Large-Scale Firmware Security Analysis for IoT Devices

Akila Abeykoon[1], Islam Obaidat[2], Ahmad Patooghy[2], Shyam Aravamudhan[3]
[1]Department of Electrical and Computer Engineering, North Carolina State University
[2]Department of Computer Systems Technology, North Carolina A&T State University
[3]Joint School of Nanosceince & Nanoengineering, North Carolina A&T State University

Introduction

- IoT devices have exponentially grown, reaching billions of interconnected units.
- IoT vulnerabilities have led to critical security incidents, e.g., Mirai botnet attack in 2016 compromised millions of devices and caused extensive network disruptions.
- Manual vulnerability detection methods are impractical due to the sheer scale and complexity of firmware.
- Automated, scalable solutions are urgently needed for timely and effective security assessments.

Static vs. Dynamic Analysis:
- Dynamic: Involved executing the firmware in an emulated or controlled environment to monitor real-time behavior
- Static: Inspection of firmware binaries without executing them

Current Work Highlights

Pipeline Automation Enhancements
- Streamlined firmware unpacking, architecture detection, and emulation using Bash scripting and database integration.
- Modified getArch.sh to automatically interface with PostgreSQL—eliminating repetitive manual input.

Scalable Analysis Infrastructure
- Emulated dozens of firmware images across ARM/MIPS using Firmadyne with updated QEMU and kernel versions.
- Built a logging system that records metadata like firmware name, architecture, IP address, and emulation status into CSV format.

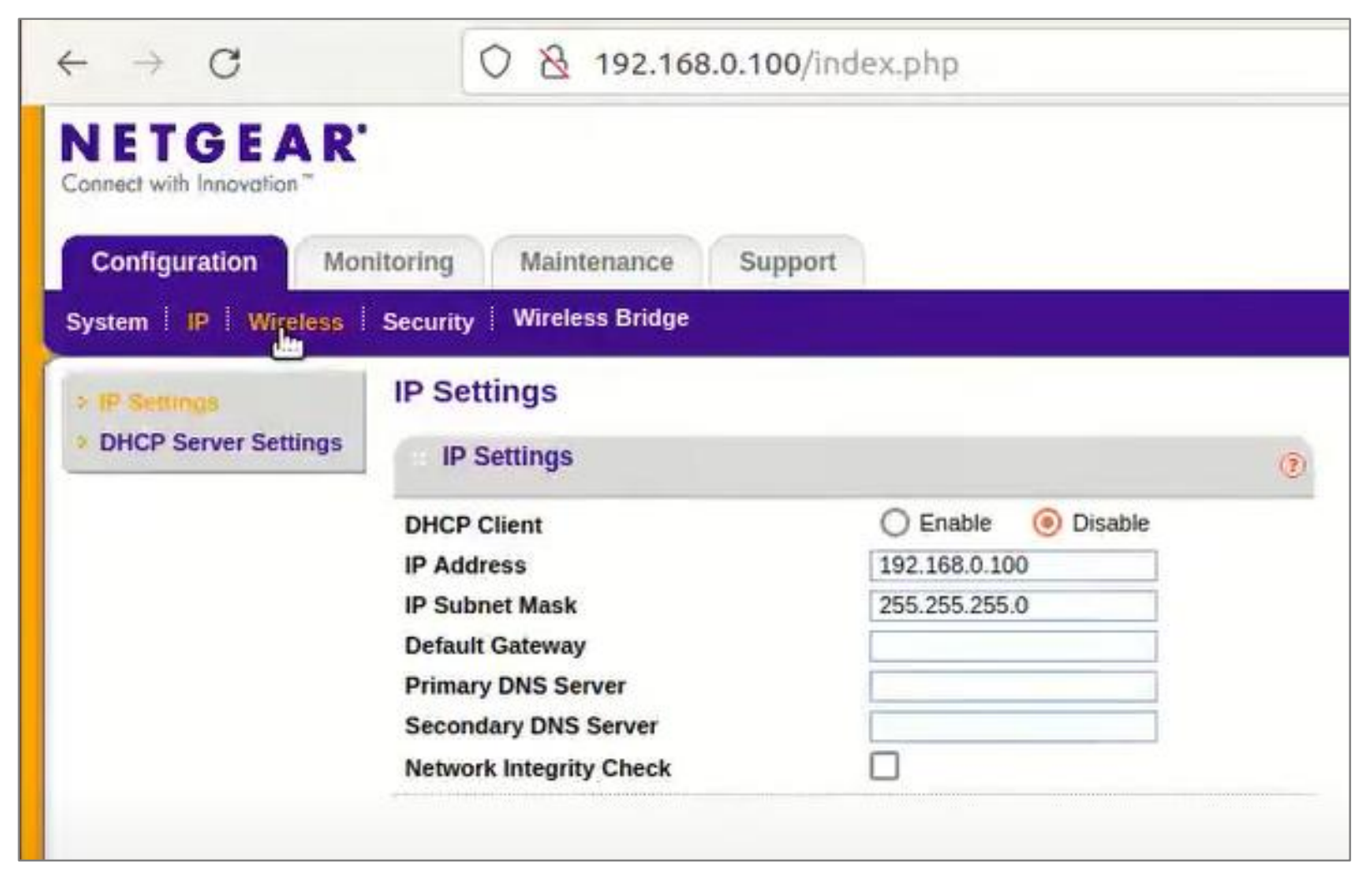
Static Analysis Integration with Karonte
- Successfully integrated Karonte into the pipeline to begin static taint analysis on emulated firmware images.
- Collected early-stage results on inter-binary data flows and vulnerability detection using Binary Dependency Graphs.

Firmadyne (Dynamic Analysis)

Overview:
Firmadyne performs automated dynamic vulnerability analysis through full system emulation of Linux-based IoT firmware.
- Analyzed over 23,035 firmware images from 42 vendors.
- Successfully extracted 9,486 firmware images and 887 vulnerable firmware images across 89 distinct products.
- Discovered 14 previously unknown vulnerabilities.

Methodology:
- Firmware Extraction: Automated using Binwalk with recursive unpacking.
- Architecture Classification: Scripted detection with PostgreSQL integration.
- Firmware Emulation: Updated QEMU and kernels for broader compatibility.
- Network Discovery: Auto-configured via improved inferNetwork.sh script.
- Metadata Logging: Structured CSV output for scalable analysis.

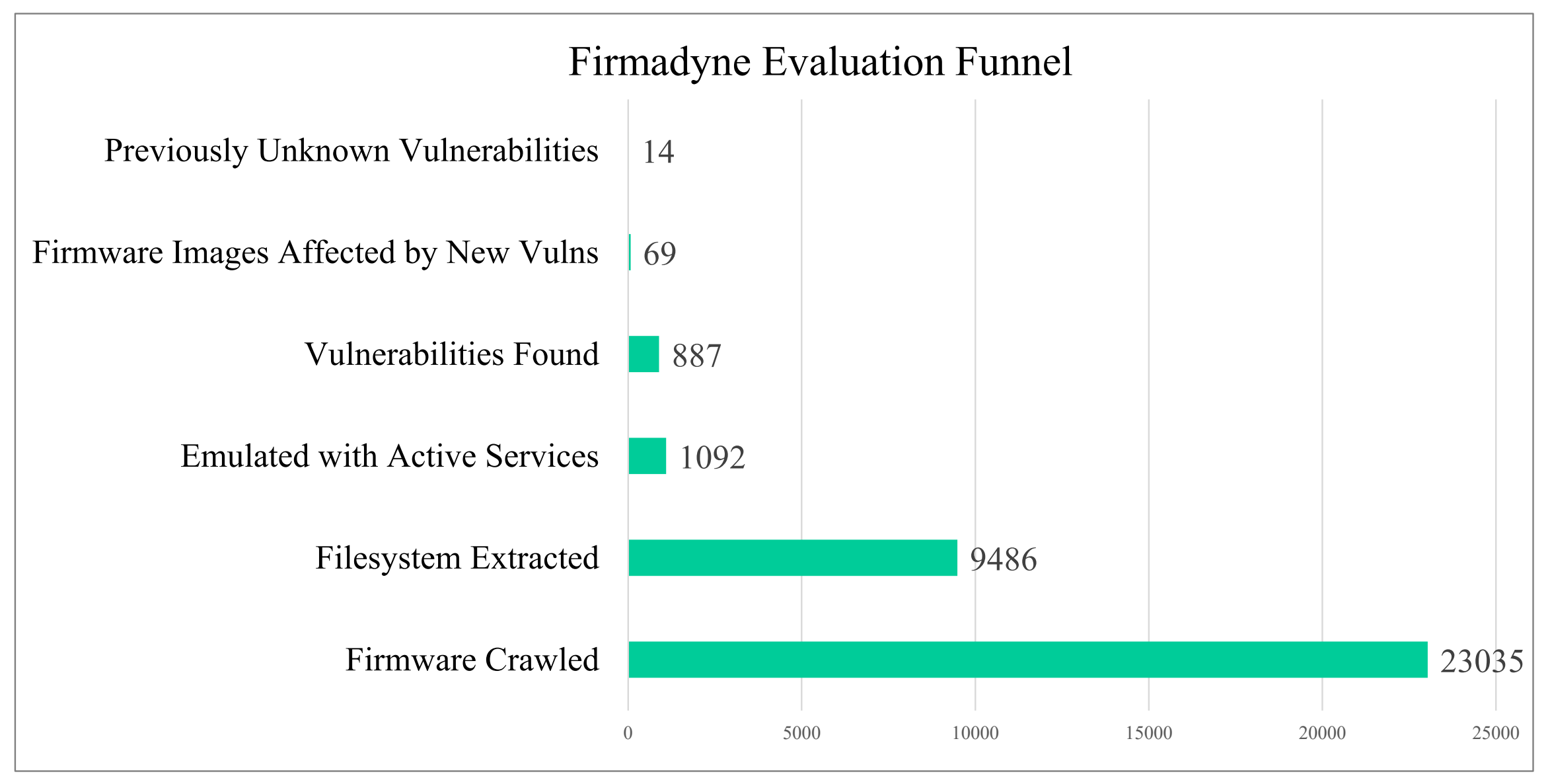
Key Innovations:
- Emulates full firmware environments using QEMU (kernel + user-space).
- Detects runtime issues like default credentials and exposed services.
- Supports key IoT architectures (ARM, MIPS) out of the box.

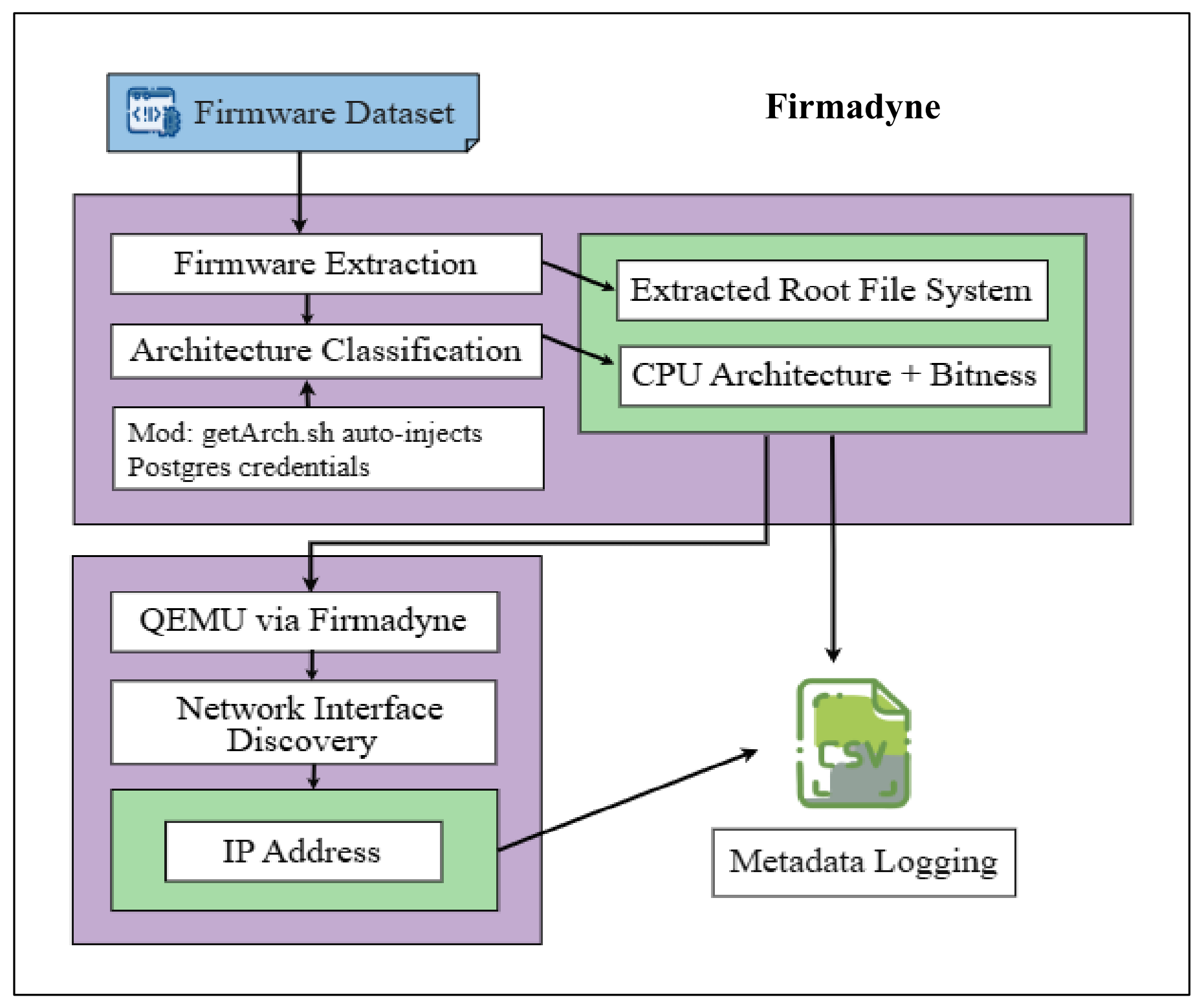


IP Settings

DHCP Client	Enable [✓ Disable]
IP Address	192.168.0.100
IP Subnet Mask	255.255.255.0
Default Gateway	
Primary DNS Server	
Secondary DNS Server	
Network Integrity Check	☐

Emulated Netgear Device



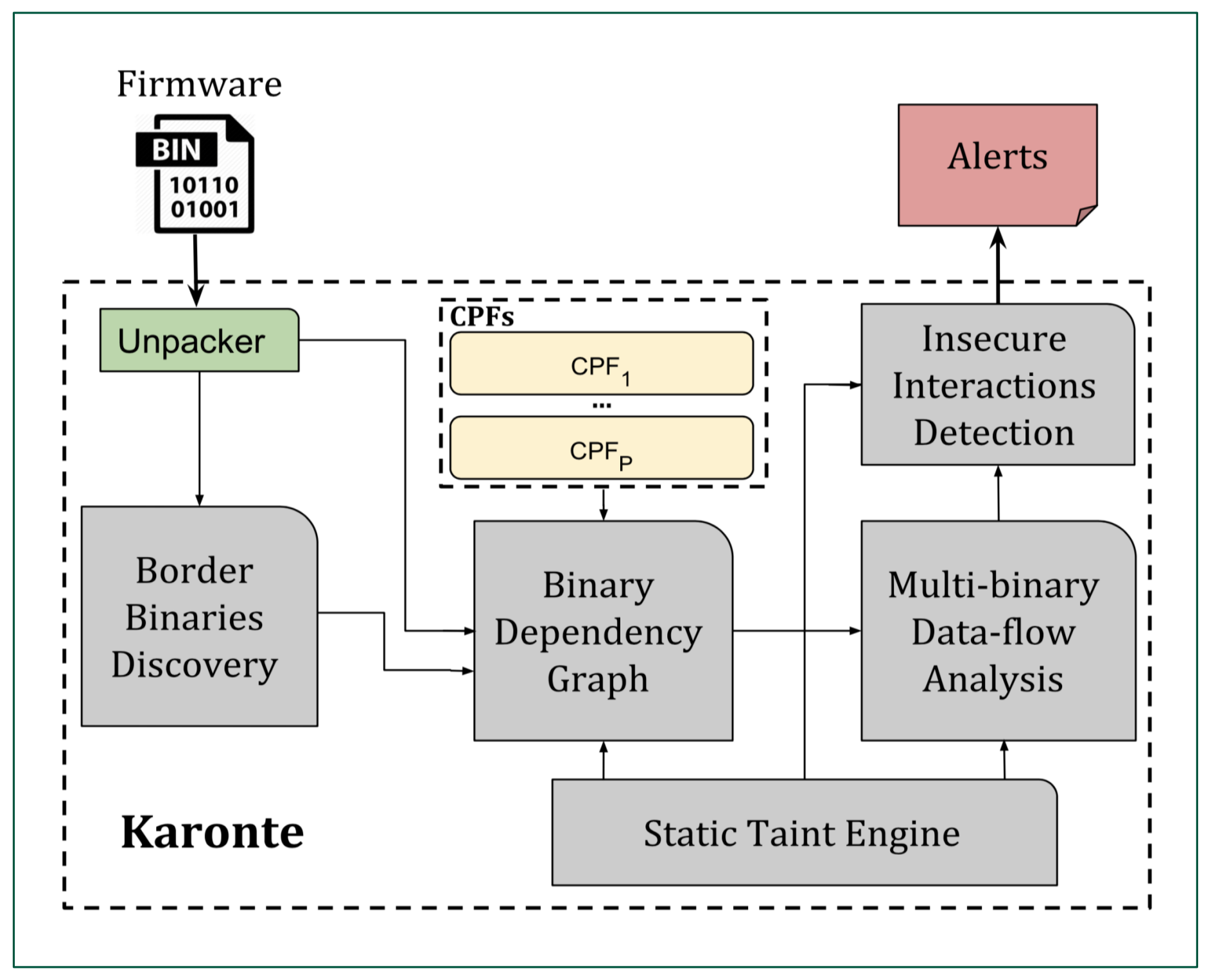

Firmware Dataset
Firmadyne
Firmware Extraction
Extracted Root File System
Architecture Classification
CPU Architecture + Bitness
Mod: getArch.sh auto-injects Postgres credentials
QEMU via Firmadyne
Network Interface Discovery
IP Address
Metadata Logging

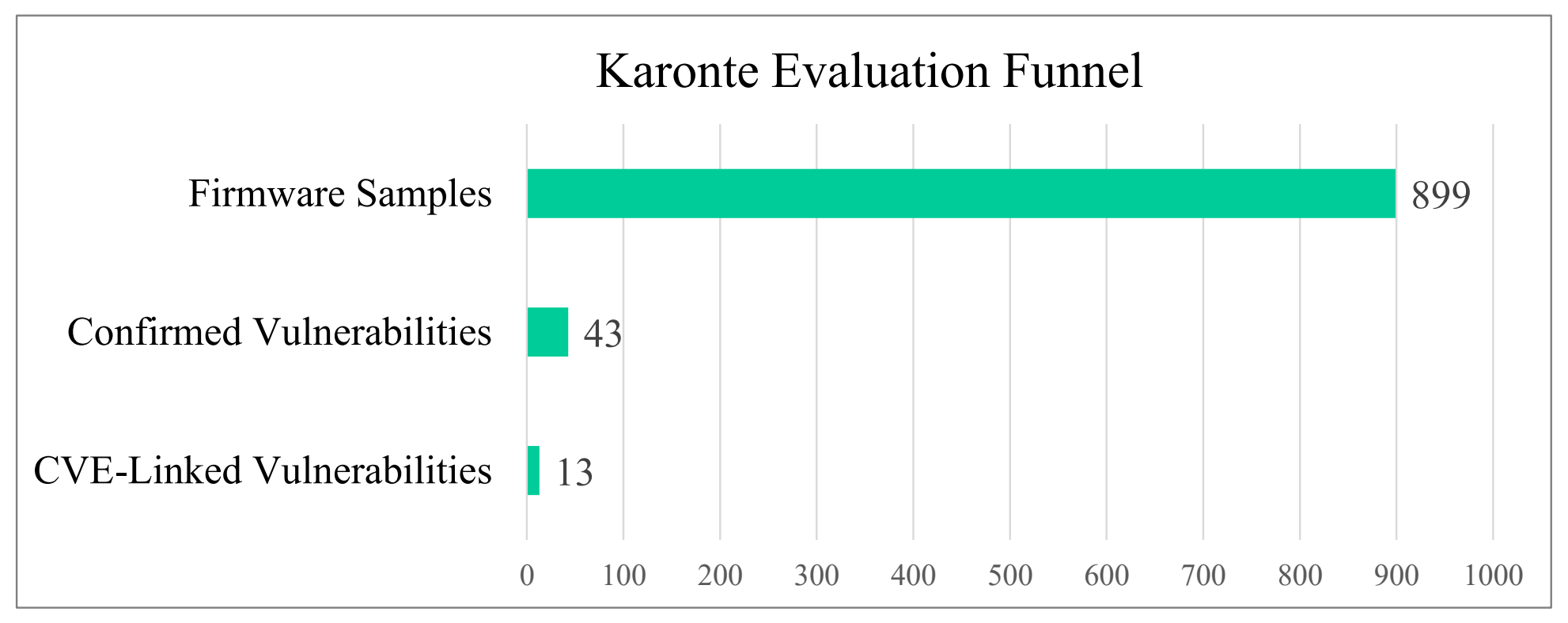
Karonte (Static Analysis)

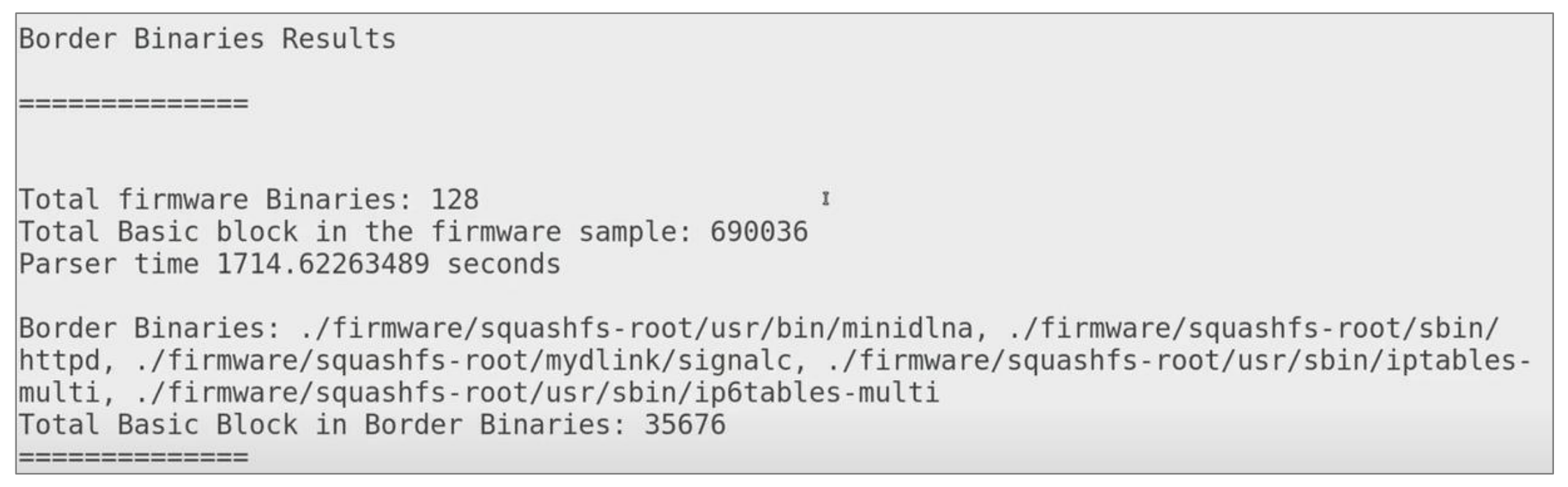
Karonte Overview:
Karonte performs static analysis by tracking data flows across multiple interacting binaries within IoT firmware to accurately identify vulnerabilities.
- Evaluated on 53 firmware samples discovering 46 zero-day bugs.
- Successfully scaled testing to 899 firmware samples, demonstrating efficient scalability.

Methodology:
- Detect network-facing binaries.
- Create Binary Dependency Graph.
- Apply multi-binary taint analysis to track insecure data flows.
- Precisely identify exploitable vulnerabilities.

Key Innovations:
- Detects vulnerabilities across multiple binaries for deeper analysis.
- Greatly reduces false positives (from hundreds to ~2 per binary).
- Employs Binary Dependency Graphs (BDGs) and IPC modeling for precise detection.







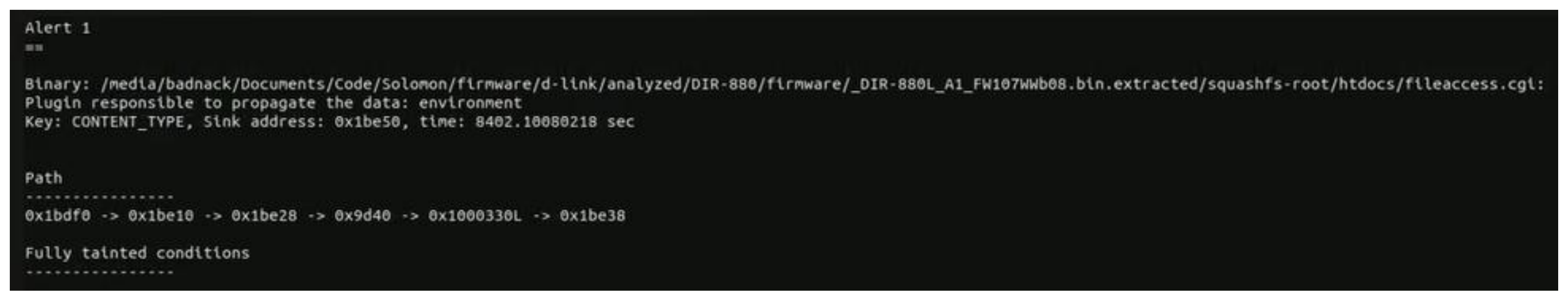
```
Border Binaries Results

==============


Total firmware Binaries: 128
Total Basic block in the firmware sample: 690036
Parser time 1714.62263489 seconds

Border Binaries: ./firmware/squashfs-root/usr/bin/minidlna, ./firmware/squashfs-root/sbin/
httpd, ./firmware/squashfs-root/mydlink/signalc, ./firmware/squashfs-root/usr/sbin/iptables-
multi, ./firmware/squashfs-root/usr/sbin/ip6tables-multi
Total Basic Block in Border Binaries: 35676
==============
```

Border Binaries Results



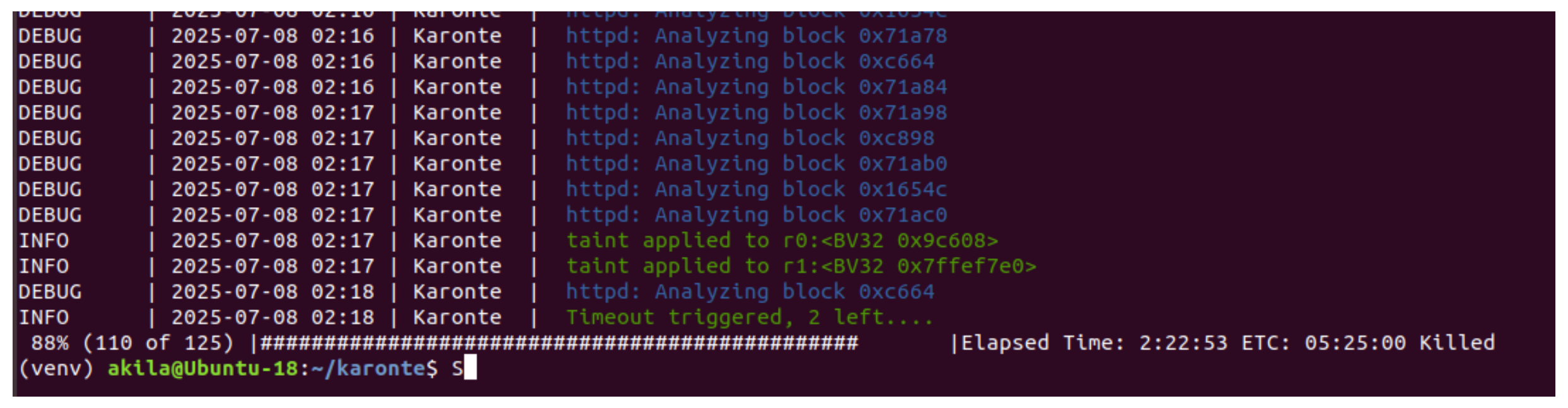
```
Alert 1
==

Binary: /media/badnack/Documents/Code/Solomon/firmware/d-link/analyzed/DIR-880/firmware/_DIR-880L_A1_FW107WWb08.bin.extracted/squashfs-root/htdocs/fileaccess.cgi:
Plugin responsible to propagate the data: environment
Key: CONTENT_TYPE, Sink address: 0x1be50, time: 8402.10080218 sec


Path
---------------
0x1bdf0 -> 0x1be10 -> 0x1be28 -> 0x9d40 -> 0x1000330L -> 0x1be38

Fully tainted conditions
---------------
```

Karonte Results



```
DEBUG      | 2025-07-08 02:16 | Karonte  |  httpd: Analyzing block 0x71a78
DEBUG      | 2025-07-08 02:16 | Karonte  |  httpd: Analyzing block 0xc664
DEBUG      | 2025-07-08 02:16 | Karonte  |  httpd: Analyzing block 0x71a84
DEBUG      | 2025-07-08 02:17 | Karonte  |  httpd: Analyzing block 0x71a98
DEBUG      | 2025-07-08 02:17 | Karonte  |  httpd: Analyzing block 0xc898
DEBUG      | 2025-07-08 02:17 | Karonte  |  httpd: Analyzing block 0x71ab0
DEBUG      | 2025-07-08 02:17 | Karonte  |  httpd: Analyzing block 0x1654c
DEBUG      | 2025-07-08 02:17 | Karonte  |  httpd: Analyzing block 0x71ac0
INFO       | 2025-07-08 02:17 | Karonte  |  taint applied to r0:<BV32 0x9c608>
INFO       | 2025-07-08 02:17 | Karonte  |  taint applied to r1:<BV32 0x7ffef7e0>
DEBUG      | 2025-07-08 02:18 | Karonte  |  httpd: Analyzing block 0xc664
INFO       | 2025-07-08 02:18 | Karonte  |  Timeout triggered, 2 left....
 88% (110 of 125) |#################################################      |Elapsed Time: 2:22:53 ETC: 05:25:00 Killed
(venv) akila@Ubuntu-18:~/karonte$ S
```

Performance Issues Encountered with Karonte

Conclusion and Path Forward

Key Accomplishments:
- Reproduced and improved Firmadyne for large-scale dynamic firmware analysis tasks.
- Automated extraction, architecture detection, emulation, and metadata logging.
- Integrated Karonte for static vulnerability analysis capabilities.

Current Work:
- Collecting structured data on Karonte for large-scale comparative studies and benchmarking.
- Scaling Karonte analysis to additional new firmware samples for broader coverage.

Skills Gained:
- Experience with QEMU, Binwalk, and advanced Linux-based emulation.
- Debugging complex real-world firmware issues (e.g., kernel mismatches, broken filesystems, driver conflicts).
- Applied taint tracking and binary dependency modeling via Karonte.
- Gained hands-on experience in practical IoT firmware security research.