Hackintro Thesis Presentation

RePort: Automatically Mapping the Attack Surface of IoT Systems

Dimitrios Stefanos Porichis

Let’s meet

Hey, I am Jim!

MSc in Advanced Computer Science

TBD

Cyber Warfare Private

Hellenic Army

BSc in Informatics and Telecommunications

National and Kapodistrian University of Athens

Let’s meet

Thesis w/ Prof Thanassis Avgerinos

RePort 2025

Thesis submitted

Collab? 2024

Knocks the door of A40 office

HackIntro Class 2024

3rd Year Jim got his mind blown up

Operating Systems, System Programming 2023

Jim loves low level stuff <3

Presentation Overview

Duration: 25 minutes

Introduction

Who is Jim and how did he end up here?

The Security State of IoT

Understanding the problems that need to be addressed

The Research Landscape

What are the tactics used to tackle similar issues

RePort

A Modular Automatic Attack Surface Mapper

IoT Security

-I like this template-

3 Types of IoT Firmware

Type I

Bare Metal, *no operating system*

Type II

Real-time Operating System, *running real-time applications with limited OS features.*

We don’t know much

Type III

Linux based firmware, *usually resource rich devices.*

The main thing

Key reasons for poor security in IoT

 # 1

Exploitation plausible because of their all-time connection to the Internet.

 # 2

Performance vs Security Trade-offs

 # 3

Lack of Security Updates

— A taxonomy of IoT firmware security and principal firmware analysis techniques, Ibrahim Nadir, Haroon Mahmood, Ghalib Asadullah

Flawed business partnership

OEMs with flaws.

Inexperienced developers

Hired by corporates to somehow reduce the time to market.

Sporadic firmware updates

Updates required to patch security issues.

Predetermined code base

Use of existing old (and free) code base and libraries to reduce code footprint and time to market.

Performance vs. Security

Consumers and Stakeholders value performance more than safety.

Features specific updates

New features with few lines of code which may possibly add new bugs.

Backdoor installation

Certain vendors themselves introduce backdoors into the firmware.
(refrain from name dropping)

End-user security awareness

Little to no knowledge about security and/or privacy.

No firmware source

Vendors do not upload or provide their product's firmware images, their code is also closed source.

Non-standard protocols

Companies implement non-standard protocols.

Lack of regulatory agencies

There is no regulatory body to enforce recommendations on IoT device vendors.

— A taxonomy of IoT firmware security and principal firmware analysis techniques,
Ibrahim Nadir, Haroon Mahmood, Ghalib Asadullah

Scary™ Numbers

48%

IoT Vendors DO NOT test for security

2017 Study on Mobile and IoT Application Security

50%

Of Online IoT devices are vulnerable to medium-high ranked attacks

2020 Paloalto's Unit 42 IoT Threat Report

~18B

Estimate of IoT devices online right now

State of IoT Summer 2024, IoT Analytics

The Research Landscape

How do people secure systems on other industries

The Research Landscape (Other Industries)

Mapping the Attack Surface

Attack surface mapping is the process of **identifying** and **examining** all the possible points where an unauthorized user can try to enter or extract data from an environment.

How?

- **Code Analysis**
- **CVE and CWE Lookup**
- **SBOM**

The Research Landscape (Other Industries)

Mapping the Attack Surface

Atta
of **id**
whe
data

How?

IoT is build different (less secure that is)

Predetermined code base

Use of existing old (and free) code base and libraries to reduce code footprint and time to market.

No firmware source

Vendors do not upload or provide their product's firmware images, their code is also closed source.

Result (to the best of our knowledge™)

No Automatic Attack Surface Mapper designed for IoT firmware

Open Research Areas

What tools are available for us to use

Static binary analysis

Infer execution behavior by using constrain models.

Dynamic analysis via Emulation

Mimic the execution environment, create custom kernels and boot up the firmware.

SBOM Tools

Identify binaries using hashing and fuzzy matching.

Introducing: RePort

A Modular Automatic Attack Surface Mapper for IoT

What could the attack surface of an IoT device look like?

Services Running	Open Ports
Embedded Web Server (httpd)	80, 443
Firmware Update Service	1234
Dishwasher Stuff idk	-
Debug Service (SSH)	22
backdoor_v3.12.31.123	4242
Device Discovery (mDNS)	5353

Components

EmulationEngine

Abstract class for connecting emulation tools to work with RePort

MappingEngines

Abstract class for connecting network tools to work with RePort

PortActivity

A class with methods encapsulating the port operations logic, deriving port activity from traced Systemcalls

CVELookUp

Abstract class for performing SBOM-based scanning for automated detection of outdated or vulnerable components

PortActivity

Where do open ports come from?

Fork()

Creates a child process, passing down a copy of *ALL* file descriptors

Exit()

When a process exits, all file descriptors are cleaned up by the OS

Setsockopt()

Changes the behavior of sockets according to passed settings

Socket()

Creates a new socket file descriptor

Bind()

Binds the socket in order to receive external traffic

Close()

Closes the file descriptor, unbinding it from external usage

PortActivity

Where do processes come from?

Fork()

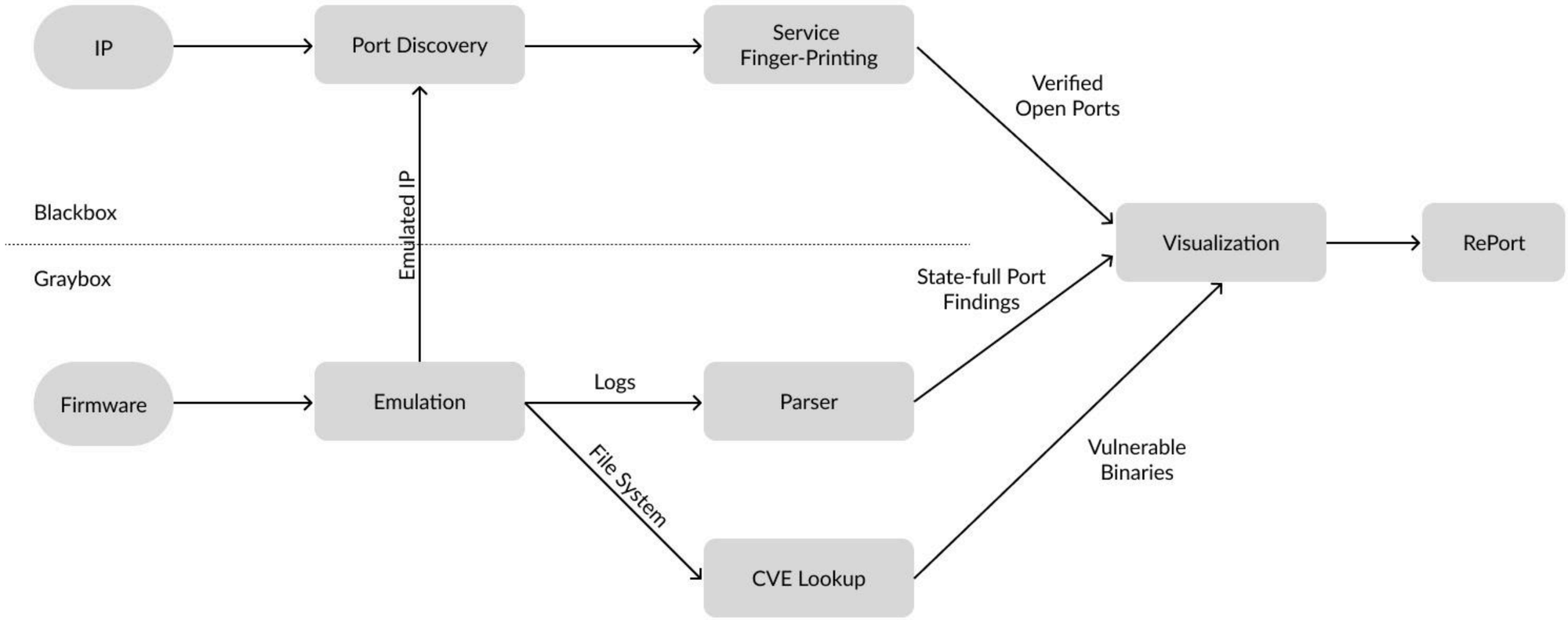
Creates a child process, with its own PID

Execv()

Overrides the process image, executing new program with given arguments

Under the Hood

RePort Operation Graph

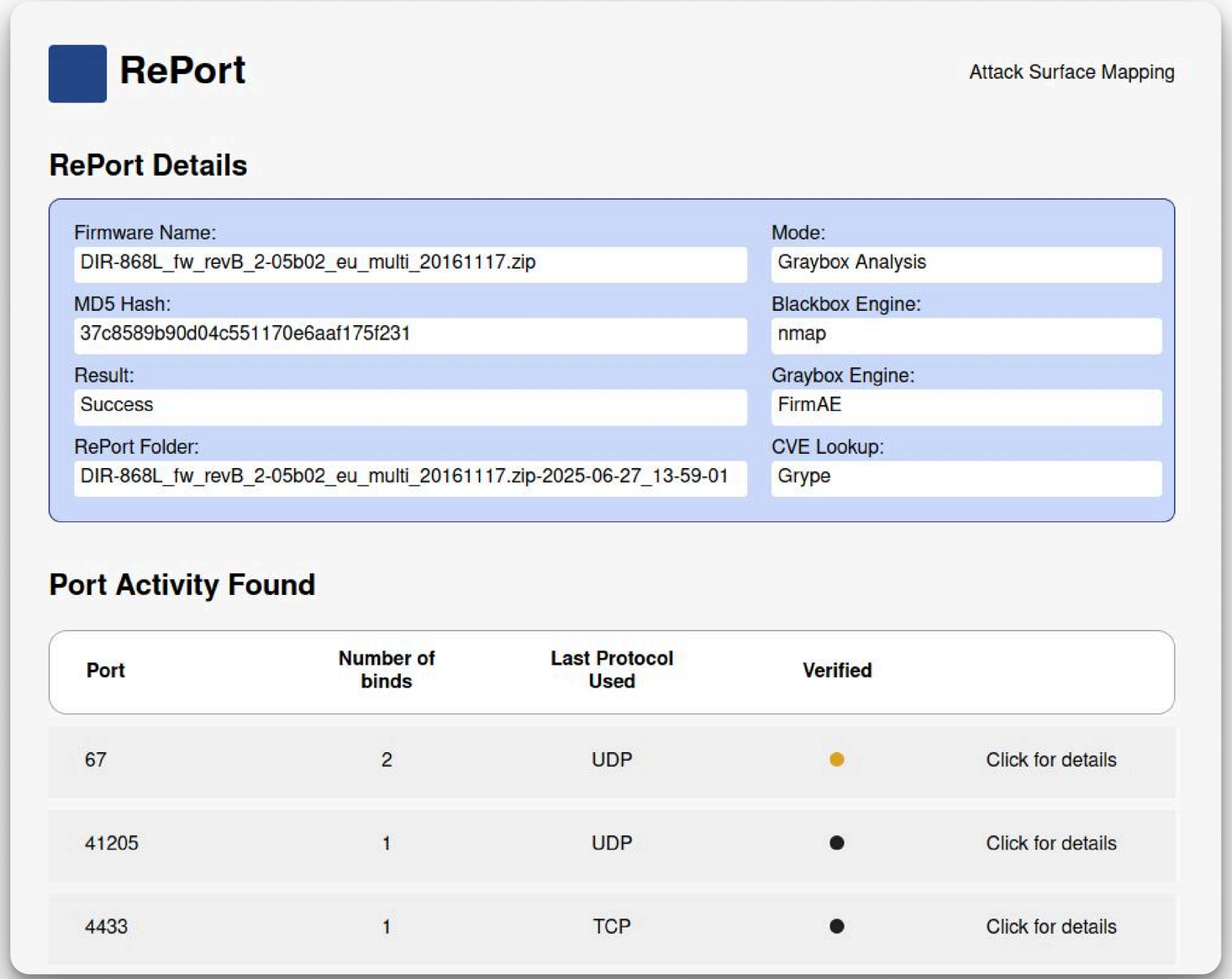


Mapping with RePort

A Small Demo

RePort

Port Activity



RePort

Binary Nutrition Labels

Outward facing binaries found

Binary	Ports binded	Ports accessed	Instances spawned	CVEs Found	
arpmonitor	2	2	1	0	Click for details

Overview of arpmonitor

Path	/home/porichis/dit-thesis/reports/DIR-868L_fw_revB_2-05b02_eu_multi_20161117.zip-2025-06-27_13-59-01/fs/usr/sbin/arpmonitor
Owned Ports	(52851, 'IPv4', 'UDP') (33542, 'IPv4', 'UDP')
Accessible Ports	(52851, 'IPv4', 'UDP') (33542, 'IPv4', 'UDP')
PIDs	772
Libraries	('libc.so.0', 'Primary')

udhcpd	1	1	2	0	Click for details
ntpclient	3	3	3	0	Click for details

RePort

CVE Lookup Findings

CVEs Report

Binary	CVE Count	Ports Reachable from that binary	
openssl	52	0	Click for details
busybox	17	0	Click for details

Overview of busybox

Path: /home/porichis/dit-thesis/reports/DIR-868L_fw_revB_2-05b02_eu_multi_20161117.zip-2025-06-27_13-59-01/fs/bin/busybox

CVE ID	Rating	Description
CVE-2018-1000517	Critical	BusyBox project BusyBox wget version prior to commit 8e2174e9bd836e53c8b9c6e00d1bc6e2a718686e contains a Buffer Overflow vulnerability in Busybox wget that can result in heap buffer overflow. This attack appear to be exploitable via network connectivity. This vulnerability appears to have been fixed in after commit 8e2174e9bd836e53c8b9c6e00d1bc6e2a718686e.
CVE-2016-2148	Critical	Heap-based buffer overflow in the DHCP client (udhcpc) in BusyBox before 1.25.0 allows remote attackers to have unspecified impact via vectors involving OPTION_6RD parsing.
CVE-2022-28391	High	BusyBox through 1.35.0 allows remote attackers to execute arbitrary code if netstat is used to print a DNS PTR record's value to a VT compatible terminal. Alternatively, the attacker could choose to change the terminal's colors.
		Directory traversal vulnerability in the BusyBox implementation of tar

RePort

Evaluation

RePort was run on firmware images obtained from routers, IP cameras, and network extenders manufactured by DLink, Asus, Trendnet, and Netgear.

Firmware Name	# Ports Used	# Instances	# Critical Binaries
DIR-868L_fw_revB_2-05b02_eu_multi_20161117	21	33	8
DIR868LWB1_FW200KR-K05	19	28	8
R7000-V1.0.5.64_1.1.88	15	20	5
DCS-8200LH_REVA_FIRMWARE_1.02.03	9	17	3
FW_RT_AC5300_300438432738	6	6	5
TEW-657BRM_1.00.1	3	3	3
FW_TV-IP110WN_1.2.2.65	2	2	1
RE450_V1_171215	2	2	1
Total	**77**	**111**	**34**

Firmware Name	# CVEs	# Files with CVEs
DIR-868L_fw_revB_2-05b02_eu_multi_20161117	69	2
DIR868LWB1_FW200KR-K05	17	1
R7000-V1.0.5.64_1.1.88	80	2
FW_RT_AC5300_300438432738	60	2
TEW-657BRM_1.00.1	17	1
FW_TV-IP110WN_1.2.2.65	17	1
DCS-8200LH_REVA_FIRMWARE_1.02.03	86	2
RE450_V1_171215	23	1
Total	**369**	**12**

RePort

Future Work

Emulation Engine Support

Utilize more cool projects to work with RePort, like Greenhouse

Network Mapping

Extend verification methods to support UDP and extensive fingerprinting

CVELookUp Addition

Can we find a more powerful tool than Grype

Deeper Port Analysis

Extending the port tracking fidelity by extracting signals and adding Unix Sockets support

Replayability

Implement custom kernels to tackle non deterministic behavior, like bind calls using the port 0

Thank you all.

Jim