://github.com/JiaoXianjun/BTLE
[11] Kaushik Lakshminarayanan, Samir Sapra, Srinivasan Seshan, and Peter Steenkiste. 2009. RFDump: an architecture for monitoring the wireless ether. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. ACM, 253–264.
[12] Gordon Lyon. 2019. Nmap: the Network Mapper. https://nmap.org/
[13] Jeremy Martin, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske, Lucas Foppe, Travis Mayberry, Erik C. Rye, Brandon Sipes, and Sam Teplov. 2019. Handoff All Your Privacy: A Review of Apple's Bluetooth Low Energy Continuity Protocol.
[14] Philipp Morgner, Stephan Mattejat, Zinaida Benenson, Christian MÃijller, and Frederik Armknecht. 2017. Insecure to the touch. *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks - WiSec 17* (Jul 2017). https://doi.org/10.1145/3098243.3098254
[15] Ruth Ogunnaike. 2017. SeeSec - IoT Vulnerablity Scanner . https://github.com/ruthogunnnaike/SeeSec---IoT-Vulnerablity-Scanner
[16] Johannes Pohl and Andreas Noack. 2018. Universal Radio Hacker: A Suite for Analyzing and Attacking Stateful Wireless Protocols. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*. USENIX Association, Baltimore, MD. https://www.usenix.org/conference/woot18/presentation/pohl
[17] RadioInspector. 2019. New Generation of Professional TSCM software. https://www.radioinspector.com/
[18] RFtap: rftap.protocol@gmail.com. 2016. RFtap. https://rftap.github.io
[19] SecDev. 2006. scapy. https://github.com/secdev/scapy
[20] River Loop Security. 2018. TumbleRF. https://github.com/riverloopsec/tumblerf