DISSERTATION

BEHAVIORAL COMPLEXITY ANALYSIS OF NETWORKED SYSTEMS TO IDENTIFY MALWARE ATTACKS

Submitted by

Kyle Haefner

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2020

Doctoral Committee:

Advisor: Indrakshi Ray

Asa Ben-Hur
Joe Gersch
Stephen Hayne
Indrajit Ray

Copyright by Kyle A. Haefner 2020

All Rights Reserved

ABSTRACT

BEHAVIORAL COMPLEXITY ANALYSIS OF NETWORKED SYSTEMS TO IDENTIFY MALWARE ATTACKS

Internet of Things (IoT) environments are often composed of a diverse set of devices that span a broad range of functionality, making them a challenge to secure. This diversity of function leads to a commensurate diversity in network traffic, some devices have simple network footprints and some devices have complex network footprints. This network-complexity in a device's traffic provides a differentiator that can be used by the network to distinguish which devices are most effectively managed autonomously and which devices are not.

This study proposes an informed autonomous learning method by quantifying the complexity of a device based on historic traffic and applies this complexity metric to build a probabilistic model of the device's normal behavior using a Gaussian Mixture Model (GMM). This method results in an anomaly detection classifier with inlier probability thresholds customized to the complexity of each device without requiring labeled data.

The model efficacy is then evaluated using seven common types of real malware traffic and across four device datasets of network traffic: one residential-based, two from labs, and one consisting of commercial automation devices. The results of analysis of over 100 devices and 800 experiments show that the model leads to highly accurate representations of the devices, and a strong correlation between the measured complexity of a device and the accuracy to which its network behavior can be modeled.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The Internet of Things (IoT) term was first coined by Kevin Ashton in 1999 when describing RFID use in supply chain management [1]. This definition has ballooned to become a catch-all phrase for any device that connects to or interacts with the Internet. Examples of these devices include medical sensors that monitor health metrics, home automation devices, traffic monitoring, and scientific research sensors. In the future these devices will be designed to last for a few weeks and be disposed of, like a sensor on food packaging. Others will be embedded into infrastructure that will be around for decades, such as sensors embedded into roads. Some devices will need to run on batteries for years, with limited processing and storage capabilities, and will spend the majority of the time in sleep mode. Others will have powerful processors, a constant power source, and a high bandwidth connection to the network. This diversity in function, capability, and life-span is at the core of what makes securing these devices so challenging.

We can take advantage of this diversity of function by developing a measure of complexity for devices based on their network traffic to classify single-purpose from general-purpose devices and use this classification to guide autonomous network enforcement decisions.

1.1 Background

The exigency of solving the security issues presented by IoT devices is underscored by both the scale and the scope of the problem. Some predictions place the number of deployed devices at 50 billion devices by the end of 2020 [2]. For perspective, in 2016 the Mirai botnet attacked the Dyn network with only 600,000 compromised devices with a sustained bandwidth of up to 1.1 Tbs, taking down hundreds of websites [3]. This was, at the time, widely considered to be the largest distributed denial of services (DDoS) attack ever.

In response, governments, industry, and standards organizations mobilized to improve the intrinsic security of devices. IoT specifications were released such as Open Connectivity Foundation

(OCF), [4] which requires encryption, authentication, and authorization by default. Several security baselines were authored by governmental and industry-based bodies that give guidance and recommendations to improve the state of IoT security. Some of these baselines include the National Institute for Standards and Technology's (NIST) *Foundational Cybersecurity Activities for IoT Device Manufacturers* [5], the Consumer Technology Association's (CTA) *The C2 Consensus on IoT Device Security Baseline Capabilities* [6], and the European Union Agency for Cyber-security's (ENISA) *Baseline Security Requirements for IoT* [7] documents.

Standards and guidance can only go so far in fixing this problem. Regulation is typically slow, and too disjointed to address the millions of devices that are currently in use and vulnerable. In addition, old exploits such as Ripple20 [8] have been around for years undiscovered in the supply-chain of long-forgotten libraries. These exploits continue to provide a path for evolving malware such as Mirai malware to invade and infect devices [9].

It is clear that no amount of standardization, guidance, or regulation will fully solve this problem. There will always be devices that are exposed, unpatched, and vulnerable. The networks that host these devices represent what may be the last line of defense to block malicious traffic. However, the current network typologies and security mechanisms are inadequate in handling the number of new devices, as well as the dynamic nature and complexity of these devices.

For example, the established methods of identifying and learning IoT devices are problematic. Some examples in the literature use a supervised approach with labeled data to identify devices with high accuracy, but these methods provide little flexibility in their ability to scale and identify unknown and previously unseen devices. Others use autonomous methods that can be applied broadly to network traffic, but they tend to overgeneralize, resulting in less accuracy when detecting anomalies. We are missing a model that is both accurate and flexible, accounting for the behavior of each individual device while allowing for the varied capabilities, attack surfaces, and risk profiles across all devices on a network.

As we look toward building effective network security methods, we must realize that it is no longer sufficient to simply apply a binary and deterministic model of trust to network traffic en-

forcement decisions. Instead, our methods must allow for more nuanced approaches to controlling flows of data within the network. The networks of the future should be capable of constructing predictive models of devices, and authorized to autonomously enforce them.

1.2 Purpose of the Study

Networks are made of an extremely diverse set of devices, we need a solution that recognizes this diversity. Measuring a device's network complexity provides the differentiator required by the network to make informed decisions on a per-device basis. This work will formulate several methods to calculate the complexity of a device's network traffic, and will demonstrate that complexity is a measure of the certainty to which we can model a device. The relationship of complexity and the certainty with which a device can be modeled leads to a natural method by which to identify devices that are best protected by an autonomous network enforcement model.

To design a network that can dynamically learn from the devices, this research builds an informed autonomous learning method that does not rely on labeled data, bridging the gap between the inflexibility of supervised methods and the over-generality of traditional anomaly detection methods. The model presented in this research takes advantage of the predictability of an IoT device's network footprint by developing a formalized measurement of complexity for each device.

This study then uses this device-complexity metric to construct an anomaly-based behavioral model specifically tuned to each device. This tuned model adapts the probability threshold of inlier behavior in proportion to the measured complexity of each device, improving the overall accuracy of the predictive model.

To demonstrate the model efficacy, this research analyzes the confidence of each device's tuned behavior against seven common types of malware traffic from infected devices. To illustrate that the model can be effectively applied to a broad spectrum of devices, four different IoT datasets were analyzed: one residential dataset, two lab datasets, and a dataset based on commercial IoT devices.

Research Contributions

- **Formalized measure of device complexity:** I create and formalize a general measure of a device's network traffic complexity. The measure is agnostic of the device type, capability, and independent of how much the device is used.

- **Complexity tuned anomaly detection:** I construct anomaly detection classifiers that employ the complexity measure to customize the model to each device.

- **Accurate anomaly detection model:** The complexity-based anomaly detection model was evaluated on a diverse set of real-device and real-attack traffic. It is shown to be accurate, particularly in low complexity devices that are vulnerable to attacks.

1.3 Research Questions and Objectives

Some of the questions answered in this study are:

1. How should complexity be measured in devices?

2. How does the measured complexity of devices vary based on the use of each device?

3. How accurate is the calculated complexity behavior at distinguishing between normal and abnormal traffic?

4. How can complexity be used to create accurate anomaly detection models?

1.4 Definition of Terms

Gaussian Mixture Model (GMM): Probabilistic model that fits all data points to a finite number of normal distributions. It is useful in multi-modal data [10].

Density-based Spatial Clustering Applications with Noise (DBSCAN): A clustering algorithm that can find high-density clusters of data of arbitrary size [11].

Network flow or netflow: A tuple of data that represents a connection state in the network, shown in Table 3.1.

Software-Defined Networking (SDN): Networking architecture that decouples the control and data planes, allowing the network to be programmatically controlled.

1.5 Assumptions

It is assumed that the device is not compromised a-priori, meaning that the traffic learned by the model, the training data, is initially not malicious in nature, and thus all explicitly considered normal.

1.6 Organization

This thesis is organized as follows:

- Chapter 2 - Related Works: I review the recent and relevant research in the area of network-based security as it applies specifically in the context of IoT, including a review of supervised and unsupervised techniques to establish identity and behavior on the network, as well as the use of anomaly detection on IoT devices.

- Chapter 3 - Methodology: I discuss how the data were collected and formatted, as well as what techniques were applied to analyze the data. I discuss the methodology applied to calculate the complexity of each device and how devices can be modeled using the Gaussian Mixture Model (GMM).

- Chapter 4 - Results: I review and discuss the complexity measurements and performance results of the GMM across the four datasets.

- Chapter 5 - Discussion: I compare and discuss how results differ from dataset to dataset, and how they compare across datasets. I discuss how the results inform the utility of the complexity metric in increasing accuracy of predictive models based on anomaly detection.

- Chapter 6 - Conclusion And Future Research: I conclude with the importance and contributions of the research to IoT security and propose future directions for further research.

Chapter 2

Related Works

This chapter provides background and related work on network-based IoT security, and focuses on methods of establishing identity, behavior, and classification of devices on the network. This chapter begins with an overview of deterministic methods of establishing identity and behavior, then moves on to probabilistic and learning-based methods, covering both supervised and unsupervised learning. It concludes with previous research on complexity, including how it is calculated and its effects on probabilistic models.

2.1 IoT and Security

As presented in the previous chapter the number of IoT devices is growing at an exponential rate. If history is any indication, insecure and vulnerable devices will grow at least at this same exponential rate. To address this growing problem, there is an increasing corpus of research and work proposing the network as an active participant in securing devices. There are two approaches in the literature that show how this can be done: (1) explicitly informing the network about the identity of the device and the policy defining its behavior, (2) training the network to identify devices by analyzing the device's network traffic to develop a dynamic policy based on learned device behavior. These are summarized below:

1. **Deterministic:** This method defines a set of rules and policies that are enforced within the network. These rules can be static or dynamic based on roles, attributes, and capabilities. The major drawbacks of this approach are that the rules and policies are either too coarse in their implementation and do not adapt to the complexities of modern networks, or they suffer from compounding policy inflation that quickly becomes overly complex and impracticable to implement.

2. **Training the Network:**. This approach applies machine learning to enable the network to learn about the devices. This can be split into two techniques.

   Supervised Learning: This method learns from labeled data to classify the device identity or type. This classification is then passed to policy engines to make enforcement decisions on the traffic. Supervised learning has a major shortcoming, in that it requires large sets of labeled data with which to train classifiers. These datasets may not be available, especially in the case of new or rare devices. Supervised learning cannot classify these unknown devices and traffic, making it inflexible and difficult to adequately scale.

   Unsupervised Learning: This method does not require labeled data and learns from unstructured data. Unsupervised learning can learn patterns in the data such as grouping similarities or by classifying inliers and outliers.

2.2 Deterministic Research

Access control policies have been around for decades, and are implemented in firewalls that block or allow traffic based on generalized rules using IP address source and destination, ports, and protocols. These rules do not truly identify what is on either side of the connection, nor do they adapt to changes on the network.

Recent network security efforts employ a cryptographic identity in the form of a Public Key Infrastructure (PKI) certificate tied to each device and validated by the network. This method is the basis for the WiFi Alliance's Easy Connect specification, also known as Device Provisioning Protocol (DPP) [12]. When a device is onboarded, the network establishes trust and identity with the device based on the asymmetric key pair embedded in the device. This PKI certificate establishes an identity that may include the make and model of the device but does not establish any policy for that device.

To fill this gap and provide a policy associated with a device, work done by the Internet Engineering Task Force (IETF) introduces a specification called Manufacturer Usage Description

(MUD) [13], using MUD the device presents the network with a URL pointing to a network policy that describes what level of communication the device requires for its normal function.

2.3 Supervised Learning Research

Several works derive device identity based on network traffic using a supervised learning approach. Miettinen et al. [14] have developed a method, called IoT Sentinel, that uses machine learning to designate a device type on the network, referred to by the authors as a device fingerprint. Using the random forest algorithm and 23 network features they were able to identify device types on the network based only on the device's traffic. The 23 features are based on layers two, three, and four of the Open Systems Interconnection (OSI) networking stack. Expecting that the body of the packet will be encrypted, all the features the authors employed are based on unencrypted parts of the traffic like IP header information.

The identification was made in the initial setup phase of the device on the network. As the device was initially joined and onboarded onto the network, the authors identified it based on up to 12 of the first packets captured. Once the device was identified, the authors query Common Vulnerabilities and Exposures (CVE) to determine if the device has any vulnerabilities. If the device is found to be vulnerable, they use a customized SDN controller and switch, to segregate the device into three zones: (1) strict - no Internet access, allowed to communicate only with other untrusted devices, (2) restricted - able to talk to a limited setup of vendor addresses and other untrusted devices, and (3) trusted - unrestricted Internet access and the ability to communicate with other devices on the network.

The authors reported that using the random forest algorithm allowed them to identify the 27 device types in the study with an accuracy of 81.5%. They noted that for 17 of the devices they were able to identify the device with a 95% accuracy. The other ten devices they achieved only a 50% accuracy. These ten devices were composed of largely different devices from the same manufacturer. The authors explain that their classifier is good at discriminating between devices that have different hardware or firmware, but does not accurately fingerprint the devices made by the

same manufacturer and have the same firmware. They make the assumption that two very similar devices are likely to have the same vulnerabilities and therefore the low accuracy in identification is inconsequential.

While IoT Sentinel presents some useful solutions, it has weaknesses that need to be addressed in future research. First and foremost is that they use a supervised learning algorithm that must be individually trained on each device type, and must be re-trained if the device firmware changes. This re-training makes the approach difficult to scale to a large set of heterogeneous devices and requires an extensive online database of trained classifiers. This approach is therefore reliant on the accuracy of, not just one but, two separate public databases: the CVE database, and a database of trained classifiers. Second, by only analyzing the device during setup, they are missing the majority of the device behavior on the network. If the device is compromised after being installed, this solution is unlikely to recognize it. Third, the authors' classifier was unable to distinguish between very similar devices with high accuracy, which could result in inaccurate identification of important devices such as those used to monitor health.

Bezawada et al. [15] build on the the IoT Sentinel method by using a machine learning approach to broadly identify the device and place it in a predefined category, i.e a light bulb. According to the authors, even devices from different manufacturers can be placed into general categories. For example, light bulbs from different manufacturers can be correctly identified and placed into a lighting category. Their results show that they can do this with an accuracy between 91-99% across the device categories.

This approach of fingerprinting devices also uses a supervised approach to fingerprinting and categorization and requires labeled data that may not be available. Additionally, the authors make the assumption that they will be able to detect a distinct command and response structure in the data from any particular device which is not always possible. As this research used a relatively low sample size of devices, it is unclear if this approach could categorize more complex devices, for example, devices with encryption that may interfere with the ability to detect the command and response structure.

2.4 Unsupervised Learning Research

Moving beyond the limitations present in supervised learning, authors Marchal et al. developed a technique to generically identify devices based on the periodicity of their network communication [16]. The authors employed signal processing techniques to analyze devices' background periodic traffic with the goal of placing the devices into virtual device identity groups. Using discrete Fast Fourier Transform, the authors then generate 33 features that they then use to train a k-nearest neighbors (kNN) model. The authors then use the model to place devices into one of 23 virtual groups based on the clusters found by the kNN.

This method generated a unique derived identity for nearly 70% of the devices, which is only slightly better than a derived identity per device. For example, the largest derived identity group contains only five devices, all of which are from D-Link. Applied to a wider set of devices across a wider set of manufacturers, this work would produce a very large number of derived identities. This large quantity of derived identity groups would not do much to simplify network and security policies.

Ortiz et al. developed a probabilistic method of measuring the distribution of traffic for an IoT device [17]. To build their model they employed a stacked autoencoder to autonomously learn network traffic features from IoT devices. The authors report that the model identified previously seen devices with a 99% accuracy and recognizing unknown devices to the extent that an IoT/Non-IoT grouping could be inferred for each device.

Ortiz et al. then used an LSTM (Long Term Short Term Memory) neural network to learn from the inherent sequencing of TCP flows [17]. The LSTM autoencoder layer learns a compact feature representation of the data by taking an initial input and forcing the output to be smaller than the input. A second pass decoder takes this compressed output and uses it as input to derive the original feature vector. The model is trained to minimize the differences between the encoder and the decoder. The goal of the LSTM autoencoder is to capture latent feature representation in the data. By chaining autoencoders, the authors claim that the final derived feature set maximizes the differences across sequence representations.

Next, the authors used the derived classes from the LSTM phase to arrive at a normal distribution over the encoded data and derive a probabilistic model for each device. This probabilistic distribution forms the root of their definition of device behavior. By comparing distributions from various devices they were able to cluster devices based on similarity.

The cluster method used in this work does not generate a true identity and cannot accurately identify what an unseen devices is, only that the device is new and has not been seen previously by the model. The authors do not have devices in their test set that are similar in nature, and they do not address whether they can distinguish between two similar devices from the same manufacturer, such as an Amazon Echo and an Amazon Echo Show. The authors report a large number of similar devices in their dataset, but they do not include any of these in the results.

Ren et al. developed a network-informed approach based on destination IP to enumerate and analyze IoT behavior [18]. The authors set up two labs, one in the US and one in the UK. The labs consisted of a total of 81 devices with 26 devices being common between the two labs. The authors then proceeded to analyze the traffic for each lab looking at the behavior of an IoT device during boot and when it was actively controlled and/or interacted with.

Next, the authors analyzed the destination IPs and categorized these destinations into three categories: first party destinations, support destinations, and third party destinations. First party destinations are those where the device contacted an IP that belonged to the device's manufacturer or company associated with the device. Support party destinations are those where the IP belonged to a cloud or content delivery network (CDN). All other connections the authors considered to be third party destinations where the owner of the IP had no clear and direct connection to the device. Some of these third party destinations include service sites such as netflix.com and advertising networks such as doubleclick.net.

The authors analyzed the entropy of encrypted device traffic and based just on the IP headers they report that they can infer types of devices such as appliances and cameras as well as activities associated with devices such as video, voice, or movement. If true this has privacy implications insomuch as determining what devices and occupants are doing even the traffic is encrypted. This

method does not use machine learning directly, but provides an example in the literature where unsupervised learning informs the network by learning patterns and aspects of the network traffic from devices.

2.4.1 Anomaly Detection

One major class of unsupervised learning algorithms is anomaly detection. This is a very active area of research, particularly within the context of IoT.

Alrashdi et al. present AD-IoT, an anomaly detection method based on the random forest algorithm [19]. The authors develop an algorithm called, "extra trees" to identify the 12 most important features in a dataset with normal/abnormal labels from the University of New South Wales titled UNSW-NB-15. The authors provide results of binary classification (normal vs. attack) on the dataset with high (>98%) F1 scores for an average of normal and attacks.

AD-IoT has some drawbacks in that it only uses a single dataset of generated synthetic attack traffic developed by a third party, none of which is IoT specific. Additionally, the authors report an F1 score of 0.87 on correctly classifying the attack traffic that makes up only 3% of the total traffic, and an F1 score of 0.99 on the normal traffic that was 97% of the total traffic. The final result of an average F1 score of 98% could be misleading as it is heavily weighted toward the analysis of the normal traffic.

Authors Hasan et al. examine several machine learning techniques for detecting anomalies in IoT data including logistic regression, support vector machines, decision trees, random forest, and neural networks [20]. This work used a synthetically generated dataset consisting of 347,935 normal examples and 10,017 anomalous examples, each with 13 features. The authors, like those in the previous work, found that the random forest method was the best overall in terms of F1 score with F1=0.99 for both training and testing.

While this work presents very good results (F1=>98%) for all the machine learning techniques used, they were all based on a completely synthetic dataset based on emulated IoT devices.

2.5 Complexity and Predictability

The correlation between complexity and predictability is an intuitive and foundational principle of probability theory that has published roots dating back as far as 350 B.C.E with Aristotle's *Posterior Analytics* where he posited on drawing conclusions from uncertainty [21]. The challenge is to determine a statistically significant way of measuring the complexity of a system to inform meaningful confidence in a predictive model of that system.

Formalized measurement of complexity as applied in a computer science context is probably most often associated with the works of Andrey Nikolaevich Kolmogorov, who defined the complexity of an object as the shortest computer program to produce the object as an output [22]. This simple notion arises again in the work of Jorma Rissanen whose work on the minimum description length principle that establishes that the best model for a set of data is one that leads to the best compression of the data [23].

Ceccatto and Huberman [24] establish a quantitative method for measuring the complexity of hierarchical systems by evaluating nodes and bifurcation factors of tree structures. The authors show that complexity saturates as the structure of the tree's lower level grows and that any large tree structure's complexity grows linearly with the number branching levels.

In the paper Predictability, Complexity, and Learning authors Bialek et al. establish a formal result that predictive information provides a general measure of complexity [25].

In this work, I propose that the relationship between predictive information and complexity is commutative, i.e. not only does predictive information lead to a measure of complexity, but that complexity provides a general measure of predictive information. In machine learning, this relationship leads to the logical notion that the less complex the model the more accurately it can be modeled.

2.6 Network Measurement

Some notion of network complexity based on a sources request has existed in previous works. Authors Allman et al. present a metric for determining if a particular IP source is a scanner or

not based on heuristic they call service fanout [26]. This service fanout is a measure of successful connections to attempted but unsuccessful connections. The authors classify a remote host as a scanner if it has at least four attempted but not successful connections and the that these unsuccessful attempts outweigh successful attempts by a ratio of two-to-one.

While this work focuses on network scanning and not specifically connections sourced from IoT devices, it does present a step toward defining the complexity of network traffic based on connections.

2.7 Previous Publications of Related Works

Two previous works on complexity and IoT have been published that relate to this dissertation. In the paper ComplexIoT: Behavior Based Flow Control For IoT Networks, Haefner and Ray describe several measures of complexity of IoT devices and how these measures can be used by a network for autonomous enforcement of network traffic [27].

The paper, Trust and Verify: A Complexity-Based IoT Behavioral Enforcement Method authors Haefner and Ray expand upon this work by using a tuned search of single class SVM models to search for a model that minimizes false positives [28].

2.8 Summary

The deterministic methods provide the network with a cryptographic secure process for determining what a device is. The MUD-based policy, if secured by the certificate presented by the device, provides an approach to giving the network both an identity and the connection requirements for the device in a deterministic and cryptographically verifiable manner. There are several drawbacks to this strategy. First, this will require compliance across every device manufacturer. This increases costs and the complexity to install and issue and manage keys for all devices. Second, a public key infrastructure at the scale of billions of devices is magnitudes of order larger than any currently deployed today. Third, even with good identity and policy, there is always the

possibility of a vulnerable device being compromised. This could compromise the private key and consequentially the trust between the network and the device.

Supervised learning has had huge success in recent years in many fields. As established by several works above, it can provide a highly accurate identity of a device without the requirement of a PKI certificate. This identity could be matched to a known policy for this device and give the network both a way to determine what device, and also connection requirements for that device. This approach's biggest deficiency is that it requires a large amount of labeled traffic for each device, something that is not readily available. Additionally, a device's traffic can change with updates, requiring that the model be completely retrained on the new labeled traffic. Last, a supervised method cannot classify devices for which it has no labeled training data. This makes it difficult to scale beyond the most common and popular devices.

Unsupervised learning does not require labeled data and instead learns patterns from the data itself. Novelty and anomaly detection techniques can learn what is statistically normal traffic from past examples and determine if new traffic is normal or anomalous. This approach can provide some broad categorization of devices based on similarities and can learn the behavior of the network traffic of a device. The drawbacks to unsupervised techniques, especially in the context of IoT analysis, are that they are often applied generically across the devices without any input as to what the underlying device is. This can lead to devices that are highly dissimilar being analyzed without taking into account their differences in function and capability.

The informed-unsupervised method presented in my research attempts to overcome some of the downsides of supervised and generic unsupervised techniques by informing the anomaly detection of the underlying device properties in terms of complexity. It does this without labeled traffic or requiring any knowledge of what the devices are. Additionally, all device-data comes from real, not generated devices and attack data comes from captures of several different types of real malware.

Take, for example, a refrigerator that is also an Android tablet. The methodologies in the related works above would struggle to characterize such a device, as a supervised model might not have a matching training set of data, and a generic unsupervised approach might treat this highly complex

device the same as it would a simple appliance. The method presented in this study does not try to recognize this device as either a refrigerator or a tablet, and it does not try to guess at the service or characterize the device's application layer data. My model does not rely on learning specific human interactions with the refrigerator, nor determining if those interactions are anomalous. My model only relies on how complex the refrigerator appears on the network and how much it stays within the learned boundary of behavior.

Chapter 3

Methodology

3.1 Introduction

This research is a study of the network traffic from devices running on four separate networks. It compares results across these four datasets, all of which contain devices of different types, categories and capabilities. This chapter discusses how the study was designed, how data were collected, how devices were analyzed for complexity and behavior, and then evaluated against a set of attack data.

3.2 Research Design

This research uses data from IP-based networks consisting of mixed IoT and general-purpose devices. All analysis is done on a per-device basis using the source address of the device as an identity for the device. Devices that had fewer than 100 flows were not analyzed as this was deemed too few flows to properly represent a device. Definition 3.2.1 establishes a consistent ontology for defining flows with respect to this research.

Definition 3.2.1. Network Flow: A sequence of packets where each has the same tuple comprised of: a source address, a destination address, a source port, a destination port, and the protocol. TCP based flows were ended at the end of the TCP session. For UDP based flows, the flow timeout was established when no data were sent or received for 15 seconds.

This research uses flow data features to measure the complexity of each device on a network based on its past traffic. Figure 3.1 shows the spectrum of device complexity. Devices such as sensors, light bulbs, were expected to have low complexity in terms of their network traffic. Conversely, devices such as laptops and smartphones were expected to be high complexity devices. Ontological Definitions 3.2.2, and 3.2.3 for single-purpose and general-purpose devices are used throughout this study.

Table 3.1: Data Features

Feature	Abbreviation	Description
IPV4_SRC_ADDR	sIP	IPv4 Source Address
IPV4_DST_ADDR	dIP	IPv4 Destination Address
IN_PKTS	iFP	Incoming flow packets
IN_BYTES	iFB	Incoming flow bytes
OUT_PKTS	oFP	Outgoing flow packets
OUT_BYTES	oFB	Outgoing flow bytes
L4_SRC_PORT	sP	IPv4 Source Port
L4-DST_PORT	dP	IPv4 Destination Port
PROTOCOL	p	IP Protocol Identifier

Figure 3.1: Device Complexity Spectrum

Definition 3.2.2. General Purpose Device A device that is capable of running multiple user-space applications. Some examples are smart-phones, tablets, laptops, some streaming devices, and smart TVs. These devices are expected to have higher network complexity measurements.

Definition 3.2.3. IoT - Single Purpose Device A device that generally runs a single application. They often are capable of running only one or two threads. These devices are expected to have lower network complexity measurements.

3.3 Datasets and Collection

This work examines data collected from the following four environments of IoT devices, and one attack dataset of malware traffic:

- **Home Network:** This dataset was developed as part of this research and contains devices from a residential network that are in regular use in an unstructured manner as part of daily use.

- **Lab Network:** This dataset was developed as part of this research and contains devices that are set up for the purposes of testing. There is very little active use of the devices in this dataset except that which was done in experiments by graduate students.

- **University of New South Wales Lab [29]:** This dataset is available online and contains several mixed-use devices. It is provided in the form of network capture files.

- **SCADA Network [30]:** This dataset is a series of captures from the Methane Emissions Test and Evaluation Center (METEC). It is provided in the form of network capture files.

- **Attack Dataset [31]:** This dataset is comprised of data captured from infected devices. It is provided in the form of network capture files, with labels describing the attack/malware type.

For the home and lab datasets, flows were collected directly from the routers using a netflow collector. For the UNSW and SCADA datasets, network captures were parsed and turned into flows using a tool called JOY [32], which can convert a capture file to network flows.

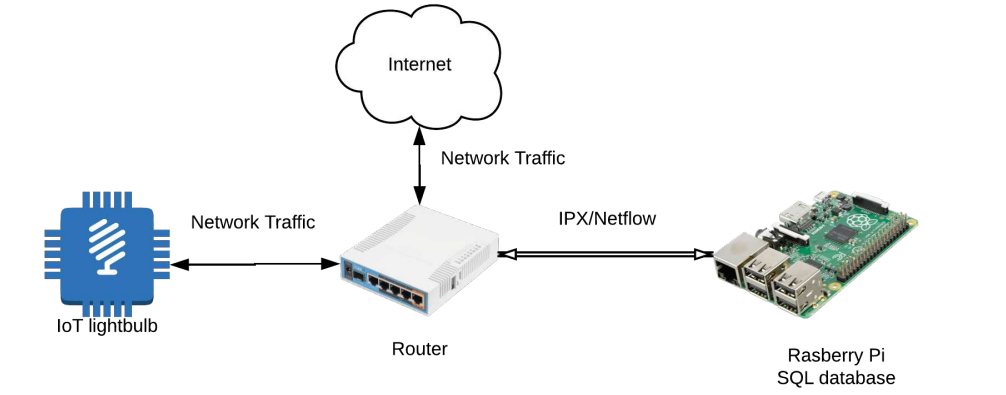


Figure 3.2: Data Collection Architecture Home and Lab

3.3.1 Home Dataset

Data were collected from a residential network with approximately 25 devices over the course of 37 days. These devices range from general computing devices like laptops and smartphones, to middle-complexity devices, such as IoT hubs with several IoT devices using Zigbee or Zwave, and to single-purpose devices, such as light bulbs and temperature sensors. Data were collected by a central MicroTik router, shown in Figure 3.2, that sends Netflow/IPX data to nprobe, a flow capture software, running on a Raspberry Pi. Flows were stored in a MariaDB relational database. Table 3.1 shows the features of the data collected.

Flows were aggregated with a maximum of 30 minutes per flow. The inactive flow timeout was set to 15 seconds. If the devices have not exchanged traffic in 15 seconds, the flow was completed and recorded. Device identity was established based on the source IP address for each device. The test environment is configured such that the devices always receive the same IPv4 address.

The list of devices in the home is shown in Table 3.2.

Table 3.2: List of Home Devices

Home Devices		
Amcrest Camera	Plex Server	Raspberry PI 3
Google Home	Galaxy Note 8	Smart Things Hub
J. Chromebook (Asus)	Xbox One (2)	Appple Macbook Pro
Philips Hue Hub	Chromecast	Echo Dot
Eufy Doorbell	Motorola Android	HP Stream Laptop (2)
Eufy light bulb	TP Link Switch	Roku Express
B. Chromebook (HP)	Brother Printer	Roku Stick
Fire Tablet (3)		

3.3.2 Lab Dataset

The lab dataset consists of netflow data collected from approximately 24 devices in a lab located in the computer science building on the Colorado State University campus. It consists of over 3 million flows that were gathered over a period of months in the spring of 2019.

The lab is managed by a thin client server running Ubuntu server 16.04. The access point is controlled by hostapd [33], a user-space access point reference implementation, and the network traffic is routed by the OpenFlow [34] switch, Openflow Virtual Switch (OVS) [35]. The lab devices are listed in table 3.3. Flows are sent by OVS to a server in the lab and stored in a SQL database. Device identity was established based on the source IP address for each device. The test environment is configured such that the devices always receive the same IPv4 address.

Table 3.3: List of Lab Devices

Lab Devices		
TP-Link Camera	WinkHub (2)	Google-Home-Mini (2)
Koogeek (2)	Amazon Echo Show	Wall Dimmer
Samsung Smart TV	MacBook Air	Arlo Q
Apple TV (2)	Amazon Echo 2nd Gen. (2)	Amazon Firestick (2)
Amazon Echo Dot (2)	Ubuntu Laptop	Android Phone
Lutron Light Bulb	Windows Laptop	

3.3.3 University of New South Wales Dataset

The dataset from the University of New South Wales [36] consists of approximately 30 devices, shown in Table 3.4, that vary from common IoT devices to general-purpose devices, such as laptops and iPhones. This dataset comes from a lab environment where students were encouraged to interact with the devices. The dataset consists of 20 capture files with a total of 12 GBs. Using the JOY [32] tool this dataset was transformed into over 2 million flows and stored in a SQL database.

3.3.4 SCADA Dataset

The SCADA dataset consists of captures from approximately 40 devices gathered over the summer of 2019 from the Methane Emissions Test and Evaluation Center (METEC) [30]. Using the JOY tool this dataset was converted into over 300,000 flows and stored into a SQL database.

The SCADA dataset consists of a single general-purpose device, a desktop to control computer, and some IoT devices such as printers. This dataset is unique in that the majority of the remaining

Table 3.4: List of UNSW Devices

UNSW Devices		
Smart Things	Amazon Echo	Samsung SmartCam
Withings Baby Monitor	Nest Protect Smoke Alarm	Samsung Galaxy Tablet
TPLink Router	Belkin Motion Sensor	Netatmo Welcome Camera
TPLink Cloud Camera	Dropcam (2)	Belkin Wemo Switch
TPLink Smart Plug	Netatmo Weather Station	Withing Smart Scale
Triby Speaker	PIX-Star Photo Frame	HP Printer
PiX Star Photo Frame	HP Printer	Insteon Camera
IHome	Withing Sleep Sensor	LifX Smart Bulb
Android Phone (2)	MacBook	Blipcare Blood Pressure
Laptop	Iphone	MacBook Iphone

devices are industrial remote telemetry units (RTUs) called LabJacks [37]. All devices in this dataset are shown in Table 3.5.

Table 3.5: List of SCADA Devices

SCADA Devices		
10.1.107.255	10.1.106.255	labjack 227
labjack 226	labjack 213	labjack 212
labjack 211	labjack 210	labjack 209
labjack 208	labjack 207	labjack 206
labjack 205	labjack 204	labjack 203
labjack 202	labjack 201	labjack 189
labjack 188	labjack 187	labjack 186
labjack 185	labjack 184	labjack 183
labjack 181	labjack 178	labjack 177
labjack 176	labjack 175	labjack 174
metec gc	camera	metec control
printer	metec test	

3.3.5 DDoS Attack Dataset

This dataset is published by Stratosphere Laboratory [31] and contains 20 captures where malware was executed on Raspberry PI computers, and 3 captures for benign IoT devices traffic. This dataset was first published in January 2020, with captures ranging from 2018 to 2019. The attack dataset consists of seven separate traffic profiles of malicious flows. The attack datasets were orig-

Table 3.6: Attack Dataset

Attack Name	Description	Unique Flows
C&C	This is traffic where a device is connecting to a remote command and control server.	30
C&C Heartbeat	This is traffic that is meant to monitor the status of an infected host.	3
C&C Torri	This is command and control traffic specifically from the Torri botnet.	1
C&C FileDownload	This is traffic from an infected device downloading a file or malicious payload	7
DDoS	This is traffic where a device is participating in a distributed denial of service attack	1
Part of Horizontal Scan	This is traffic where a device is scanning locally on the network.	49959
Okiru	This is traffic specifically from the Okiru botnet.	99888

inally over 12GB of data. To reduce analysis by redundant data the dataset was trimmed down to unique flows per profile listed in Table 3.6. For example, the DDoS profile had over 19 million examples, but after trimming had only a single unique flow.

3.4 Data Analysis

The data analysis of the four device datasets can be broken into three steps per dataset:

- **Device Complexity Analysis:** a measurement of the variance of the traffic of devices, an analysis of the complexity of the set of IP addresses each device contacts, a sum of unique flows from each devices, and a noise to signal ratio (NSR) measurement.
- **Device Behavior Analysis:** an anomaly detection algorithm based on the a Gaussian mixture model (GMM) to establish a normal set of flows for a device during a training period.
- **Attack Analysis:** a demonstration of how well the model for the device can differentiate normal device traffic from the attack traffic.

3.4.1 Device Complexity Classification

This research examines several ways to measure the complexity of a device on a network. The following methods are constructed based on their network traffic, and the pros and cons of each are noted:

- **Traffic Variance:** An examination of how the traffic varies from flow-to-flow using the packets-per-second (pps) and bits-per-second (bps) on outbound connections from a device.
- **IP Complexity:** An analysis of the number and hierarchical structure of destination IP addresses connected to by a device, and includes both Wide Area Network (WAN) and Local Area Network (LAN) based traffic.
- **Unique Flows:** The canonical set of unique flows generated from a device, taking into account both the destination features and the aggregate of the flows.
- **Noise to Signal Ratio (NSR):** The grouping by density of the destination port and destination IP address tuple, where high-density points are formed into clusters, designated as signals, and low-density points are designated as noise.

3.4.2 Device Variance

The variance metric comes from the simple notion that devices on a network present different variances based on their behavior on the network. To calculate the variance in Equation 3.1, I employ the unbiased variance score computed over the flow history of the device. The variance score gives us a normalized measure of the dispersion of the data between a training subset and a test subset. In this research, variance is only calculated on the aggregates, oFP and oFB of data for each flow shown in Table 3.1. This is shown programmatically in Algorithm 1.

$$Var(f) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{3.1}$$

$Where : \{x_1, x_2, \ldots, x_n\} =$ flow sample

Input: n flows

Output: Device Variance d_v

let n = 0, sum =0, sumSq =0

for *x in set of flows* **do**

n++

sum = sum + x

sumSq = sumSq + x^2

end

d_v = (sumSq -(sum^2)/n)/(n-1)

Algorithm 1: Device Variance

3.4.3 Device IP Complexity

This metric examines how devices form outbound connections. It is not sufficient to simply count the number of unique IP addresses that a device connects to, as it would not capture the hierarchical structure formed when many IP addresses belong to the same IP subnet. IP based subnets have strong correlations to the same company and/or service given how IP address space is allocated in hierarchical blocks. To provide cloud-based services, companies will often use many servers and often will use specific blocks of addresses for specific services.

To adequately capture this grouping of IP addresses: this research introduces two concepts used in the analysis of IP addresses, IP spread and IP depth. IP spread is the number of unique first-order octets that a device connects with. IP depth is the ratio of fourth-order octets to second and third-order octets. Conceptually, these connections form tree structures where the first-order octet is the root, the second and third octets are branches and the fourth-order octets are leaves as described below:

Definition 3.4.1. IP Root: An IP root is a unique first-order octet, plus all common addresses that share this octet.

Definition 3.4.2. IP Branch: A second, or third-order octet that has one or more fourth-order octets (Leaf) under it.

Definition 3.4.3. IP Leaf: A unique fourth-order octet.

Definition 3.4.4. IP Spread: The sum of total unique IP addresses that have a unique first octet interacting with a device.

Definition 3.4.5. IP Depth: The ratio of IP Leaves to IP Branches.

IP complexity shown in Equation 3.4 is calculated by taking the IP Spread in Equation 3.2 divided by the IP Depth calculated in Equation 3.3. This is shown programmatically in Algorithm 2.

Device IP Spread

$$IP_{Spread} = \sum IP_{trees} \tag{3.2}$$

Device IP Depth

$$IP_{Depth} = \frac{\sum IP_{leaf}}{\sum IP_{branch}} \tag{3.3}$$

Device IP Complexity

$$d_{ipc} = \frac{IP_{Spread}}{IP_{Depth}} \tag{3.4}$$

```
Input: Set of IP addresses for device, stored as a trees
Output: Device IP Complexity
for ipTree in ipForest do
    if ipTree.FirstOctet is unique then
        ipSpread++
    else
        if ipTree.SecondOctet is unique then
            totalBranches++
        end
        if ipTree.ThirdOctet is unique then
            totalBranches++
        end
        if ipTree.FourthOctet is unique then
            ipLeaves++
        end
    end
end
ipDepth = ipLeaves/totalBranches
ipComplexity = ipSpread/ipDepth
```

Algorithm 2: Device IP Complexity

A large number of IP trees with few branches indicate a large IP spread. A small number of IP trees with many branches and leaves indicate a large IP depth. IP depth/spread is used as one measure of a device's complexity. Devices belonging to a single ecosystem, such as Google Home, should have a small number of broad trees, as they connect mostly to Google's networks which are dedicated to these types of devices. Other devices such as laptops and smartphones should have a larger IP spread, with each IP having fewer branches and leaves.

3.4.4 Unique Flows

Unique flows F is a cardinality of the set of flows from a device where flow tuple $f = (dIP, dP, oFP, oFB, p)$. Unique flows are shown calculated in Equation 3.5.

Unique Flow

$$F = \mid \{f_i\} \mid_{i \in (1,\ldots,n)} \tag{3.5}$$

$n =$ number of analyzed flows

3.4.5 Noise to Signal Ratio (NSR)

This measure of complexity uses the DBSCAN [11] clustering algorithm to compute the number of clusters (signals) and the non-clusters (noise), from the data points defined by the destination IP and destination port. This algorithm is good at finding areas of high density that are separated by areas of low density. The DBSCAN algorithm has several advantages in that it can find clusters of arbitrary shapes and sizes, including clusters that are non-convex (unlike k-means).

The DBSCAN is initialized with two important parameters used to tune how clusters are found: a distance parameter ϵ and the number of points that are within a specified distance, $min\_samples$, to form a cluster. To calculate the distance parameter I use the ip_spread found in Equation 3.2 multiplied by 128 (the midpoint of the address space of a class C network). The calculation for the ϵ parameter can be found in Equation 3.6.

$$\epsilon = 128 * IP_{Spread} \tag{3.6}$$

For the second parameter, $min\_samples$, I found experimentally that $min\_samples = 10$ was a good starting value for the total number of neighborhood points necessary for a point to be calculated as a core point. The number of clusters found by the DBSCAN algorithm and the number of non-clusters are used to calculate the Noise to Signal Ratio (NSR) for the device using Equation 3.7.

$$NSR = \frac{n\_noise}{n\_clusters} \tag{3.7}$$

3.4.6 Behavior

The behavior of an IoT device in this research dissertation is defined by the points of data learned to be inliers from the training set. To distinguish an inlier from an outlier, I employ a clustering method called Gaussian Mixture Model (GMM) [38]. The GMM is a finite mixture model used to find the probabilities of multivariate distributions. The traffic features of netflows from IoT devices consist of multivariate data with high probability densities based on the connections that

devices make. The theory is that devices make connections to common sets of endpoints and over similar sets of destination ports, leading to high density areas over the destination port/destination IP ordered pairs in Euclidean space. An example of this can be seen in Figure 3.3 where there are clear sub-populations of probability density in several of the features.

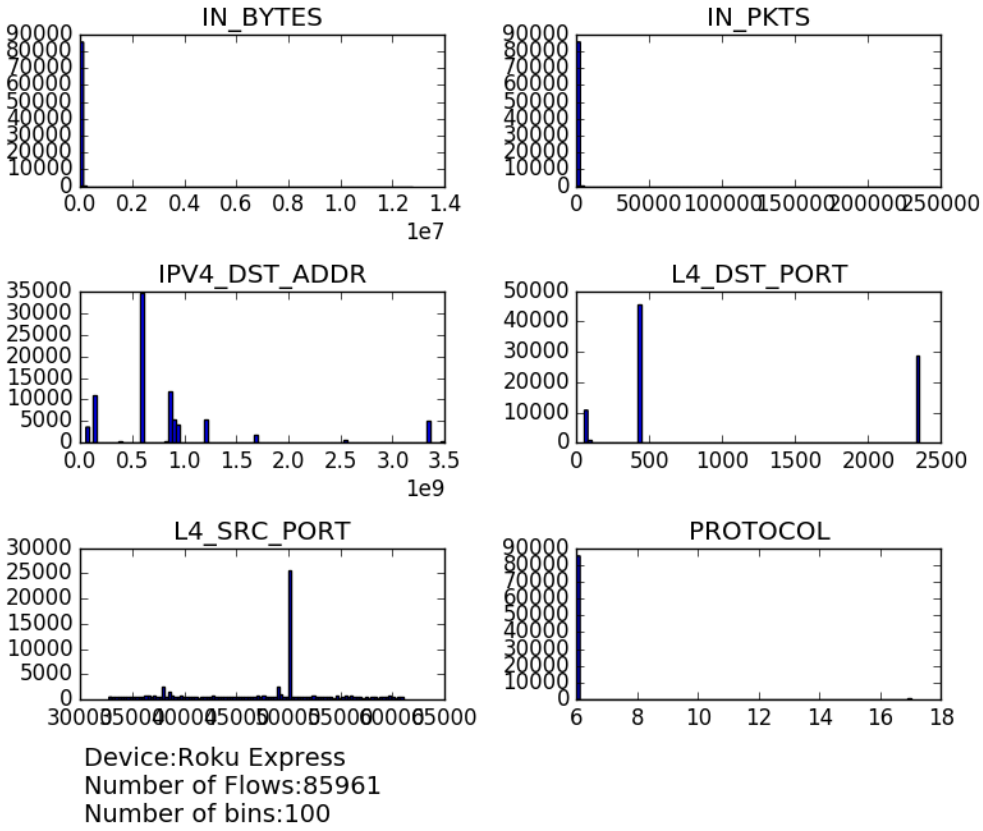


Figure 3.3: Data Distribution of Roku Express

The Gaussian Mixture (GMM) takes an input of the expected number of Gaussian distributions to use, then gives a probabilistic assignment of each point to each Gaussian. This method is useful to quantify both certainty and uncertainty, and allows me to pick the boundary of probability where a point is assigned as an inlier or an outlier. Equation 3.9 shows the mean. Equation 3.10 shows the standard deviation. The Gaussian probability is shown in Equations 3.11 and 3.12. Outliers are calculated where the probability is less than X and inliers are calculated where the probability is greater than X. For this work, X was statically set for all devices to the Chi-Square likelihood of

$X = 5.991$ given two degrees of freedom this value provides a 95% probability confidence. Prior to analysis by the GMM all data were standardized and scaled to unit variance as calculated 3.8.

Data Standardization Standardizing the the inputs of destination IP address and destination port is important as these two value are of very different scales. The standard score of a sample x is calculated as:

$$f(x) = \frac{x - u}{s} \tag{3.8}$$

where u is the mean of the training samples, and s is the standard deviation of the training samples.

Definition 3.4.6. Gaussian Behavior Model (GBM): The total Gaussian mixture for the device, i.e. the set of all Gaussians found on the training data.

Definition 3.4.7. Inlier: Points are defined as those that are probability X to be in a particular Gaussian.

Definition 3.4.8. Outlier: Points are defined as those that are probability less than X to be in a particular Gaussian.

Mean

$$u_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)} \tag{3.9}$$

Standard Deviation

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - u_j)^2 \tag{3.10}$$

Outliers

$$X \leq \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{x_j - \sigma_j)^2}{2\sigma_j^2}) \tag{3.11}$$

Inliers

$$X \geq \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\sigma_j}} exp(-\frac{x_j - \sigma_j)^2}{2\sigma_j^2}) \tag{3.12}$$

Picking N-Components for Gaussian Mixture

The number of clusters in a GMM is very important in that it affects how well the model fits the training data. Too many clusters results in over-fitting and too few results in under-fitting. Determining the correct number of clusters can be ambiguous and context specific. The Bayesian Information Criterion (BIC) is well known in the literature as a general method for model selection [39]. The BIC prevents models from over-fitting by introducing a penalty term,

$$k * log(n),$$

on the number of parameters found, where n is the sample size and k is the total number of parameters. At least within the narrow context of IoT traffic as presented in this work I show that the BIC methods tends to produce too many clusters for the given IoT data.

In this work I present an alternative to the BIC using the 'signal' part of the NSR (based on DBSCAN). This method of model selection has two distinct advantages over BIC. First, it is directly related to the underlying data, in this case the distance parameter used in DBSCAN is calculated from the architecture of the Internet and is directly related to size of subnets common to IP networks. Second, the NSR signal method produces models with predictive accuracy equal to those fond using BIC, but with fewer clusters/parameters as shown in Table 3.7. The NSR method exemplifies what Box et al. formulated with the principle of parsimony in modeling as the application of the "smallest possible number of parameters for adequate representation" of the data [40].

Table 3.7 compares the cluster/parameter selection using the NSR signal method to the BIC across the four device datasets. To compute the results each device's average F1 score was calculated against each malware using the GMM for each device. Next, these device-averages were then summed and averaged to calculate an average for the entire dataset. Each dataset-average was then calculated 10 times with the resulting total averages shown in the table.

Overall, BIC finds an average of 2-2.5 more cluster/parameters than the NSR method. The NSR and BIC methods have approximately equal predictive accuracy as shown by the F1 scores across the four datasets. The largest difference is in the SCADA dataset where the BIC-based models resulted in average of approximately 89% more false positives than the NSR signal method on the test data. This led to BIC-based models performing slightly worse in terms of F1 scores on the SCADA data.

Table 3.7: NSR vs. BIC Comparison

Dataset	Avg. Comp. (NSR)	Avg. Comp. (BIC)	Avg F1 (NSR)	Avg. F1 (BIC)
Home	9.897	22.879	0.883	0.893
CSU Lab	8.757	21.060	0.879	0.869
UNSW	7.88	18.136	0.929	0.923
SCADA	4.047	8.023	0.968	0.943

3.4.7 Model Evaluation

Every device from each dataset had its NSR complexity calculated. Using that NSR complexity a GMM was trained to produce a specific model for each device. To evaluate each device model, malware traffic from Dataset 5, was used to see how well the model was able to identify normal traffic from abnormal attack traffic. Each device was modeled with up to 1000 flow examples split into 80% for training, 20% for testing. Devices that had less than 100 flows were not analyzed for behavior. When evaluating the model the test data are assumed to be normal data and the malware was considered abnormal data. Evaluating the model produces four metrics on how the model is performing. These four metrics are:

- true positives (TP): malware traffic correctly identified as malware traffic.
- true negatives (TN): test traffic correctly identified as normal traffic.
- false positives (FP): test traffic incorrectly identified as malware traffic.
- false negatives (FN): malware traffic incorrectly identified as normal traffic.

The confusion matrix for this is shown in Table 3.8.

Table 3.8: Confusion Matrix for Device Traffic

	Predicted: Normal Traffic	**Predicted: Maleware Traffic**
Actual: Normal Traffic	TN	FP
Actual: Malware Traffic	FN	TP

From these four metrics, there are two important factors that are often used. First, precision is the ratio of correct positive predictions to the total predicted positives (Equation 3.13), and second recall is the ratio of correct positive predictions to the total positive examples (Equation 3.14). A balanced method for measuring the efficacy of a prediction model is called the F1 score. The F1 score is the harmonic mean of precision and recall (Equation 3.15).

Precision

$$P = \frac{TP}{TP + FP} \tag{3.13}$$

Recall

$$R = \frac{TP}{TP + FN} \tag{3.14}$$

F1 Score

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{3.15}$$

3.5 Summary

This chapter presented the design overview, data collection, and data analysis of the research. Each of the four device datasets and the malware dataset was described. Next, I presented several definitions of complexity: traffic variance, IP complexity, unique flows, and NSR complexity and how each was calculated. This was followed by discussion on how the number of clusters in the DBSCAN algorithm was tuned using the IP_Spread of the device and how this cluster calculation was used to inform the GMM of how many Gaussians to produce. This chapter continued in

defining a behavior model for a device based on a Gaussian mixture model and how outliers and inliers were calculated in this model. Finally, I described how the device models are evaluated against the malware traffic using F1 scores.

Chapter 4

Research Findings

4.1 Introduction

This chapter documents the results found in this study of IoT devices. It describes results from the four individual datasets, including a complexity analysis for variance, IP complexity, unique flows and the NSR method. Next, results from the NSR-based GMM analysis show the efficacy for tuning the anomaly detection model by complexity. This section is organized by dataset and the order is shown below:

- Home Network
- Lab Network
- University of New South Wales Lab Network
- METEC Scada Network

4.2 Home Dataset Results

The home dataset consisted of the devices found in Section 3.3.1.

Figure 4.1 shows the home devices' variance of traffic calculated over 1000 flows. The variance is calculated as a tuple (bits per second, packets per second) as described in Section 3.4.2. The devices are ordered from left to right with the devices showing with the highest calculated variance on the right.

Figure 4.2 shows the total IP complexity of each device on the home network. This graph is based on a sampling of up to 1000 network flows per device, and shows the complexity of destination IP addresses. The right side of the graph shows the devices that were found to be the most complex as measured by the IP complexity metric discussed in Section 3.4.3.

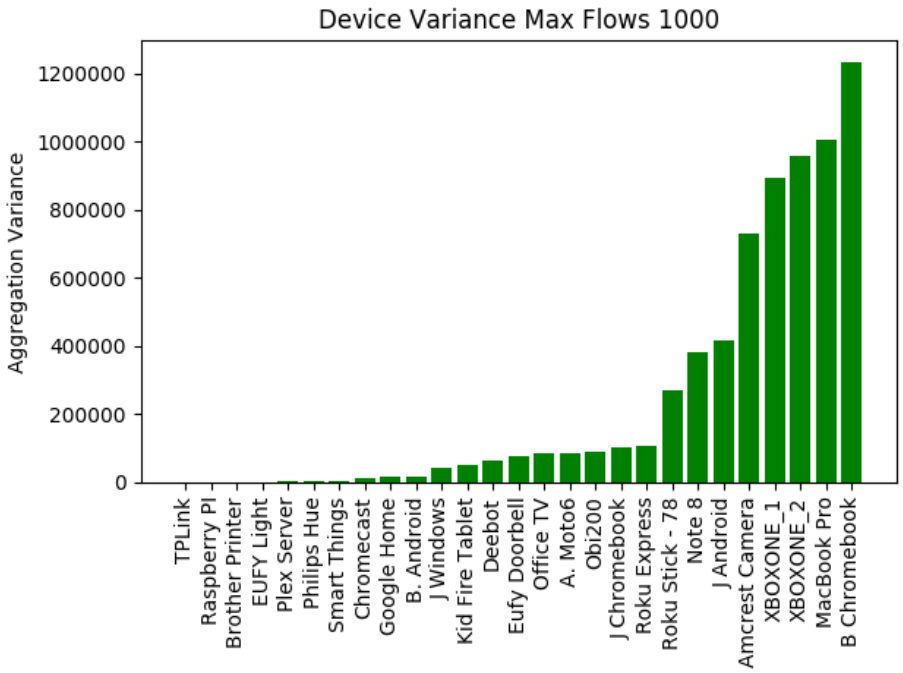


Figure 4.1: Average Device Network Variance

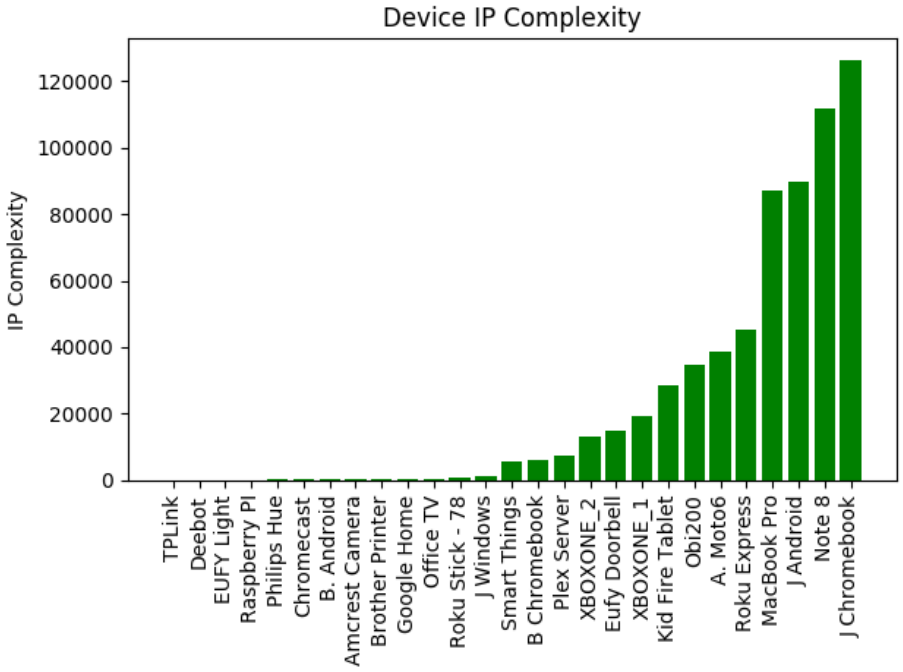


Figure 4.2: IP Device Complexity

Figure 4.3 shows the device complexities based on unique flows. This figure shows devices ranked by the number of unique tuples of network traffic described in 3.1. It represents an aggregation of IP complexity and variance. Figure 4.4 shows the noise to signal (NSR) complexity for each device on the home dataset with the highest complexity devices shown on the right. The line drawn at 8.438 NSR shows the approximate threshold between an IoT device and those that are more general-purpose devices, this value is the average NSR value for the home, lab, and UNSW datasets.

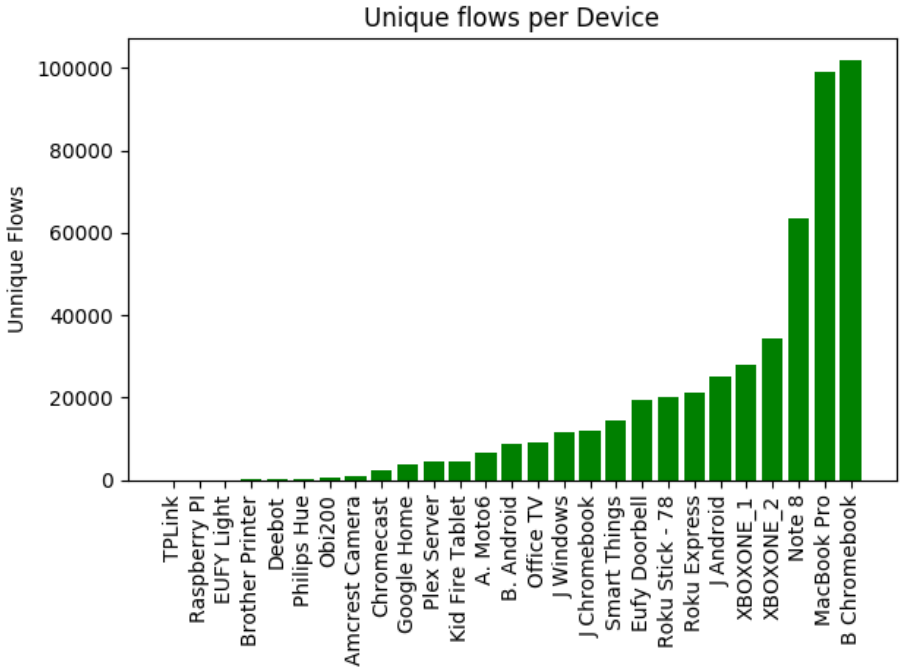


Figure 4.3: Home: Flow Complexity

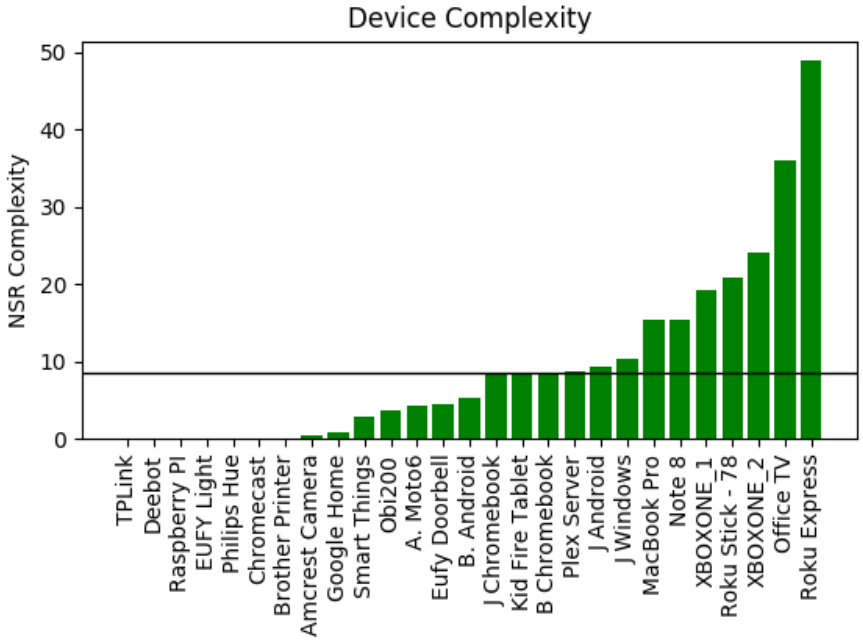


Figure 4.4: Home: NSR Complexity

The following three Figures [4.5, 4.6, 4.7] show the NSR for three devices: the highest NSR, the Roku Express (52.25), an NSR close to the dataset average, the J. Chromebook (9.35), and the lowest NSR, the Eufy light bulb (0). IP addresses are shown in scientific notation. Signals are show as black dots highlighted in green and noise is shown by small yellow circles. These figures show that high complexity devices would typically have more noise than they do signals this can be seen in Figure 4.5 of the Roku Device. Low complexity devices will have a higher ratio of clusters to noise, this can be seen in the Figure 4.7 where there are two cluster and zero noise points.

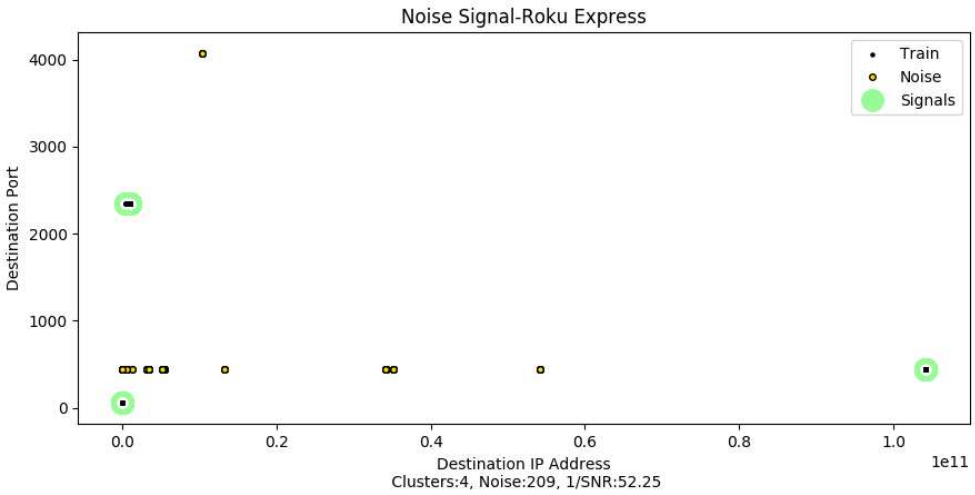


Figure 4.5: Home: Noise to Signal for Roku Express

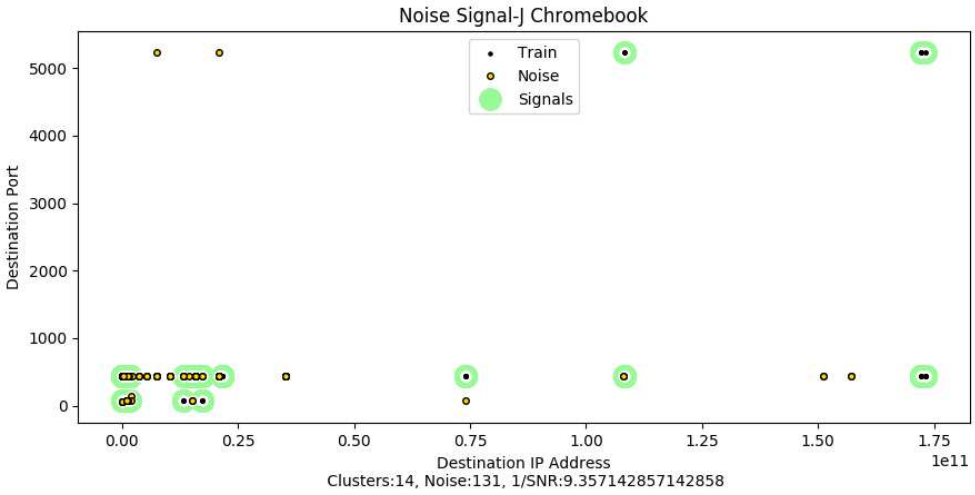


Figure 4.6: Home: Noise to Signal for J. Chromebook

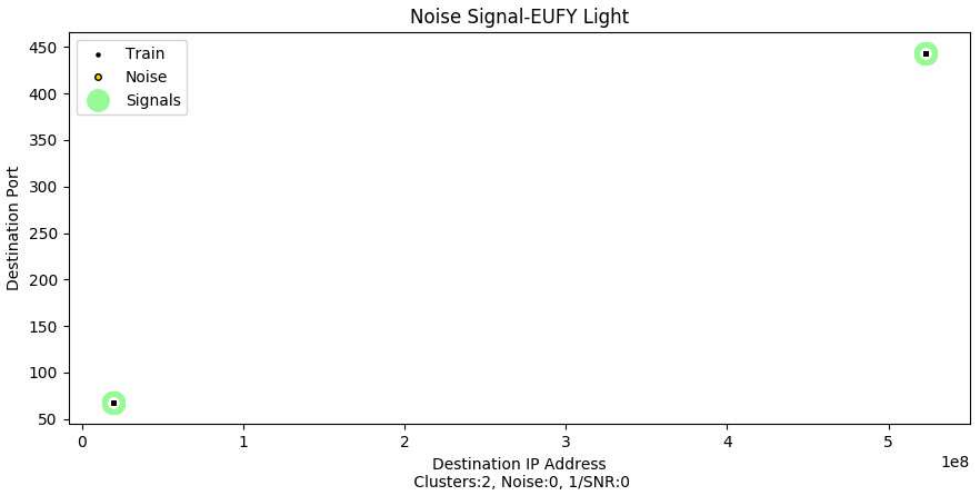


Figure 4.7: Home: Noise to Signal for Eufy Light

Malware Attack Analysis

This section shows the attack analysis results for the three devices in the home dataset, that span the range of NSR complexity scores: the Roku Express, J. Chromebook, and the Eufy light bulb. Each device is shown against two types of malware traffic from Table 3.6: C&C, and C&C File Download traffic. These attacks were chosen as they represent the largest variance in both destination IP and destination port. Each figure in this section shows the standardized destination port and standardized destination address. Normal test data is shown as black dots. The attack data is shown in red dots. Light blue ellipsoids show *estimates* of the confidence of inlier behavior learned by the GMM. These ellipsoids were generated for the figures in a generalized manner to show details in each figure and do not perfectly align with the actual probablity threshold of each device model. The results of all of the attacks against all of the home devices can be found in Appendix B.

The following Figures [4.8, 4.9] show the Gaussian probability surfaces for the Roku Express device against the C&C malware traffic and against the C&C file download traffic. This is a highly complex device as measured by the NSR method.

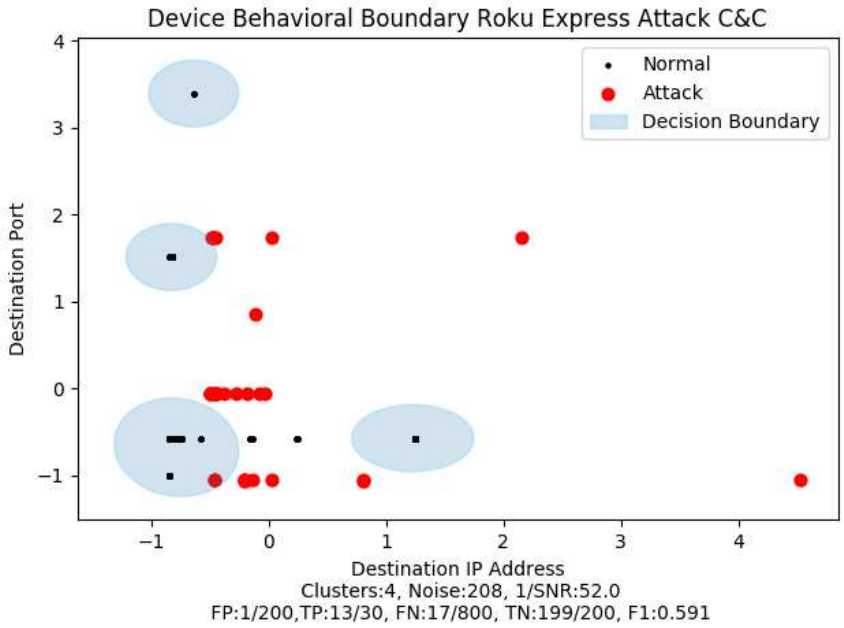


Figure 4.8: Home: C&C Attack And Gaussian Boundary Roku Express

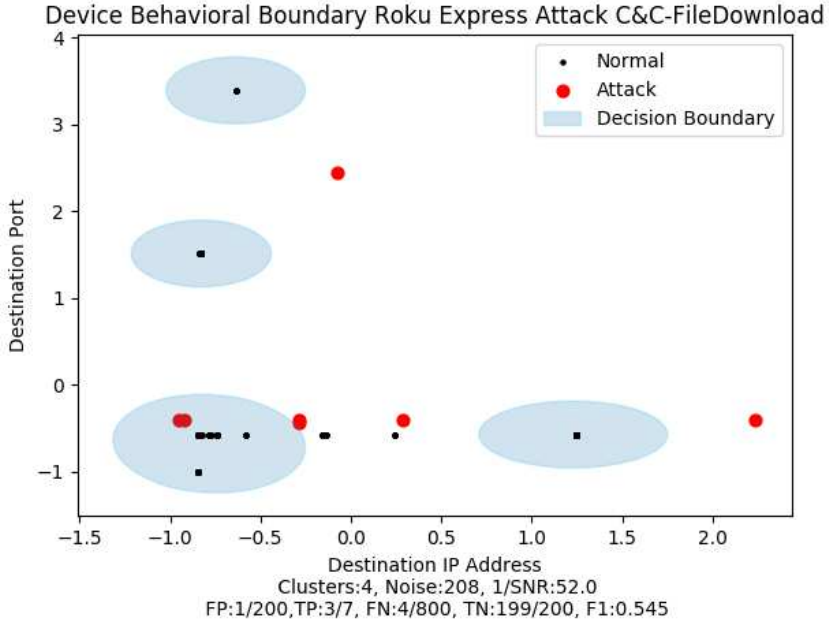


Figure 4.9: Home: C&C File Download Malware And Gaussian Boundary Roku Express

Figures 4.10 and 4.11 show the Gaussian mixture boundaries for the J Chromebook device, an average NSR complexity device against the C&C and the C&C Filedownload traffic. The

boundaries show a higher probabilistic fit of the normal traffic than the boundaries shown in Figures 4.8 and 4.9.

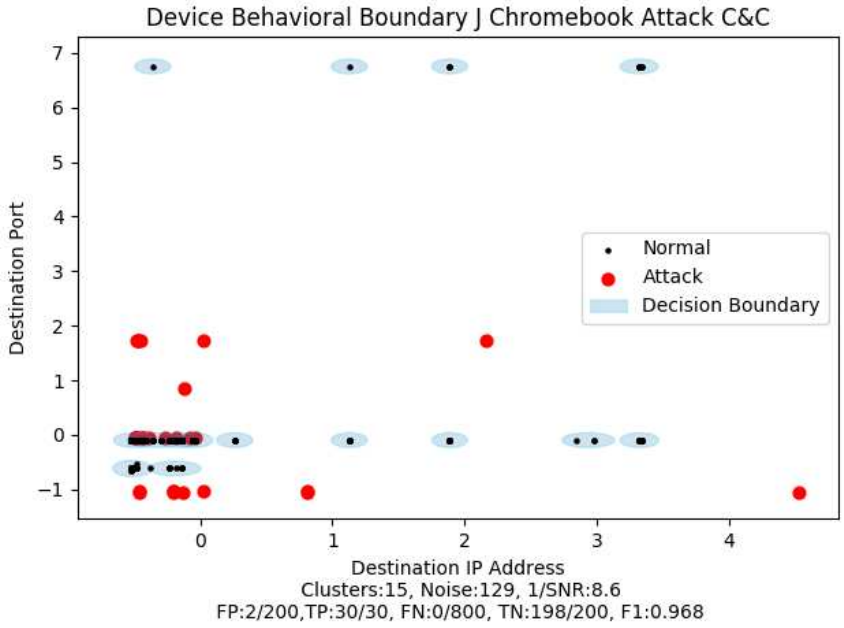


Figure 4.10: Home: C&C Attack And Gaussian Boundary J Chromebook

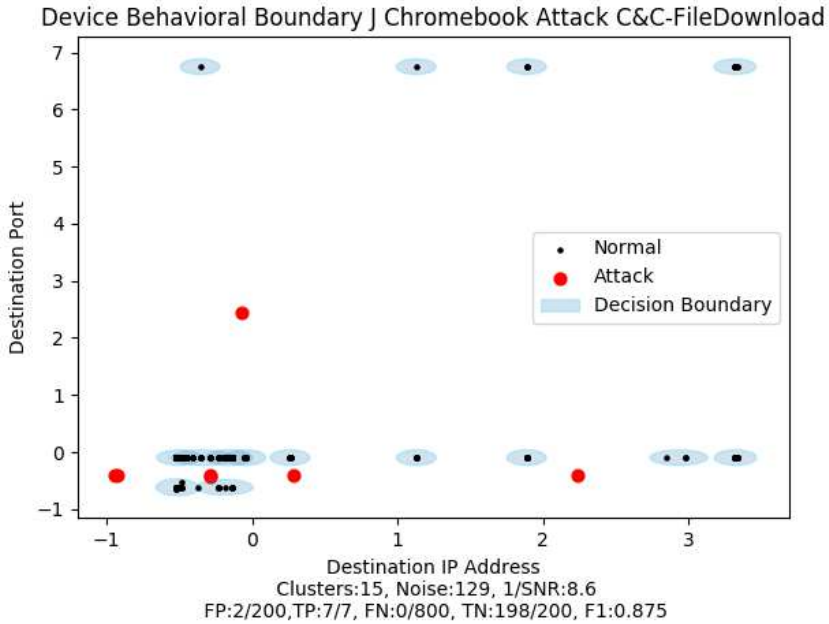


Figure 4.11: Home: C&C File Download Attack And Gaussian Boundary J Chromebook

Figures 4.12, and 4.13 show the Gaussian boundaries for the Eufy light bulb device, a low NSR complexity device, against the C&C and the C&C Filedownload traffic. The boundaries in these figures are much more tightly coupled to the normal traffic than in the previous two devices.

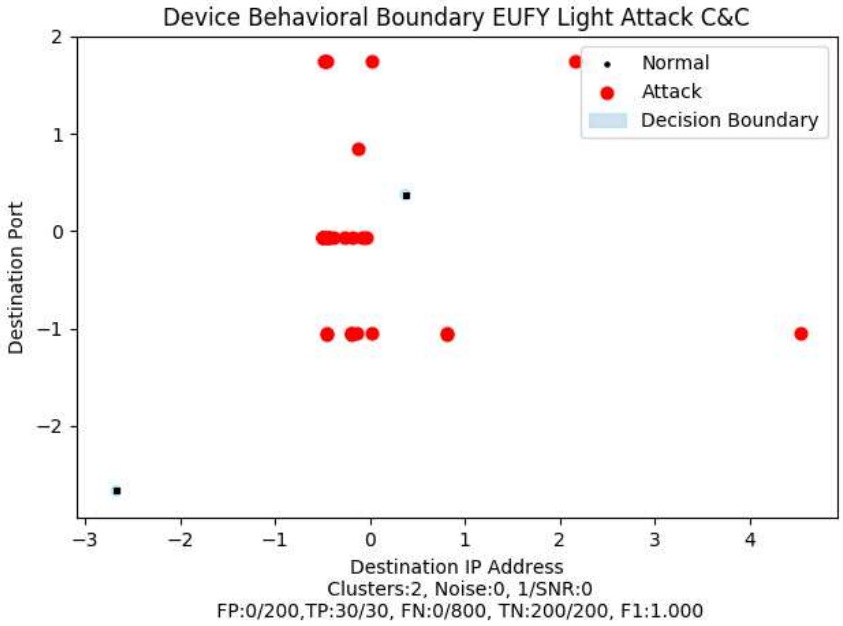


Figure 4.12: Home: C&C Attack And Gaussian Boundary Eufy Light

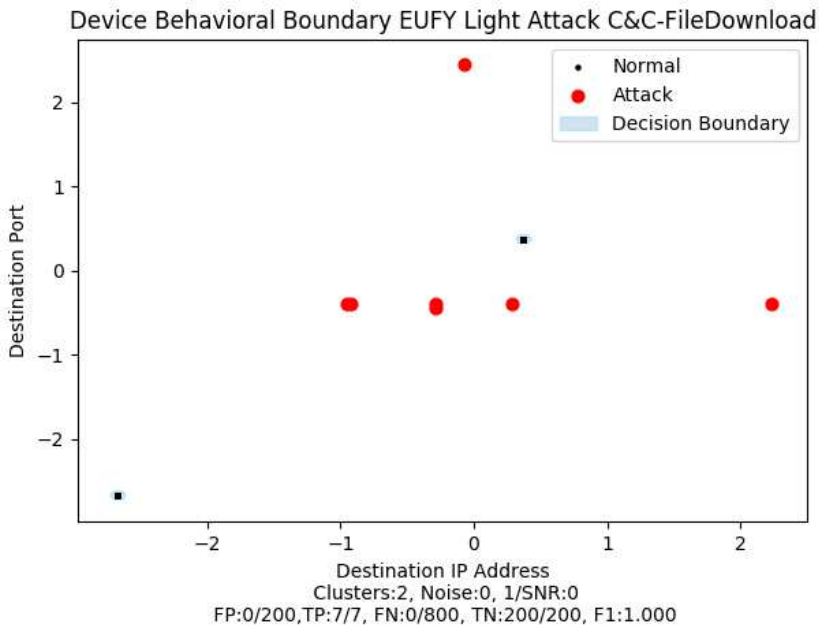


Figure 4.13: Home: C&C File Download Attack And Gaussian Boundary Eufy Light

Figure 4.14 shows the F1 score for every device in the home dataset modeled using the complexity based GMM on non-standardized data. The red line drawn at a F1=0.6 marks the threshold which generally showed that the majority devices above this line are single-purpose devices and those below this line are general purpose-devices. Figure 4.15 shows the average F1 score using the complexity based GMM on standardized data. Models built using standardized data generally performed better than those using non-standardized data, however, no line is drawn on this figure as there is no obvious correlation between the F1 score and the complexity once the data were standardized.

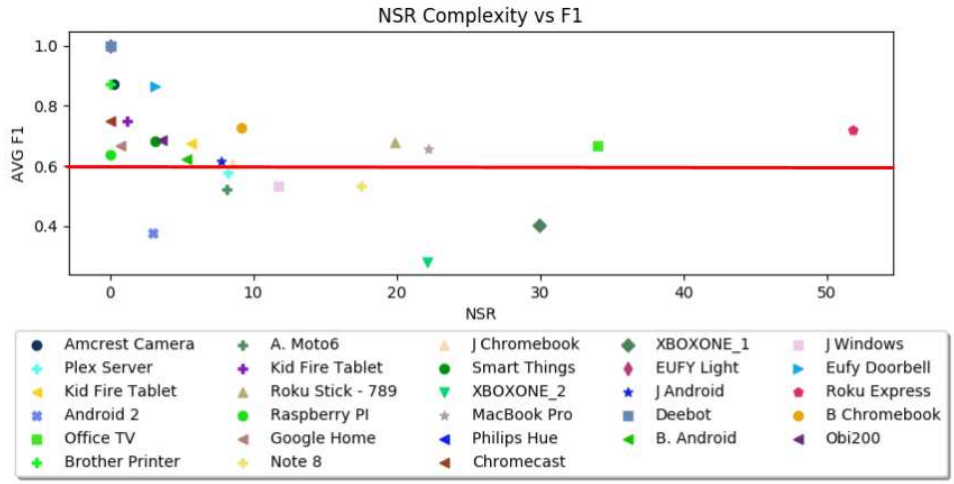


Figure 4.14: Home: NSR Vs. F1

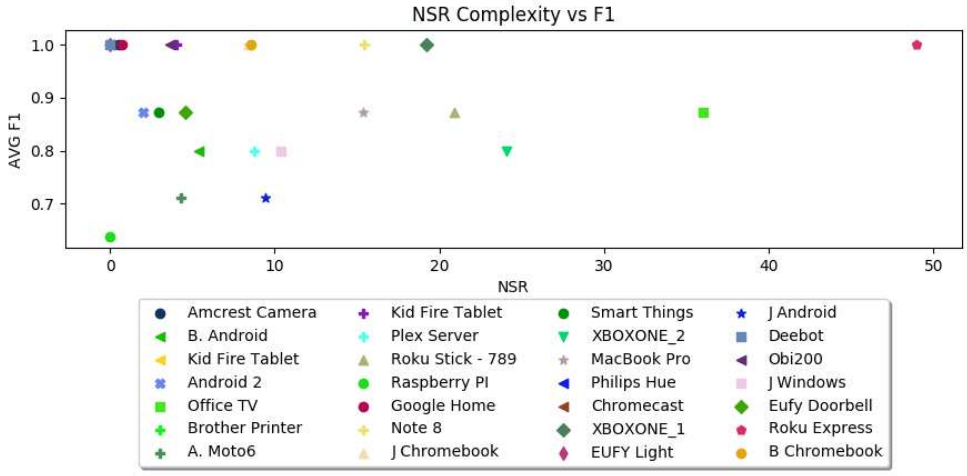


Figure 4.15: Home: NSR Vs. F1 Normalized

4.3 Lab Dataset Results

This section shows the results from the Colorado State University Lab dataset summarized in Section 3.3.2. Figure 4.16 shows the variance of lab devices as calculated in Section 3.4.2. Devices are sorted by variance from left to right, and the figure was generated using at most 1000 flows per device.

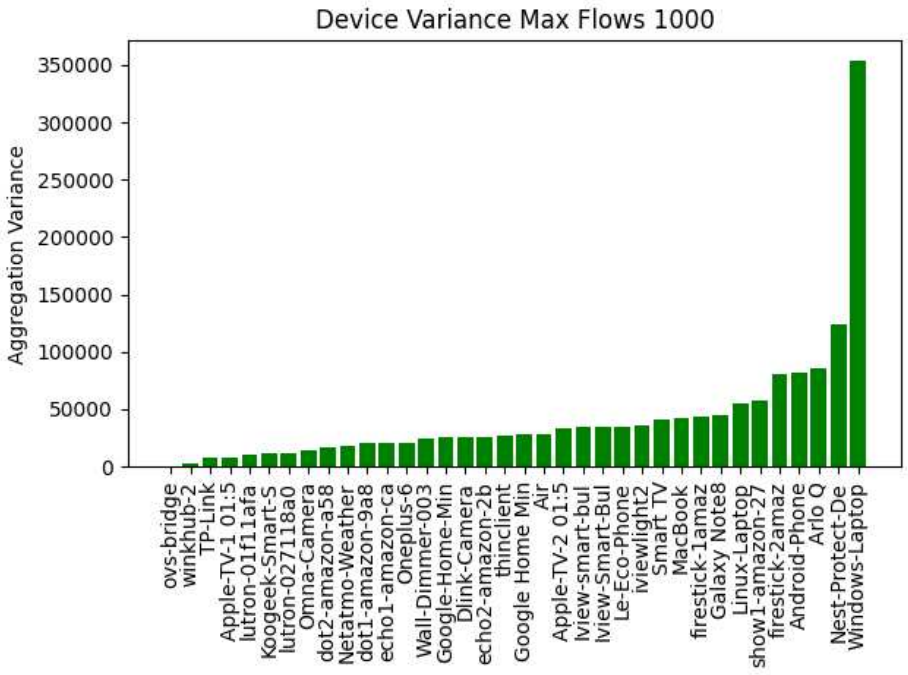


Figure 4.16: Lab: Average Device Network Variance

Figure 4.17 shows the IP complexity as calculated in Section 3.4.3 for lab devices. Devices are sorted by IP complexity from left to right.

Figure 4.18 shows the flow complexity of the lab devices ranked ascending from left to right.

Figure 4.19 shows the noise to signal (NSR) complexity for each device on the lab dataset with the highest complexity devices shown on the right. The line drawn at 8.438 NSR shows the approximate threshold between an IoT device and what is a more general-purpose devices, this value is the average NSR value for the home, lab, and UNSW datasets.

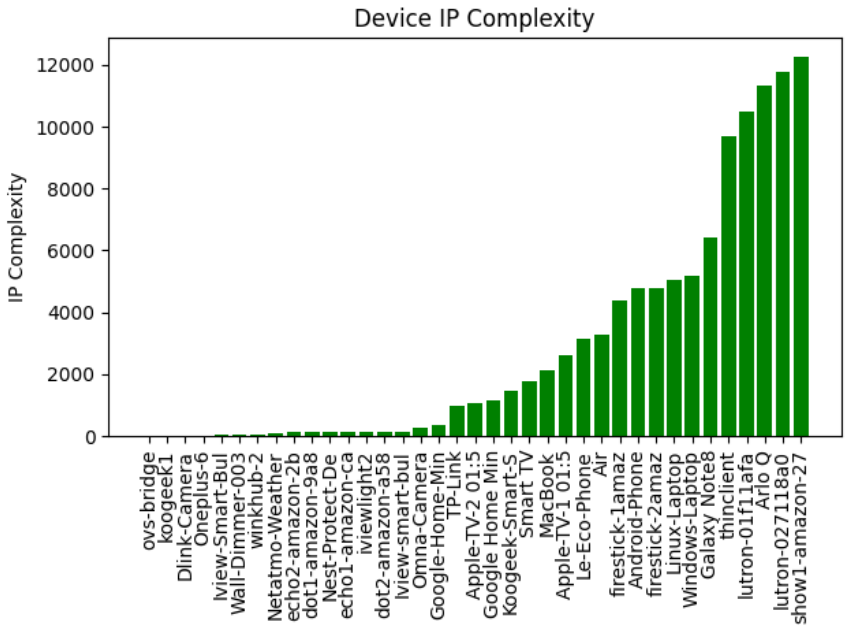


Figure 4.17: Lab: IP Device Complexity

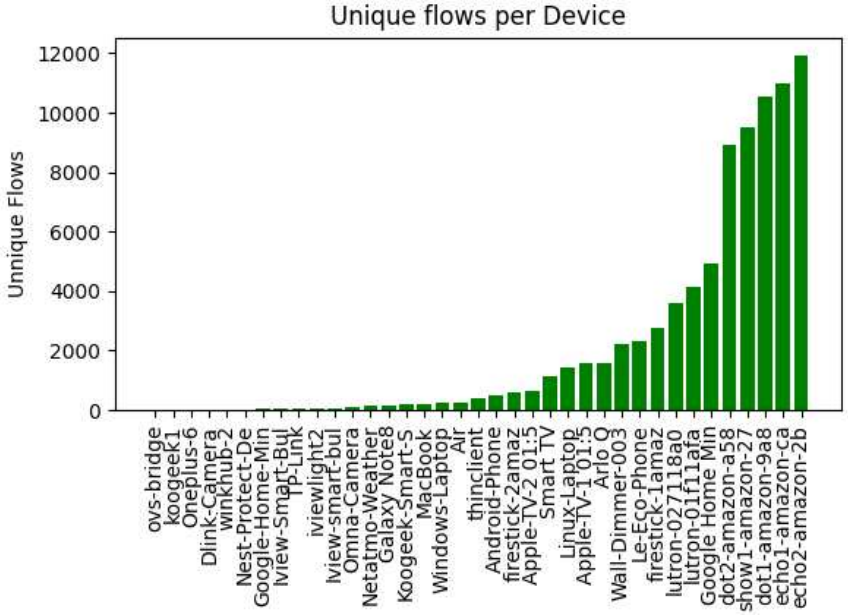


Figure 4.18: Lab: Flow Complexity

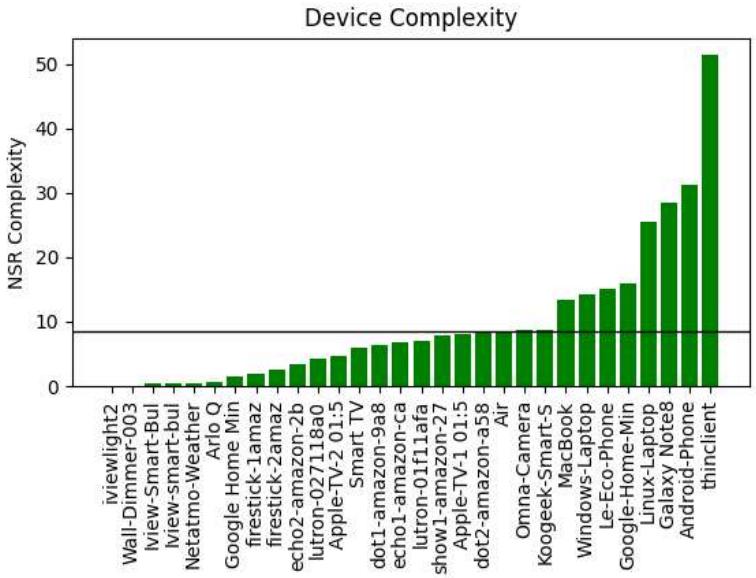


Figure 4.19: Lab: NSR Complexity

Figures [4.20, 4.21, 4.22] show the NSR for a high, average, and low device: the Note8, Apple TV and the IviewLight light bulb respectively. The IP address is shown in scientific notation for each figure. Signals are show as black dots highlighted in green and noise is shown by small yellow circles. These figure show that high complexity devices would typically have more noise than they do signals this can be seen in Figure 4.20 of the Note8 Device. Low complexity devices will have a higher ratio of clusters to noise, this can be seen in the Figure 4.22 where there are two cluster and zero noise points.

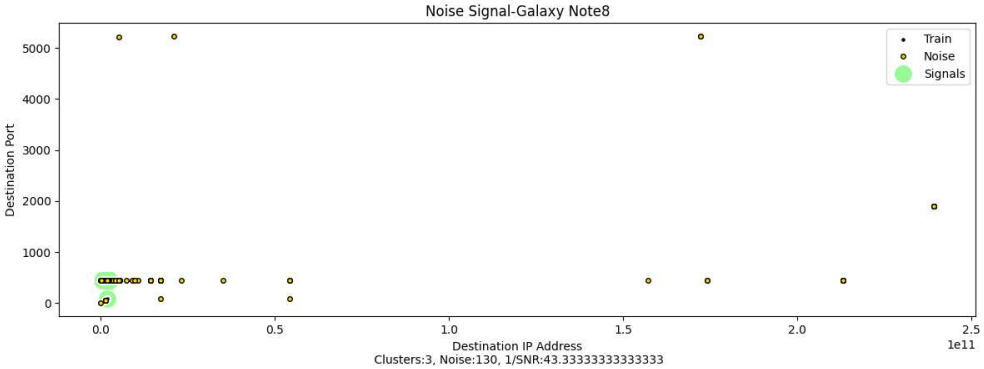


Figure 4.20: Lab: Noise to Signal for Note 8

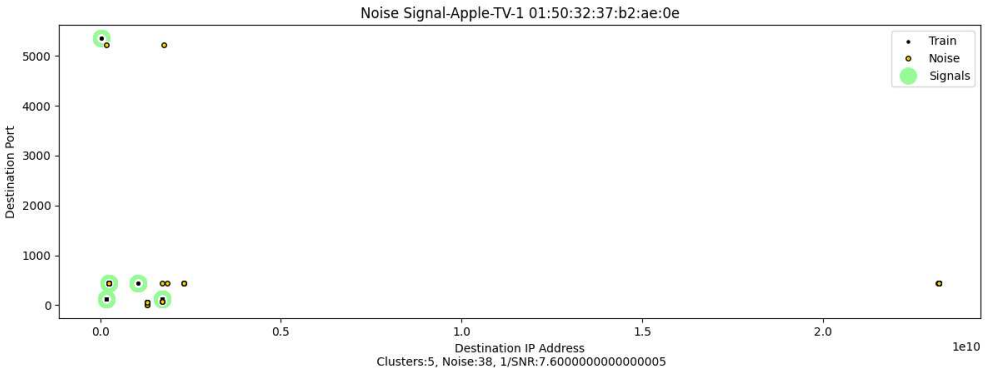


Figure 4.21: Lab: Noise to Signal for Apple TV

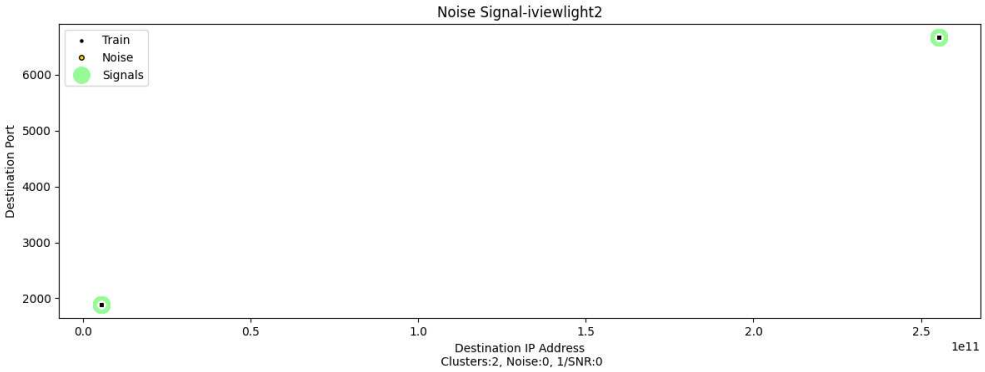


Figure 4.22: Lab: Noise to Signal for IviewLight 2

Malware Attack Results

These results show how the attacks listed in Table 3.6 are detected as abnormal traffic compared normal test traffic in the lab device models. I show a high complexity device (Note8), a device of average complexity (Apple TV), and a low complexity device (Iview Light Bulb). Each device is shown against two types of malware traffic from Table 3.6: C&C, and C&C File Download traffic. These attacks were chosen as they represent the largest variance in both destination IP and destination port. Each figure in this section shows the standardized destination port and standardized destination address. Normal test data is shown as black dots. The attack data is shown in red dots. Light blue ellipsoids show *estimates* of the inlier probability threshold learned by the GMM. These

ellipsoids were generated for the figures in a generalized manner to show details in each figure and do not perfectly align with the actual inlier probablity threshold of each device model. The results of all of the attacks against all of the home devices can be found in Appendix C.

Figures 4.23, and 4.24 show the Gaussian mixture boundaries for the Note8, a high NSR complexity device against the C&C and the C&C Filedownload traffic. The figure shows Gaussian boundaries that do not form precise boundaries over the normal training traffic, leading to poorer performance.

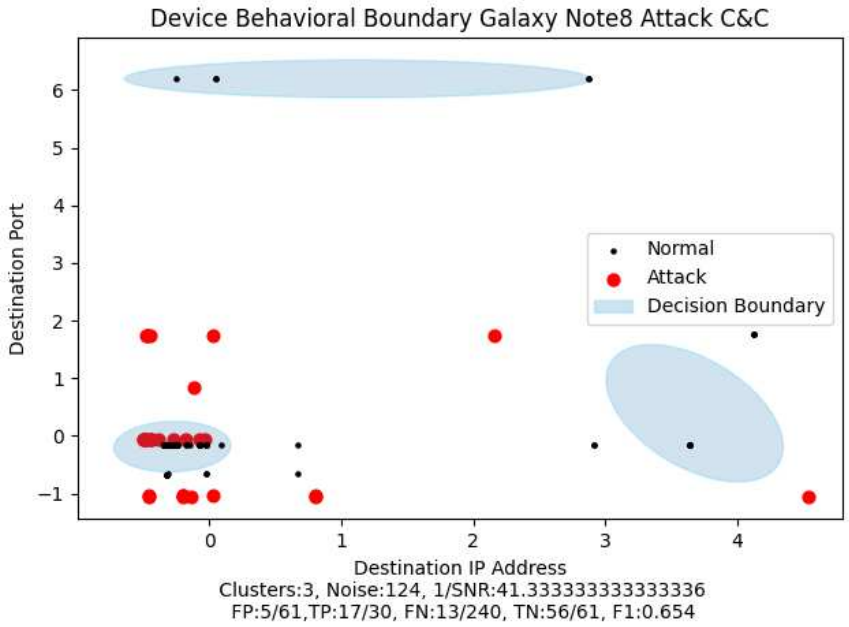


Figure 4.23: Lab: C&C Attack And Gaussian Boundary Note 8

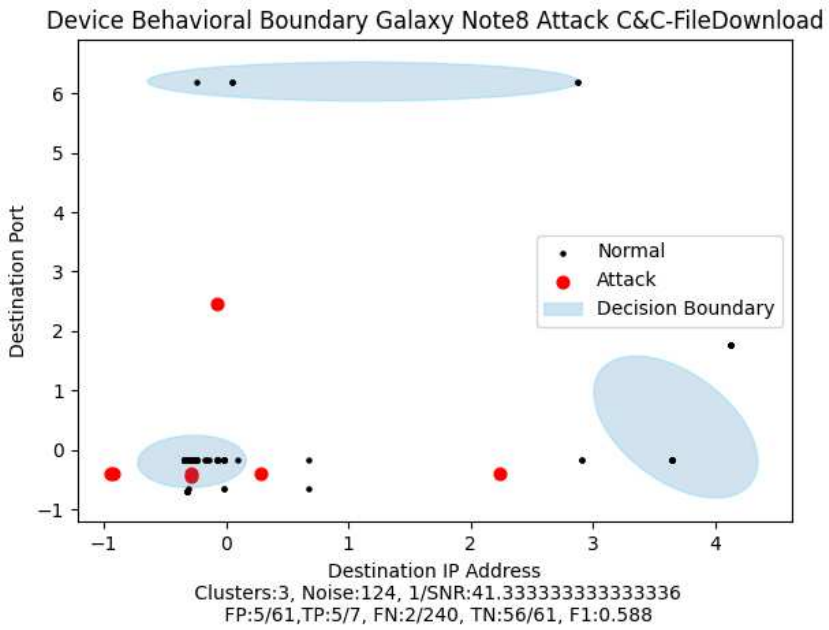


Figure 4.24: Lab: C&C Filedownload Attack And Gaussian Boundary Note 8

Figures 4.25, and 4.26 show the Gaussian mixture boundaries for the Apple TV, an average NSR complexity device against the C&C and the C&C Filedownload traffic. The inlier probablity threshold represented by the ellipses shown in these figures show smaller more precise boundaries for this average complexity device.

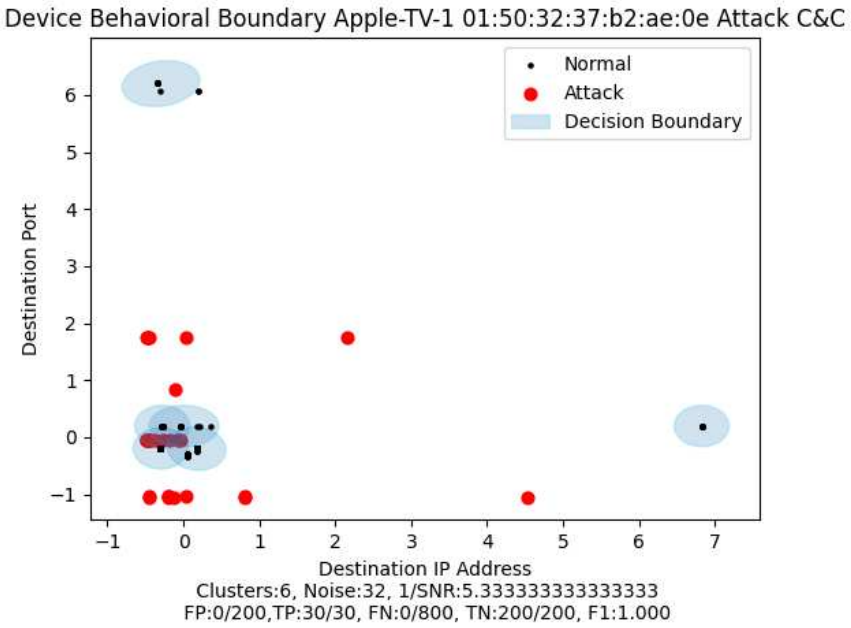


Figure 4.25: Lab: C&C Attack And Gaussian Boundary Apple TV

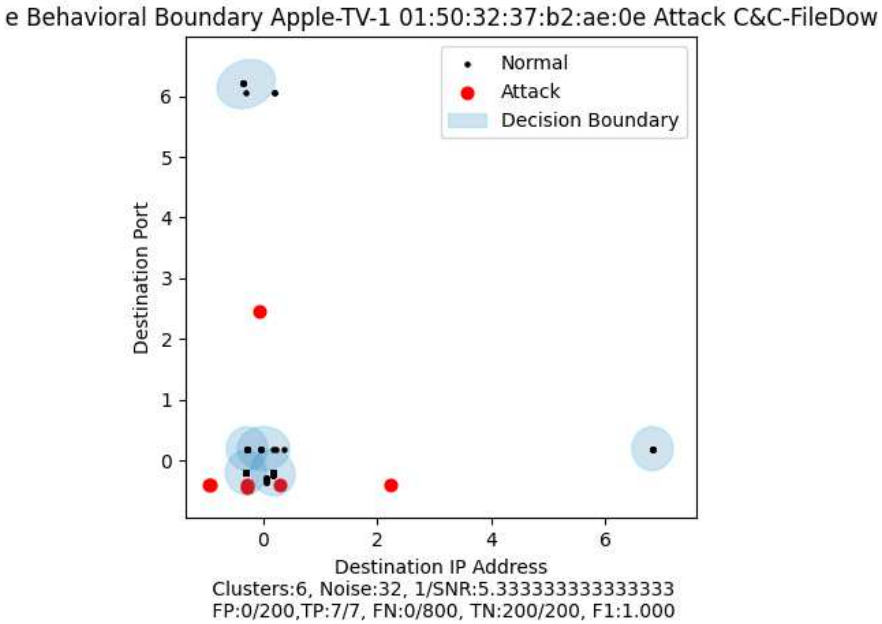


Figure 4.26: Lab: C&C File Download Attack And Gaussian Boundary Apple TV

Figures 4.27, and 4.28 show the Gaussian mixture boundaries for the Iview SmartBulb, an low NSR complexity device against the C&C and the C&C Filedownload traffic. This device is

the lowest complexity of the dataset. The Guassian boundaries shown in the figure reflect a more precise inlier probablity threshold , due to a smaller standard deviation of the learned behavior. This model misclassified one example of each of the attack traffic types, giving a false positive rate of 0.05% and an F1 scores of 0.984 and 0.933 respectively.

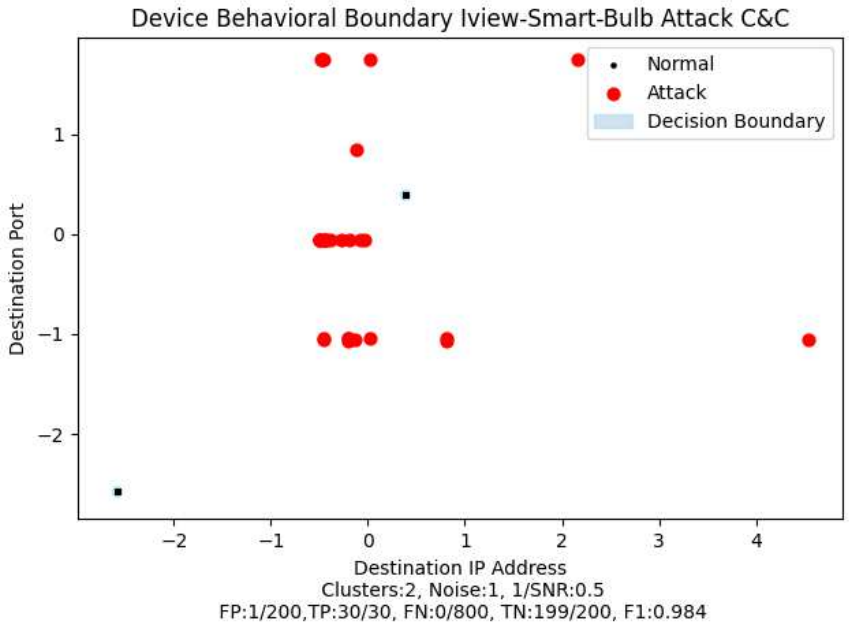


Figure 4.27: Lab: C&C Attack And Gaussian Boundary IviewLight 2

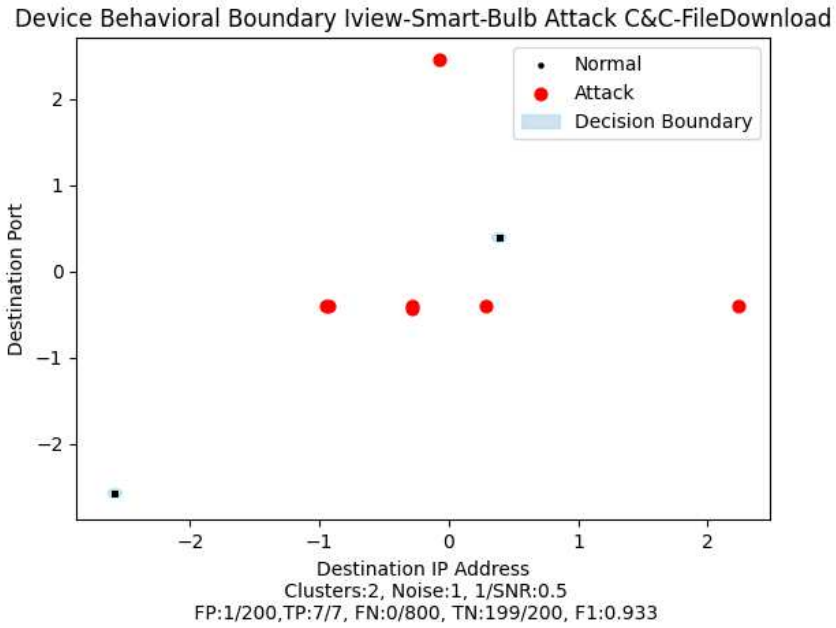


Figure 4.28: Lab: C&C File Download Attack And Gaussian Boundary IviewLight 2

Figure 4.29 shows the F1 score for every device in the lab dataset modeled using the complexity based GMM on non-standardized data. The red line drawn at a F1=0.6 marks the threshold where the majority devices above this line are IoT devices and those below this line are general purpose-devices. Figure 4.30 shows the average F1 score using the complexity based GMM on standardized data. Models built using standardized data generally performed better than those using non-standardized data, however, no line is drawn on this figure as there is a decreased correlation between the F1 score and the complexity once the data were standardized.

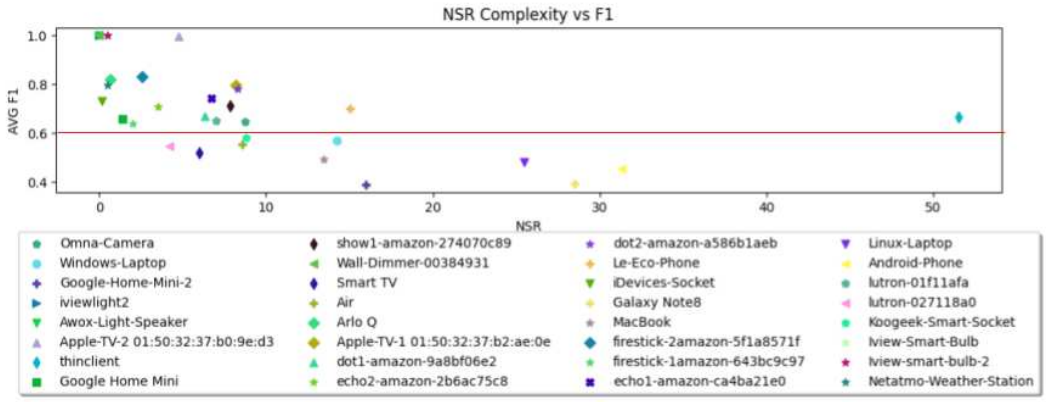


Figure 4.29: Lab: NSR Vs. F1 Non-Normalized

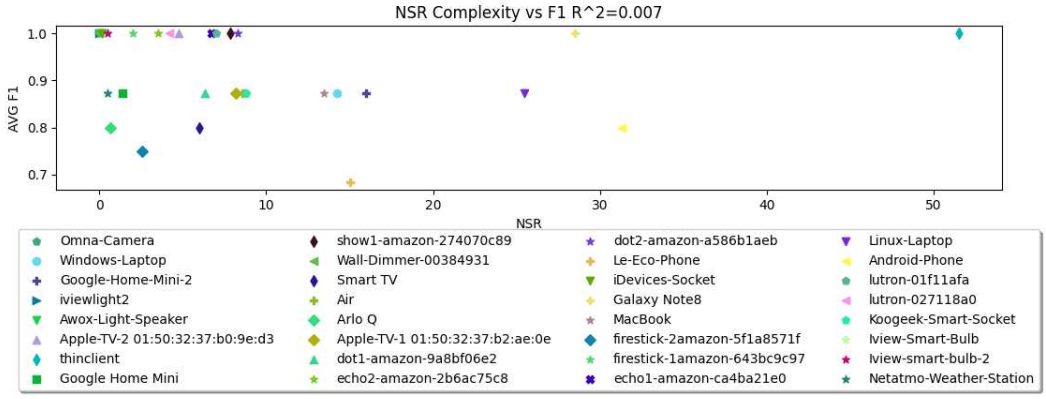


Figure 4.30: Lab: NSR Vs. F1 Normalized

4.4 UNSW Dataset Results

This section shows the results for the University of New South Wales dataset. Figure 4.31 shows the variance of UNSW devices as calculated in Section 3.4.2. Devices are sorted by variance from left to right and the figure was generated using at most 1000 flows.

Figure 4.32 shows the IP complexity as calculated in Section 3.4.3 for UNSW devices. Devices are sorted by IP complexity from left to right, and was generated using at most 1000 flows for each device.

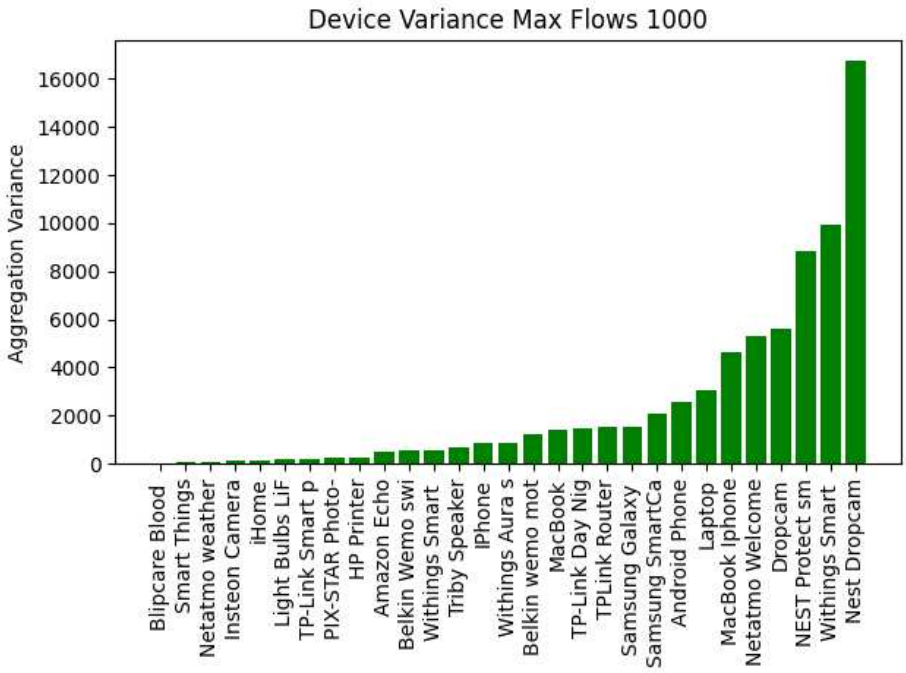


Figure 4.31: UNSW: Average Device Network Variance

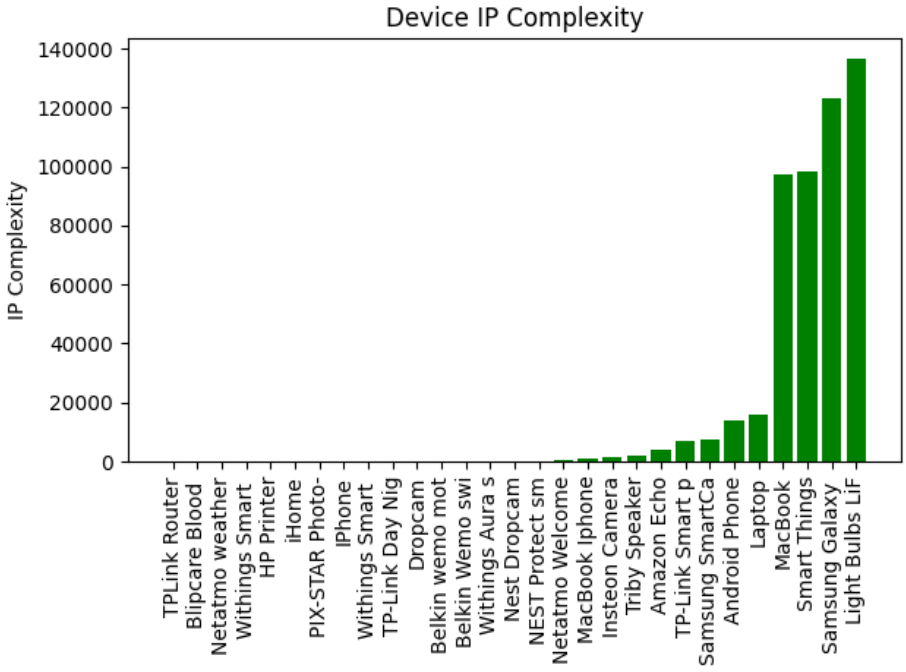


Figure 4.32: UNSW: IP Device Complexity

Figure 4.33 shows the unique flow complexity for each device in the UNSW dataset using at most 1000 flows per device.

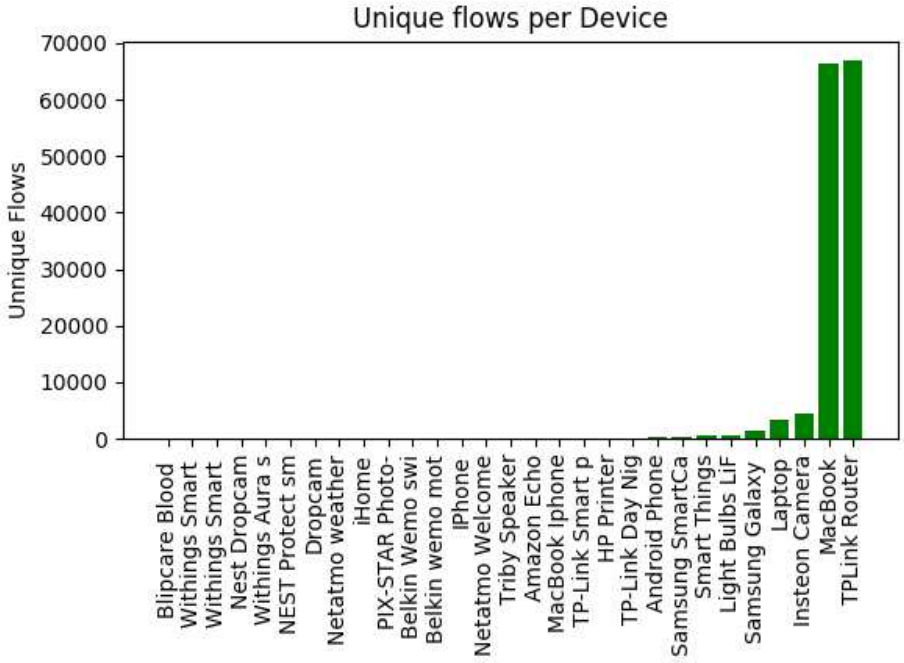


Figure 4.33: UNSW: Flow Complexity

Figure 4.34 shows the noise to signal (NSR) complexity for each device on the UNSW dataset with the highest complexity devices shown on the right. The line drawn at 8.438 NSR shows the approximate threshold between an IoT device and what is a more general-purpose devices, this value is the average NSR value for the home, lab, and UNSW datasets.

Figures [4.35, 4.36, 4.37] show the NSR for a high complexity device, an average complexity device, and a low complexity device. These are the Apple iPhone, TPLink Router, and the Samsung SmartCam respectively. Figure 4.35 shows a single signal/cluster. Figure 4.36 shows 12 signals/clusters some are elongated along the same destination IP address with many varied ports. Figure 4.37 shows four signals/clusters.

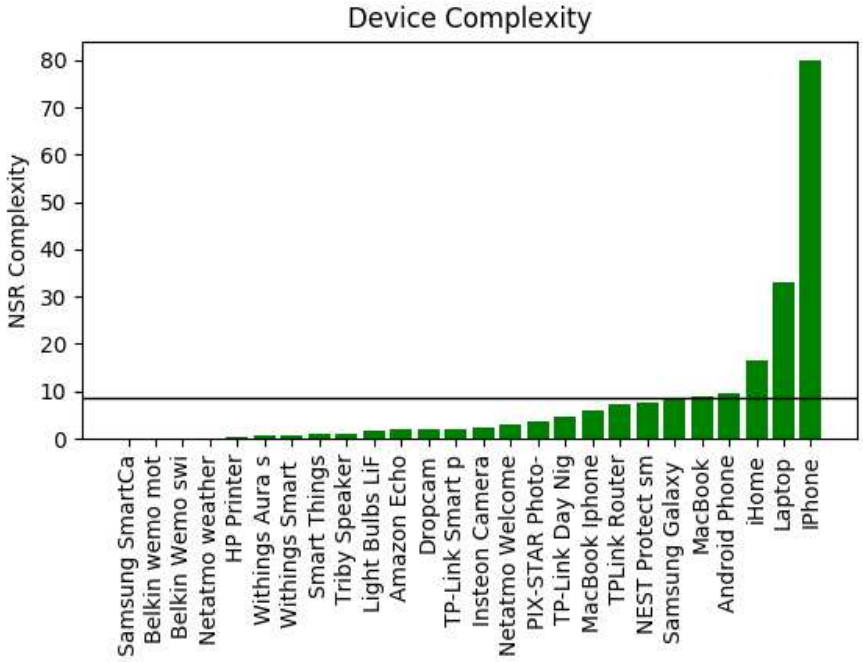


Figure 4.34: UNSW: NSR Complexity

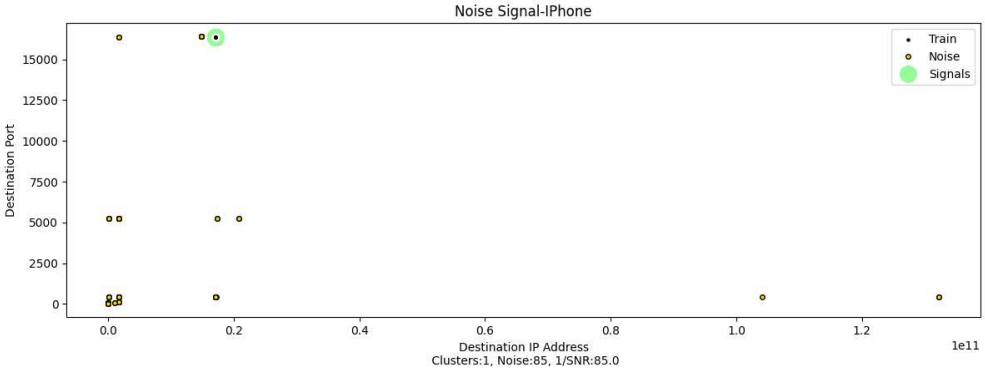


Figure 4.35: UNSW: Noise to Signal for iPhone

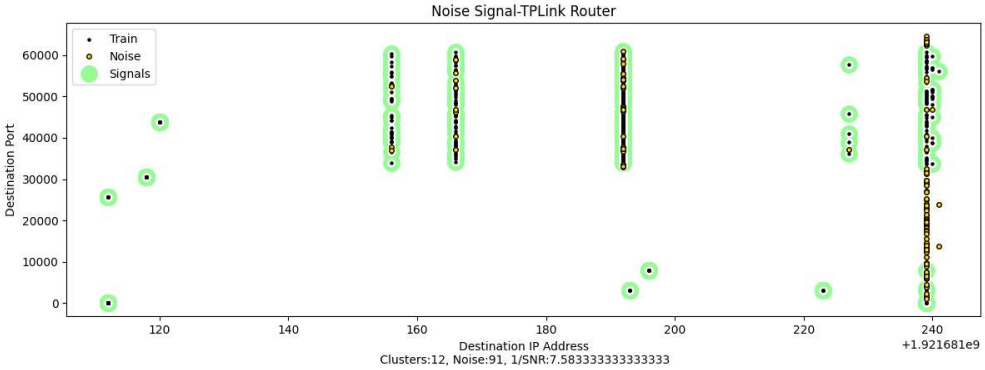


Figure 4.36: UNSW: Noise to Signal for TPLink Router

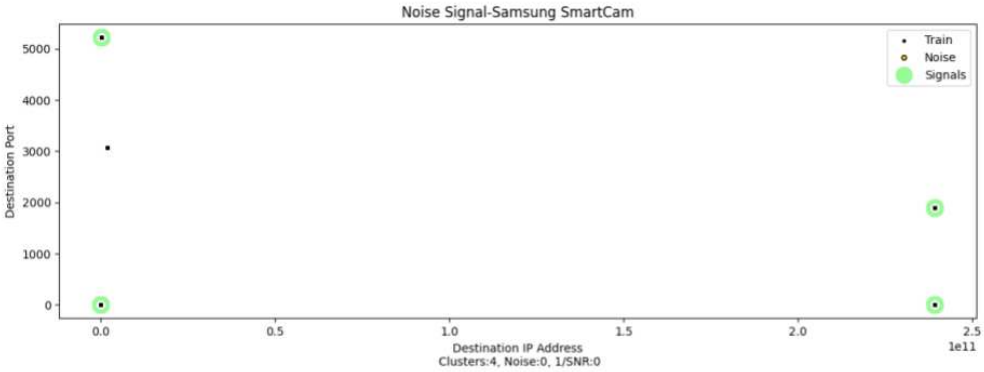


Figure 4.37: UNSW: Noise to Signal for SmartCam

Malware Attack Results

These results show how the attacks listed in Table 3.6 are detected as abnormal compared to the UNSW devices models. I show the most complex device (iPhone), a device of average complexity (TPLink), and the lowest complexity device (Samsung SmartCam). Each device is shown against two types of malware traffic from Table 3.6: C&C, and C&C File Download traffic. These attacks were chosen as they represent the largest variance in both destination IP and destination port. Each figure in this section shows the standardized destination port and standardized destination address. Normal test data is shown as black dots. The attack data is shown in red dots. Light blue ellipsoids show *estimates* of the inlier probablity threshold learned by the GMM. These ellipsoids were generated for the figures in a generalized manner to show details in each figure and do not perfectly align with the actual inlier probablity threshold of each device model. The results of all of the attacks against all of the home devices can be found in Appendix D.

Figures 4.38, 4.39 show the Gaussian mixture boundaries for the iPhone, an high NSR complexity device against the C&C and the C&C Filedownload traffic. These figures show only a single ellipsoid reflecting the sole inlier probablity threshold of the GMM.

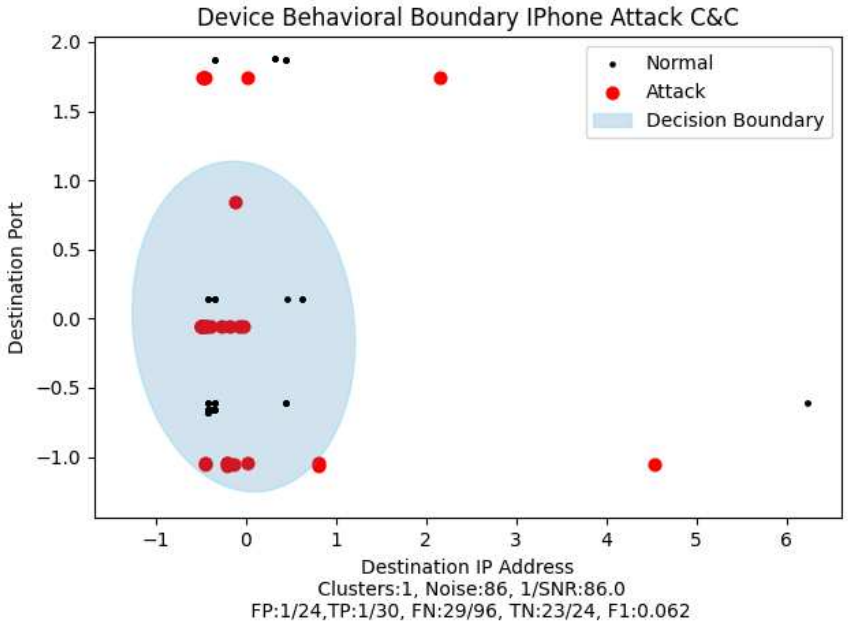


Figure 4.38: UNSW: C&C Attack And Gaussian Boundary iPhone

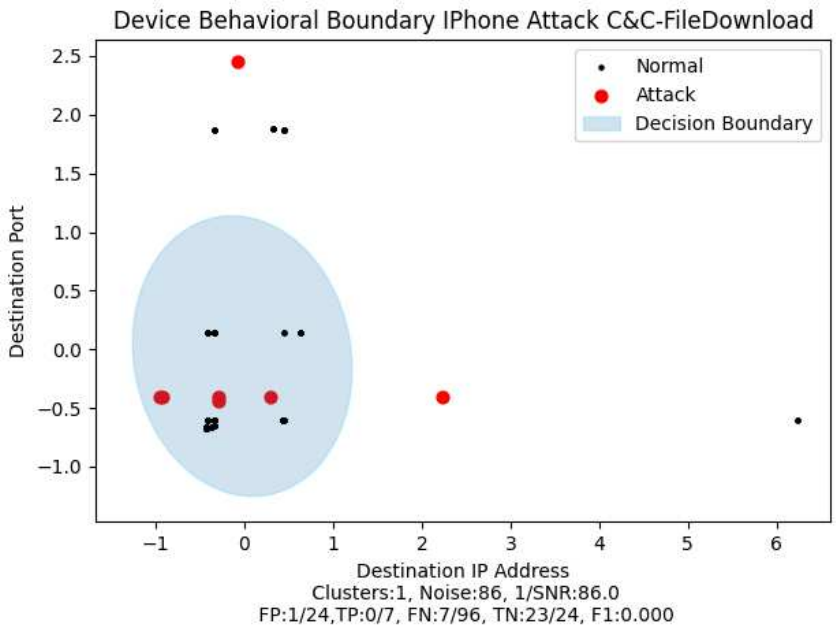


Figure 4.39: UNSW: C&C File Download Attack And Gaussian Boundary iPhone

Figures 4.40 and 4.41 show the Gaussian mixture boundaries for the TPLink Router, an average NSR complexity device against the C&C and the C&C Filedownload traffic. These figures show 13 ellipses approximately lining up with the Gaussians found by the GMM. Several of these ellipsoids show elongation along traffic with a common destination IP address and across a wide range of destination ports.

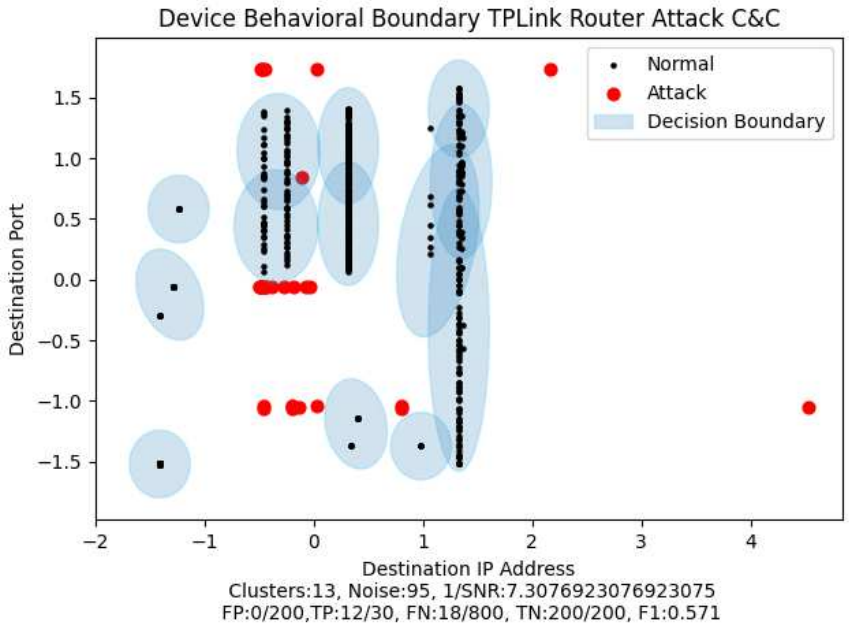


Figure 4.40: UNSW: C&C Attack And Gaussian Boundary TPLink Router

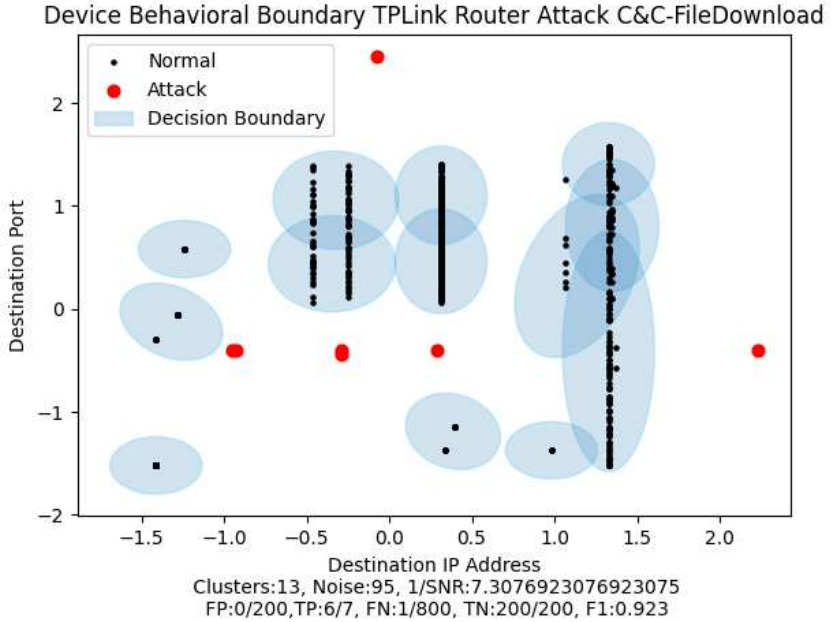


Figure 4.41: UNSW: C&C File Download Attack And Gaussian Boundary TPLink Router

Figures 4.42 and 4.43 show the Gaussian mixture boundaries for the Samsung Smartcam, a low NSR complexity device against the C&C and the C&C Filedownload traffic. These figures show

four ellipsoids reflecting the four Gaussians of the GMM model. One point is not encompassed by an ellipsoid, despite the device model having a perfect F1 score. This shows an error in the figure where the ellipsoid does not line up perfectly with the calculated model.

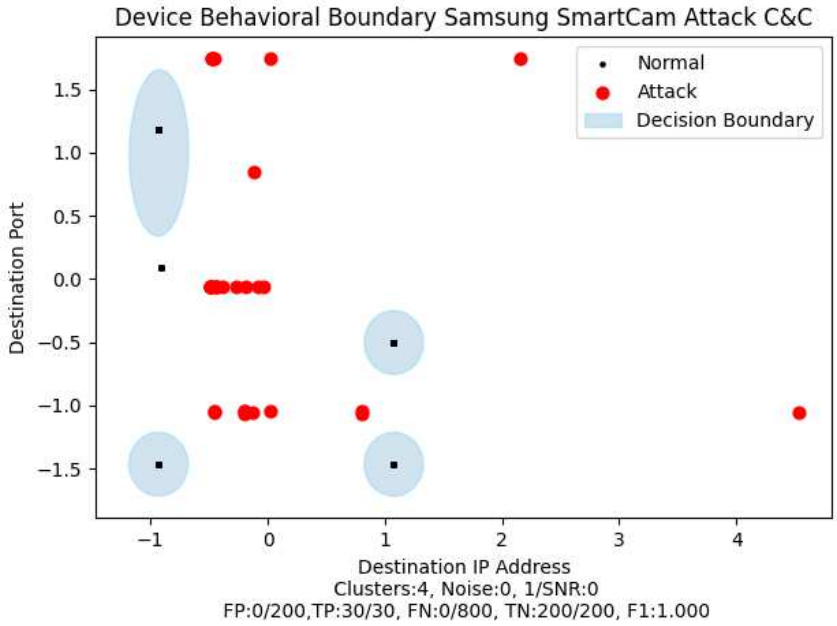


Figure 4.42: UNSW: C&C Attack And Gaussian Boundary Samsung SmartCam

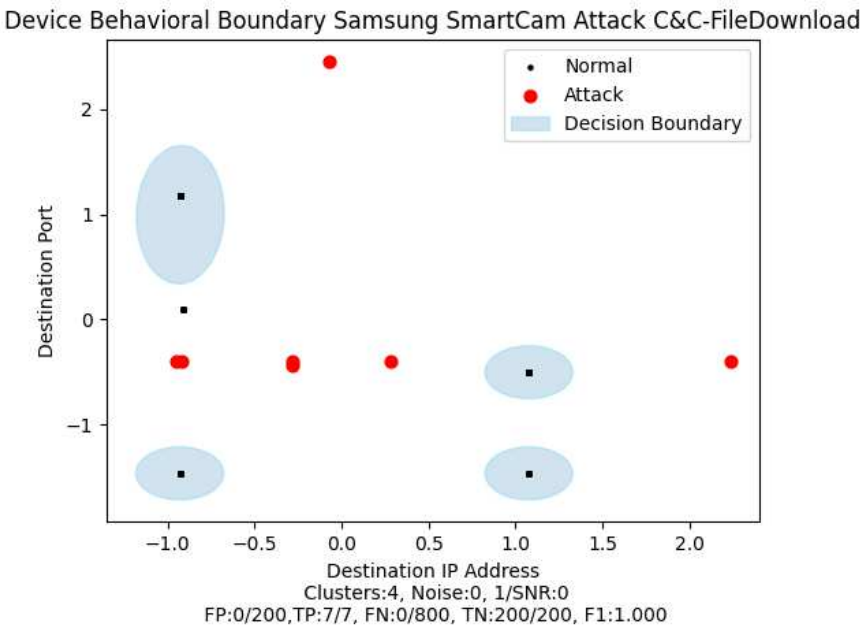


Figure 4.43: UNSW: C&C File Download Attack And Gaussian Boundary Samsung Smartcam

Figure 4.44 shows the F1 score for every device in the UNSW dataset modeled using the complexity based GMM on non-standardized data. The red line drawn at a F1=0.6 marks the threshold where the majority devices above this line are IoT devices and those below this line are general purpose-devices. Figure 4.45 shows the average F1 score using the complexity based GMM on standardized data. Models built using standardized data generally performed better than those using non-standardized data, however, no line is drawn on this figure as there is a decrease in the correlation between the F1 score and the complexity once the data were standardized.

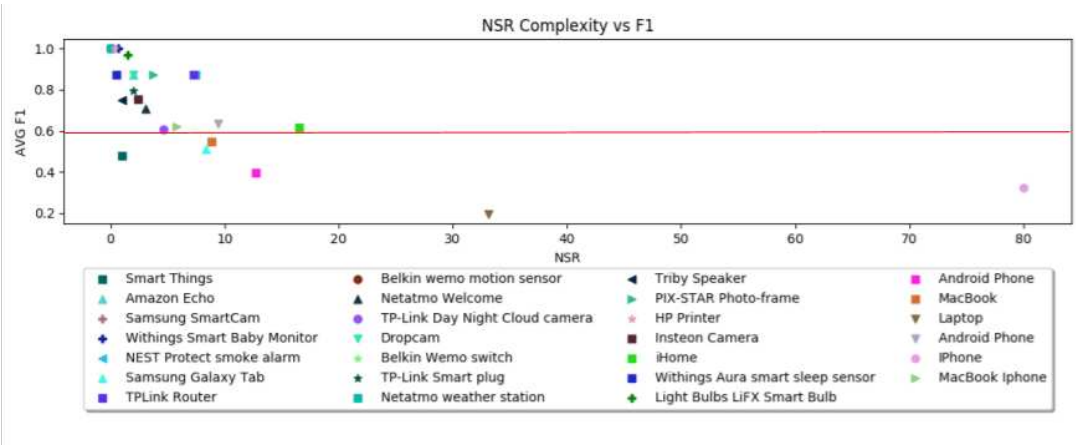


Figure 4.44: UNSW: NSR Vs. F1

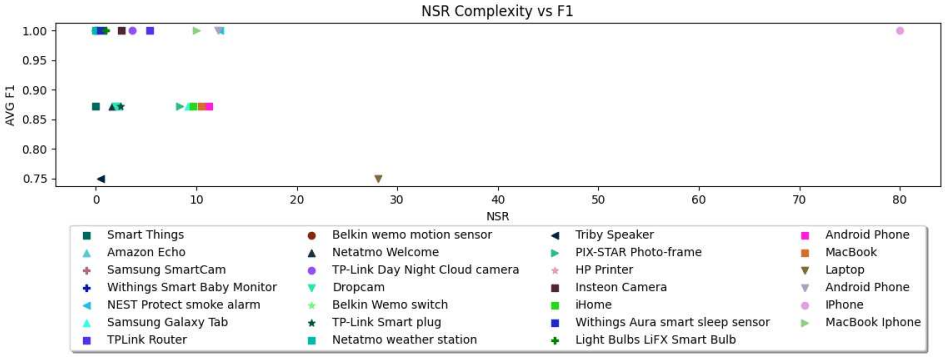


Figure 4.45: UNSW: NSR vs Average F1 Normalized

4.5 SCADA Dataset Results

This section shows the results of the METEC natural gas testing site network dataset. Figure 4.46 shows the variance of METEC devices as calculated in Section 3.4.2. Devices are sorted by variance from left to right and the figure was generated using 1000 flows (if they were available).

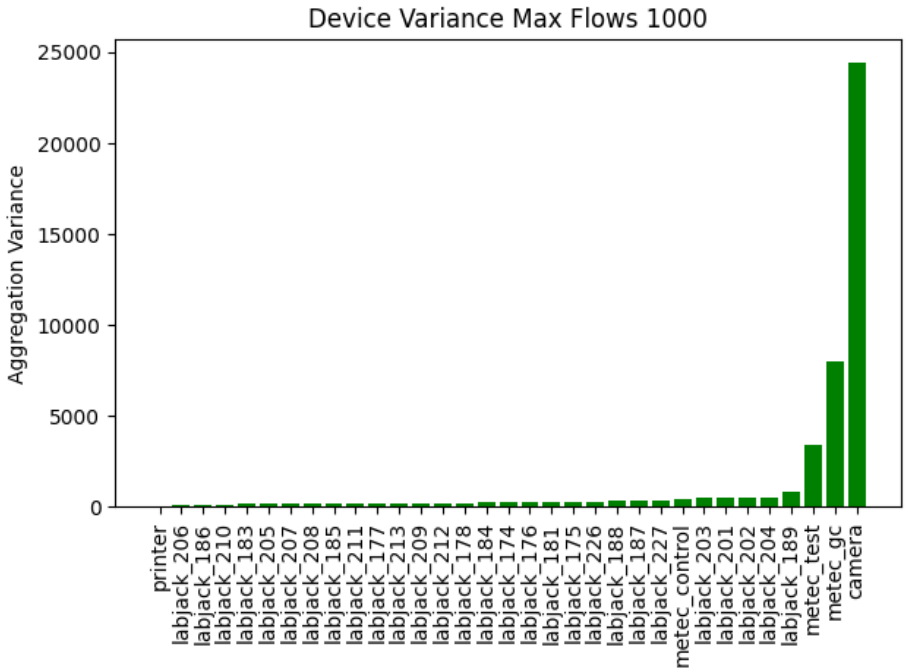


Figure 4.46: SCADA: Aggregate Device Variance

Figure 4.47 shows the IP complexity as calculated in Section 3.4.3 for METEC devices. Devices are sorted by IP complexity from left to right. The figure was generated using maximum of 1000 flows if they were available.

Figure 4.48 shows the ranked complexity increasing from left to right of he SCADA devices as measure by the total number of unique flows.

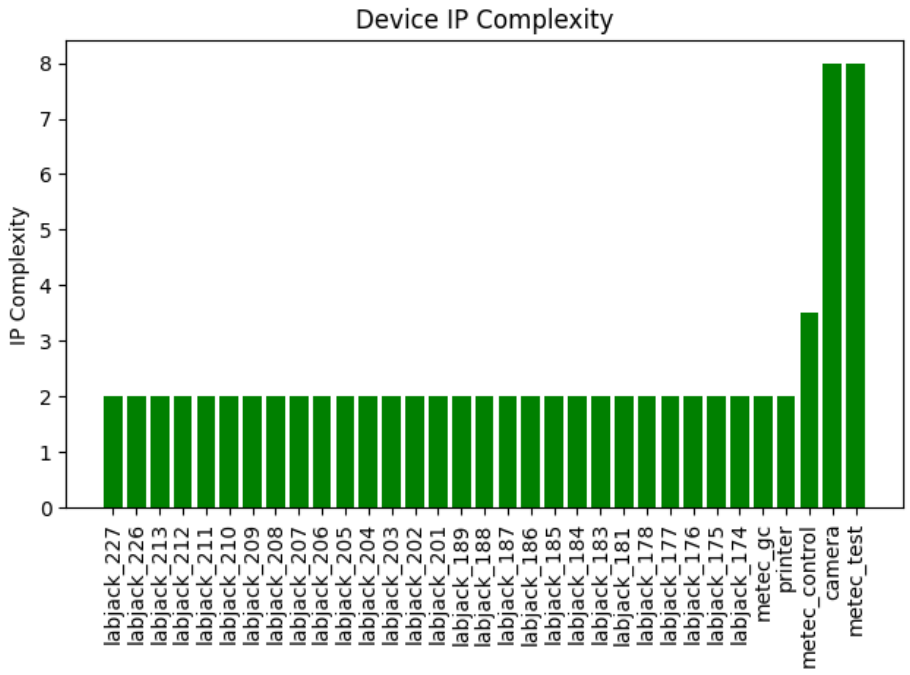


Figure 4.47: SCADA: IP Device Complexity

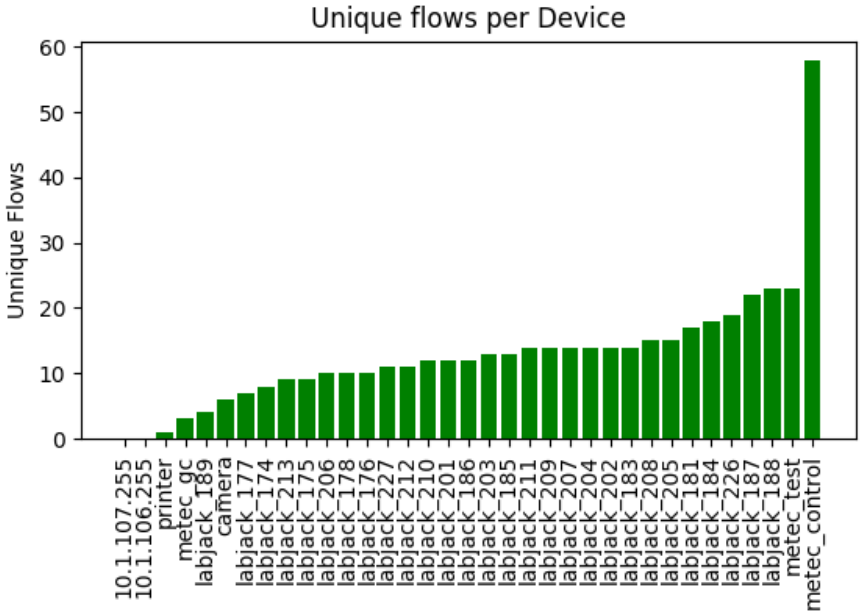


Figure 4.48: SCADA: Flow Complexity

Figure 4.49 shows the NSR complexity of devices with increasing complexity from left to right. For comparison, a line is drawn at an NSR of 8.438, the average NSR value for the home, lab, and UNSW datasets. Unlike previous results, which had a mix of general purpose devices with enough flows to analyze, no similar observations can be made about the SCADA dataset, which consists mostly of single-purpose industrial control devices.

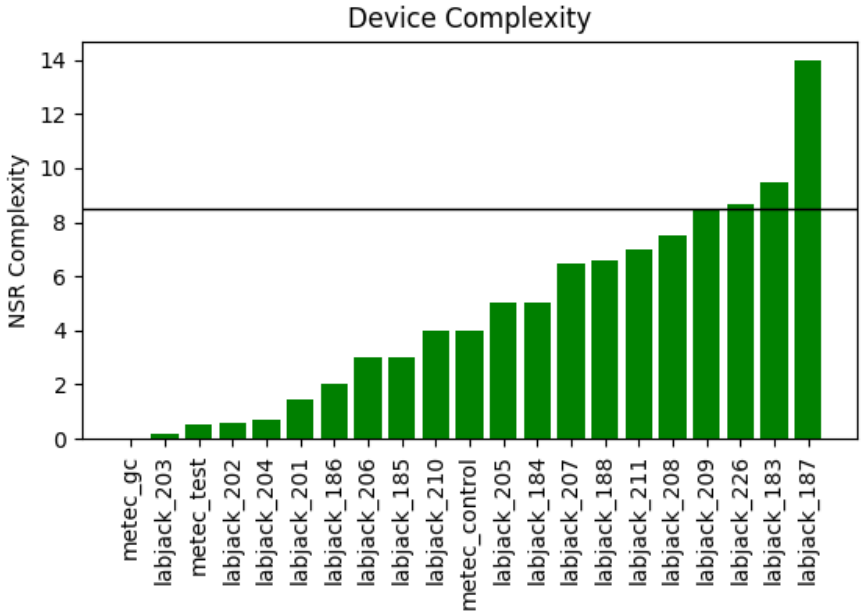
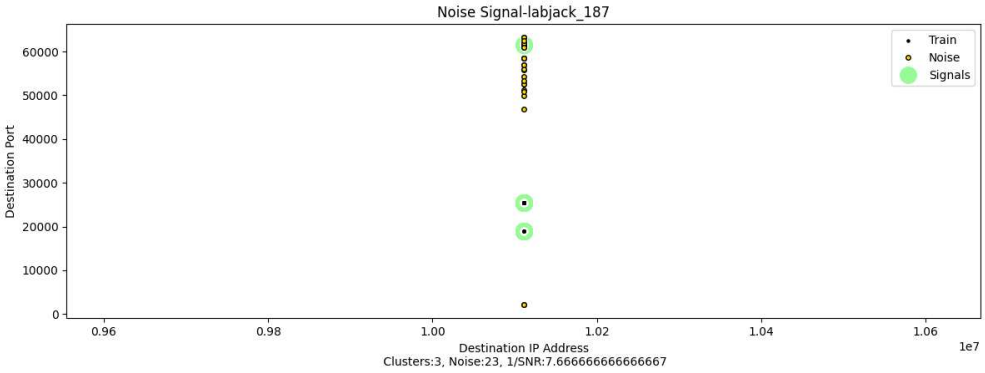


Figure 4.49: SCADA: NSR Complexity

Figures 4.50, 4.51, and 4.52 show the results for the NSR complexity of three devices: the Labjack 187 (a high NSR) device, the Metec Control (an average NSR) device, and the Metec GC (a low NSR) device on the SCADA network. The LabJack 187 in Figure 4.50 show three signals/clusters and 23 points of noise. All traffic from this devices is directed at a single destination IP address with a wide range of ports. Figure 4.51 has two signals/clusters and 7 points of noise. Figure 4.52 is the lowest complexity device of the dataset. It was found to have one signal and no points of noise. The signal shown in the figure encompasses the two points even though they are not covered by a green circle.

Figure 4.50: SCADA: Noise to Signal for Labjack 187



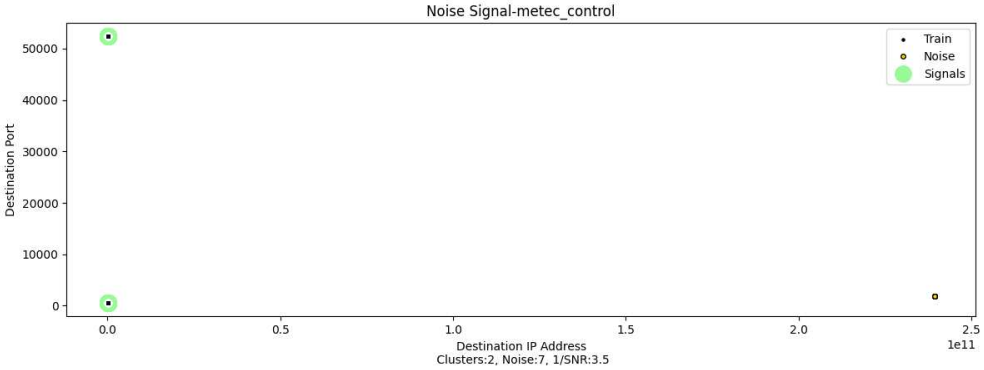


Figure 4.51: SCADA: Noise to Signal for Metec Control

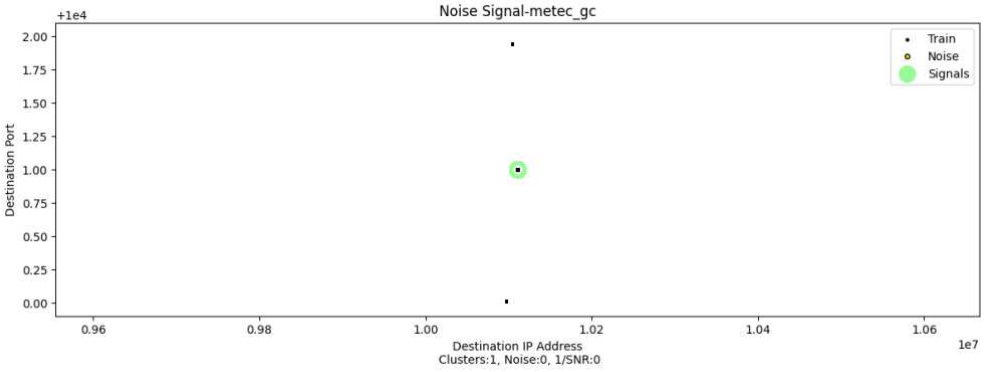


Figure 4.52: SCADA: Noise to Signal for Metec GC

Malware Attack Results

These results show how the attacks listed in Table 3.6 are detected as abnormal from the device models in the SCADA dataset. I show the most complex device (LabJack 187), a device of average complexity (Metec Contol), and the lowest complexity device (Metec GC). Each device is shown against two types of malware traffic from Table 3.6: C&C, and C&C File Download traffic. These attacks were chosen as they represent the largest variance in both destination IP and destination port. Each figure in this section shows the standardized destination port and standardized destination address. Normal test data is shown as black dots. The attack data is shown in red dots. Light blue ellipsoids show *estimates* of the inlier probablity threshold learned by the GMM. These ellipsoids were generated for the figures in a generalized manner to show details in each figure and do not perfectly align with the actual inlier probablity threshold of each device model. The results of all of the attacks against all of the home devices can be found in Appendix E.

Figures 4.53 and 4.54 show the Gaussian boundaries for the Labjack_187. The Labjack_187 was the only device with the relative highest NSR to achieve a perfect F1 score against both the C&C and C&C Filedownload traffic. These figures show two ellipsoids that approximate the two Gaussians found by the GMM model (there was some probabilistic difference in the NSR calculations resulting in slightly different numbers of signals). The ellipses shown in the figure only approximate the inlier probablity threshold accounting for the perfect F1 score of this device model despite the overlap between ellipse and attack traffic points.

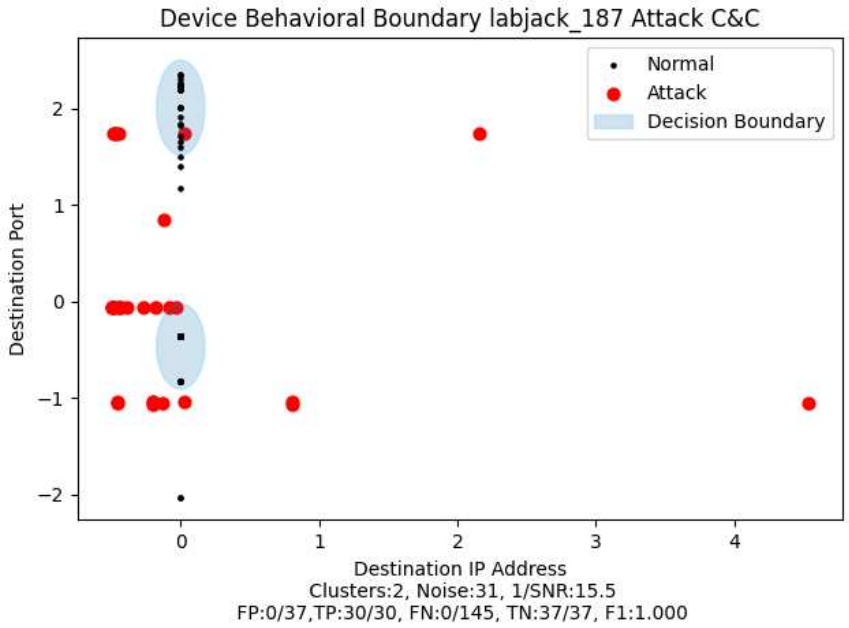


Figure 4.53: SCADA: C&C Attack And Gaussian Boundary Labjack 187

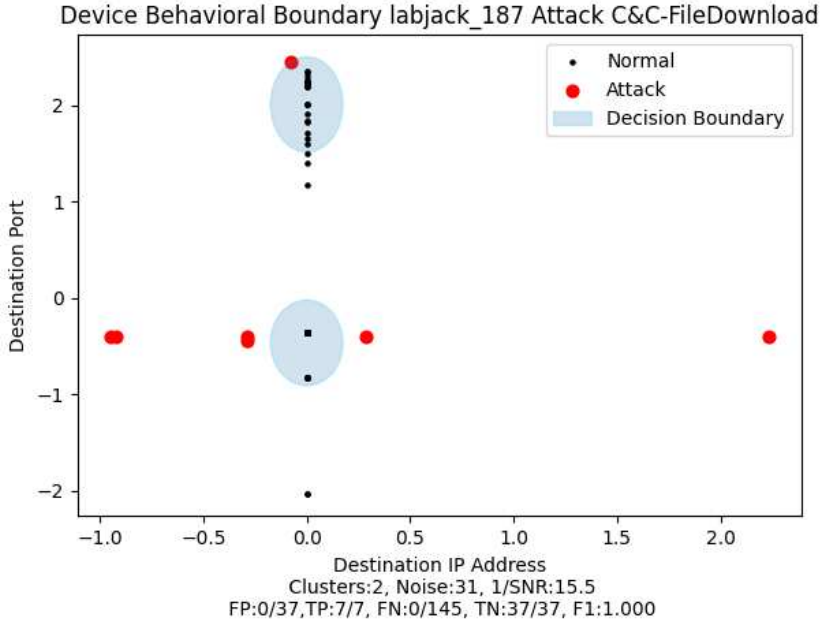


Figure 4.54: SCADA: C&C Attack And Gaussian Boundary LabJack 187

Figures 4.55 and 4.56 show the Gaussian mixture boundaries for the Metec Control, an average NSR complexity device against the C&C and the C&C Filedownload traffic. These figures show

two ellipses reflecting the inlier probablity threshold found by the GMM. In both device models there was one false positive, the estimated boundaries shown in the ellipses do not accurately reflect the boundary of the GMM.

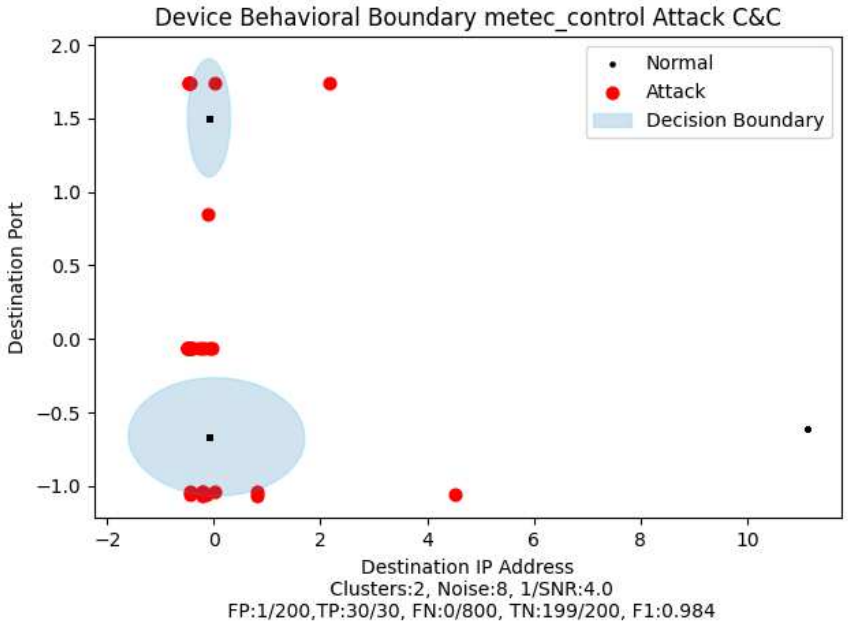


Figure 4.55: SCADA: C&C Attack And Gaussian Boundary Metec Control

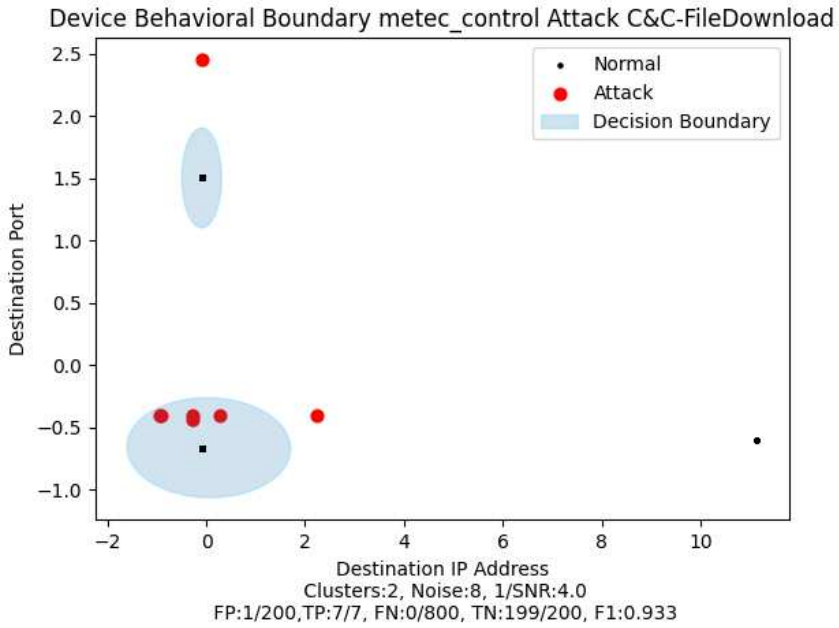


Figure 4.56: SCADA: C&C File Download Attack And Gaussian Boundary Metec Control

Figures 4.57 and 4.58 show the Gaussian mixture boundaries for the Metec GC, a low NSR complexity device against the C&C and the C&C Filedownload traffic. These figures show the least NSR complex device of the SCADA dataset with a single ellipsoid. The true inlier probablity threshold of this model must be tall and thin aligning with the single destination IP address and small range of ports.

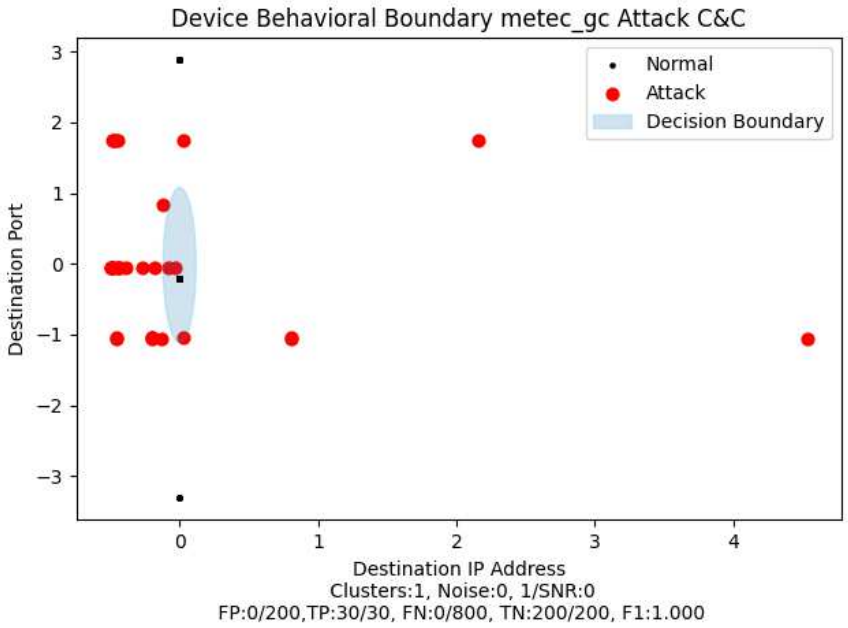


Figure 4.57: SCADA: C&C Attack And Gaussian Boundary Metec GC

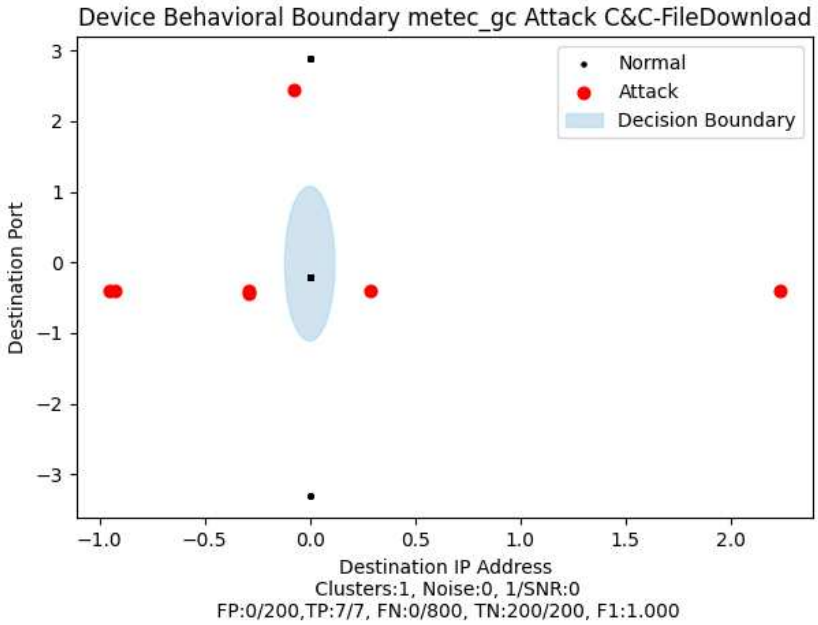


Figure 4.58: SCADA: C&C File Download Attack And Gaussian Boundary Samsung Metec GC

Figure 4.59 shows the F1 score for each device averaged over every attack versus the complexity of the devices as measured by the NSR method. Figure 4.60 shows the SCADA devices averaged F1 scores for all the devices against all seven types of malware traffic.

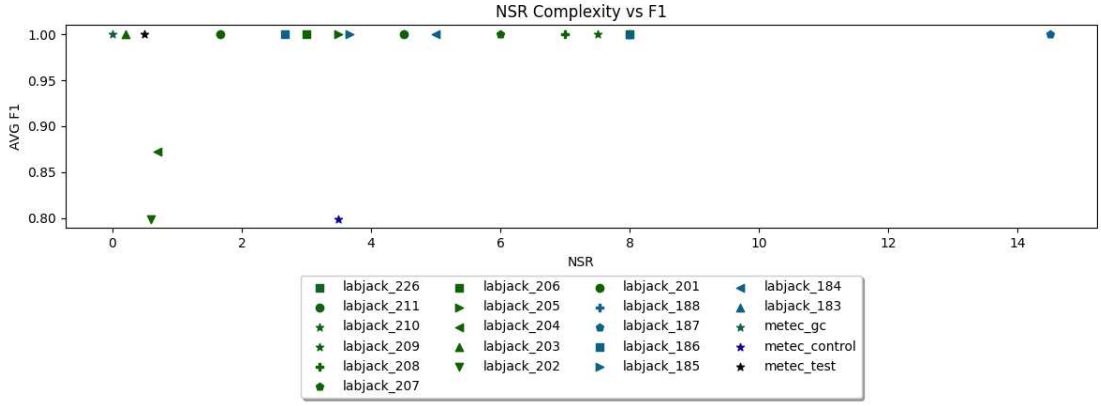


Figure 4.59: SCADA: NSR Vs. F1 Non-Normalized

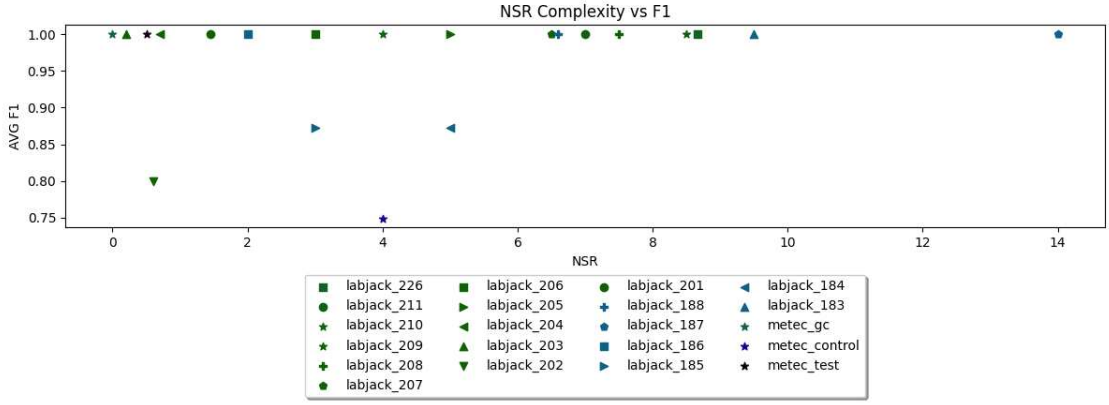


Figure 4.60: SCADA: NSR Vs. F1 Normalized

4.6 Cross Dataset Results

Table 4.1 shows the averages for variance, IP complexity, unique flows, and NSR complexity for each dataset. Table 4.2 shows the mean, standard deviation and coefficient of variance for each

measure of complexity. Table 4.3 shows the average NSR complexity and the F1 score for each dataset.

Table 4.1: Listing of Complexity Measurements by Dataset

Dataset	Avg. Variance	Avg. IP Complexity	Avg. Unique Flows	Avg. NSR
Home	256744.12	23590.55	13483.57	8.548
CSU Lab	39713.87	2521.40	2029.02	9.596
UNSW	2445.90	17049.18	6497.26	7.170
SCADA	1337.09	2.271	135.57	4.65

Figure 4.61 shows the trendline and R^2 value for the average F1 score for each dataset and the average NSR complexity. The R^2 of 0.968 shows a strong correlation between the NSR complexity and the accuracy of the GMM model.

Table 4.2: Cross-Dataset Comparison of Complexity Measurements

Complexity Measure	Mean	Standard Deviation	Coefficient of Variation
Variance	75059.75	106026	1.41
IP Complexity	10790.56	9845.89	0.91
Unique Flows	5536	5136	0.93
Noise to Signal	7.51	1.84	0.24

Table 4.3: NSR Complexity and F1 Score by Dataset

Dataset	Average NSR	F1 Score
Home	8.548	0.901
CSU Lab	9.596	0.879
UNSW	7.170	0.942
SCADA	4.65	0.975

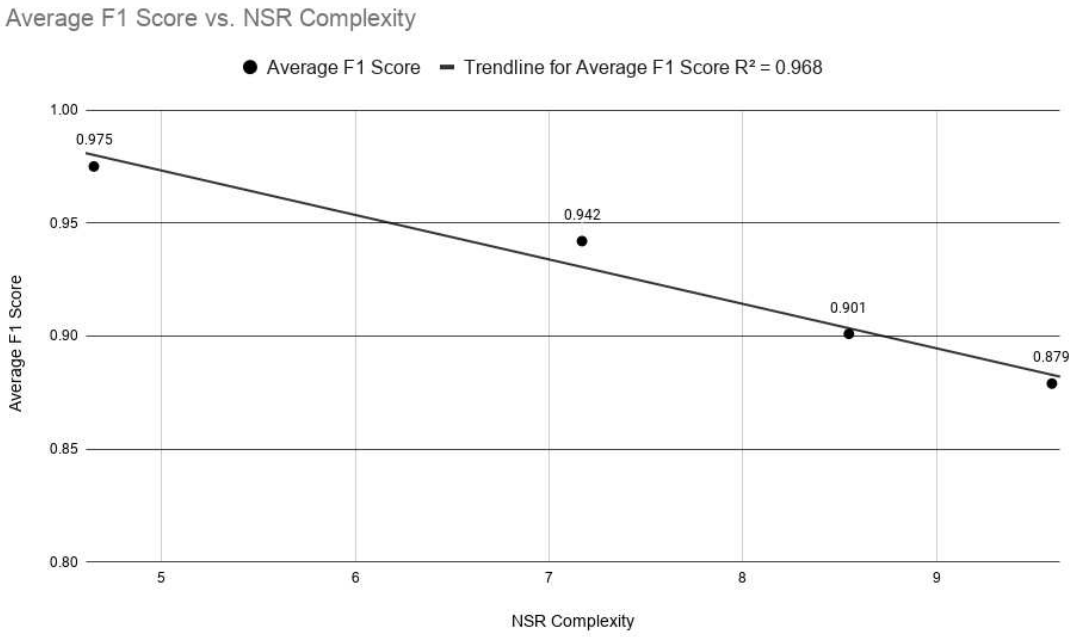


Figure 4.61: F1 Score Vs Complexity

Chapter 5

Discussion

5.1 Introduction

This chapter discusses the results shown in Chapter 4. It is organized with an overview of the key findings of this dissertation, then the results are discussed more specifically in sections that addresses each of the four datasets: home, lab, UNSW, and SCADA. Within each of these sections, I review the complexity measures pertaining to that dataset followed by a section that reviews three devices' behavior: one a high complexity device, an average complexity devices, and a low complexity device. This chapter concludes with a discussion comparing the results across the four datasets and the advantages and disadvantages of each method for measuring the complexity.

5.2 Key Findings

The results across all four of the datasets support the hypothesis that devices can be measured for complexity based on their network traffic, and that this measurement can be used to broadly categorize devices into two separate groups: single-purpose devices and general-purpose devices. Further, the results show that constructing anomaly detection models using the NSR measurement of complexity results in accurate behavioral models, particularly for low complexity devices that often do not have embedded security features.

The results in Table 4.1 show that other measures of complexity including variance, IP complexity, and unique flows range widely in value from dataset to dataset. Similarly, Table 4.2 shows that the coefficient of variation (CV) for each of these measures is 0.91 or greater. In contrast, the NSR complexity measure shows much smaller ranges, and its CV of 0.24 shows that the NSR metric has a low variance across all of the datasets. These results show that the ratio of noise and signal used in the NSR method is more consistent across the diverse set of devices measured.

Consistency and low variance make the NSR metric an effective standard measure of complexity for devices. Figures 4.4, 4.19, and 4.34 support this observation, showing that the average NSR complexity of the home, lab and UNSW datasets of 8.438, is an approximate threshold that separates general-purpose devices from single-purpose devices. General-purpose devices largely fall above this threshold and single-purpose fall below. The SCADA dataset is the exception to this result as it is composed of nearly all single-purpose devices.

An anomaly detection model that includes NSR was found to be accurate, particularly for low complexity devices. As shown in Table 4.1, the more simple a device is, the more accurate it can be modeled. This conclusion is demonstrated by Figure 4.61 showing an R^2 value of 0.968 for the average NSR complexity vs the average F1 score of the four datasets.

This finding is important as it establishes the NSR metric as an effective measure of device complexity and a valid method for constructing a device-specific anomaly detection algorithm. Autonomous network enforcement methods can use NSR complexity to identify devices that would benefit most from the anomaly detection model - low complexity, single purpose devices that are most vulnerable to attack and least capable of protecting themselves on their own.

5.3 Home Discussion

Overall, the results from the analysis of the home dataset show that low complexity devices can be more accurately modeled than higher complexity devices. The following sections discuss how complexity measures performed on the home dataset and how the GMM behavior model performs on three devices, the highest complexity device (Roku Express), an average complexity device (J Chromebook), and the lowest complexity device (Eufy Light) as measured by the NSR method.

Home Complexity

The four methods presented to measure complexity, IP_complexity, variance, unique flows, and NSR complexity, are seen in Figures 4.1, 4.2, 4.3, and 4.4 each can be used with varied success to sort devices from single-purpose to general-purpose.

Beginning with the variance shown in Figure 4.1, we see that with the exception of the Amcrest Camera, the top ten most complex devices, on the right side of the figure, are all general purpose devices. The devices filling out the left side of the figure are all single-purpose devices.

In Figure 4.2 the rightmost six devices are general-purpose devices, and the five leftmost are single purpose. The middle consists of a mix of both single and general-purpose devices.

The unique flows measure for the home dataset is shown in Figure 4.3. The measure performs well at grouping the general-purpose devices to the right, with the top eight in this category. The ten low complexity devices on the left side of the graph are all single-purpose.

The NSR complexity results, shown in Figure 4.4, provided the clearest distinction between groups, with general-purpose devices in the top 14 positions on the right, and 10 single-purpose devices at the left side of the figure.

Home Behavior

The GMM behavior models discussed in this section using only the NSR complexity measure. The highest, average and lowest devices by this measure are seen in Figures 4.5, 4.6, and 4.7. These three figures show the clusters found that feed into the behavioral GMM model. s

As shown in Figure 4.5 the Roku Express has an NSR value of 52, the highest of the home dataset, with only four clusters and 208 points of noise. It has approximately a 0.5% false-positive rate across both types of malware traffic. Visually, the figure shows large ellipses that do not contain all of the normal traffic within their boundaries and some overlap with the malware traffic. The number of true positive outliers detected is also poor with the model only detecting 43% of the malware traffic in the C&C traffic and 42% of the C&C File download traffic. As a result, the model has low F1 scores of 0.59 and 0.55 respectively, showing that the model was not accurate for this high complexity device.

Figure 4.10 shows that the J. Chromebook device has an NSR value of 8.6 which is close to the dataset average of 8.548 shown in Table 4.1. The DBSCAN method found a total of 15 clusters and 129 points of noise, leading the GMM to find a total of 15 ellipsoids. As compared to the model of the Roku, the J. Chromebook shows smaller margin ellipsoids centered around the normal traffic as seen in both Figures 4.10 and 4.11. The false-positive rate of the J. Chromebook model is relatively high at 1%. However, the ellipsoidal boundaries are more tightly bound around the normal traffic, leading to less overlap with the malware traffic, and consequently a true positive accuracy of 100% for the two types of malware traffic. This in turn leads to a much higher F1 score of 0.968 for the C&C malware and 0.875 for the C&C File Download malware.

Finally, Figures 4.12 and 4.13 show the Eufy light bulb with the lowest NSR score of 0, it had two signals and no noise. The model generated ellipsoidal inlier confidence intervals that are tightly coupled around the normal traffic, with a 0.0% false positivity rate and a 100% accuracy in identifying the malware traffic. This leads to a perfect F1 score of 1.00, showing that the model performed perfectly for the low complexity device.

The overall F1 vs. NSR results shown in Figure 4.14 shows the average of the F1 scores across all 7 types of malware traffic for each device. The general trend is for devices of lower complexity, identified by lower NSR values, to have higher F1 values, indicating a better predictive model. The red line on the figure demonstrates that single purpose devices can be separated from general-purpose devices by how accurately they can be modeled with single-purpose devices modeled

accuracy above F1=0.6 and general-purpose devices below. Figure 4.15 shows the same relationship of devices' average F1 scores vs NSR complexity with device models that used standardized data. This figure shows how standardization reduces the correlation between F1 and complexity, but improves GMM and thus the F1 score across all of the devices.

5.4 Lab Discussion

Overall the results from the analysis of the lab dataset show that low complexity devices can be more accurately modeled than higher complexity devices. The following sections discuss how complexity measures performed on the lab dataset and how the GMM behavior model performs on three devices, the highest complexity device (Note8), a median complexity device (Apple TV), and the lowest complexity device (Iview Smartbulb) as measured by the NSR method.

Lab Complexity

Figure 4.16 shows the variance of devices from the lab dataset, with the highest variance attributed to the Windows laptop. This figure shows a trend of general-purpose devices as high complex devices and were grouped on the right side of the figure. There was more mixing of single-purpose devices in the high end of complexity as measured by variance. The lowest variance device was the ovs-bridge, which was the SDN controlled switch for the network.

Figure 4.17 shows the IP complexity for the lab dataset. In this figure we see several IoT devices with high complexity, such as the show1-amazon-27, two Lutron switches, and the Arlo Q a video camera. It is not until we get down to the fifth most complex device with the thinclient that we first see what I consider a general-purpose device. For the lab dataset the IP complexity measure groups most of the general-purpose devices on the right hand side of the figure, however, there are several IoT devices mixed in, such as the Lutron devices which are light bulbs indicating that this measure of complexity is not forming distinct groups of single-purpose and general-purpose devices.

Unique flows for the lab network in Figure 4.18 also show several IoT devices with the most complex unique flows, as it groups all of the five Amazon devices at the top. In fact, we do not see a general-purpose device in the figure until the 10th spot with the LeEco-Phone.

The NSR complexity for devices in the lab is shown in Figure 4.19. All devices with a NSR complexity score greater than 8.438 (the lab average NSR is 9.5) are general-purpose devices with the exception of the Google-Home-Mini. All devices that have an NSR complexity lower than 8.438 are single-purpose devices.

Lab Behavior

Beginning with the device that has the highest NSR of 29, the Galaxy Note 8 has three clusters and 130 noise points as shown in Figure 4.20. When fed into a GMM model as seen in Figures 4.23 and 4.24, it shows the Note 8 with a false positive rate of 8% against both attacks. The model also shows that it misclassified 5% of the C&C malware traffic as normal traffic, and only correctly identified 91% of the malware traffic as anomalous, resulting in an F1 score of 0.054. For the C&C File Download traffic, the model only found 57% of the traffic, leading to an F1 score of 0.588. The low F1 scores indicate that this high complexity devices is not accurately modeled.

The average device, the Apple TV, has an NSR of 8.2 as shown in Figure 4.21, with 5 clusters and 38 points of noise. When analyzed with the GMM model, there are 6 Gaussians (note that there are some probabilistic differences between computational iterations that result in slightly different numbers of clusters). The model for the Apple TV has a false positive rate of 0.0%, and correctly identified all of the malware traffic, resulting in F1 scores of 1.00 for each malware type. The figure shows that the ellipsoids generated by the GMM more closely fit the normal data than the higher complex Note 8 device. This shows how the inlier confidence interval is more precise for lower complexity devices.

Finally, the lab device with the lowest NSR value of 0.5 from Figure 4.22 was the Iview Smart-bulb. Its GMM model has two Gaussians and a false positive rate of 0.5% across the two attacks. Both types of malware were 100% correctly identified in the models, leading to F1 scores of 0.984 and 0.933 showing a strong predictive accuracy.

The lab dataset, like the home dataset, shows that higher complexity devices have models that are less precise, with more errors that incorrectly classified normal traffic, malware traffic, or both. When the GMM model is used to analyze devices that are less complex, the model results in more precise boundaries of behavior. This trend is reflected in the F1 scores in Figure 4.29, which shows the average F1 score across the seven types of malware traffic for each device. Higher F1 scores are attributed to devices with lower NSR values. The red line on the figure demonstrates that single purpose devices can be separated from general-purpose devices by how accurately they can be modeled with single-purpose devices modeled accuracy above F1=0.6 and general-purpose devices below. Figure 4.30 shows the same relationship of devices' average F1 scores vs NSR complexity with device models that are standardized. This figure shows how standardization reduces the correlation between F1 and complexity, but improves GMM and thus the F1 score across all of the devices.

5.5 UNSW Discussion

The results from analysis of the UNSW dataset show that low complexity devices can be more accurately modeled than higher complexity devices. The following sections discuss how complexity measures are performed on the UNSW dataset and how the GMM behavior model performs on three devices: the highest complexity device (iPhone), an average complexity device (TPLink Router), and the lowest complexity device (Samsung SmartCam), as measured by the NSR method.

UNSW Complexity

Figure 4.31 shows the variance of devices ordered from low variance to high variance in the UNSW dataset. There does not appear to be as strong a correlation of general-purpose devices having high variance as was seen in the home dataset and to a lesser extent in the lab dataset. The Nest Dropcam showed the highest variance of traffic, however this result is not consistent with the home and lab datasets where cameras did not have the most variance in traffic. This high could be related to compression of the video stream or due to motion detection activation.

The IP Complexity of the devices in the UNSW dataset in Figure 4.32 does tend to have more general-purpose devices measured as more highly complex. The notable exceptions are the LiF light bulb and the SmartThings hub. Overall we can see that most of the devices in this dataset have very low IP complexity.

The unique flows in the UNSW dataset in Figure 4.33 show that the TPLink router has the highest number of unique flows, followed by the MacBook. The fact that the router has such as high number of flows in the UNSW data is interesting, as it must have a relatively large amount of layer three traffic on the network as can be seen in Figure 4.36.

The NSR complexity shown in Figure 4.34 performs well in ordering single-purpose devices to general-purpose devices. One of the exceptions is the iHome, which according to the original paper linked to the data, is a smart switch. The iHome has 37 noise points, a relatively high number with only two clusters. Additional information about this device may be needed to explain its high NSR value.

UNSW Behavior

The highly complex iPhone device shown in Figures 4.38 and 4.39 has a single Gaussian and 86 points of noise, which results in a poorly performing model of the device. Against the C&C attack the model has a false positive rate of 4% and a false negative rate of over 30%. The F1 score is very low at 0.062, which is heavily influenced by the model only identifying only 3% of the malware traffic, this shows that this model failed for the highly complex device.

The TPLink Router shown in Figures 4.40 and 4.41 is an average complexity device and has 13 Gaussians and 95 points of noise, for a NSR value of 7.30. This is slightly higher than the average for the UNSW dataset. The TPLink router had no false positives against either the C&C traffic or the C&C Filedownload traffic. Against the C&C traffic, the router only identified 40% of the malware traffic, contributing to an F1 score of 0.571. The router model fared better against the C&C FileDownload traffic with an F1 score of 0.923. The router shows some interesting characteristics in that its traffic seems to be between some very specific destination addresses where many of the connections have a high number of destination ports. It is possible that devices outside

of the dataset were configuring the router as similar patterns do not show up on other devices. This leads to the unique pattern of destination address and port pairs as shown in the figure.

Figure 4.42 shows the Gaussian behavior of the Samsung Smart Camera, a device with low complexity. This device had four clusters with zero noise points for an NSR of 0. The model for this device performed perfectly against the two types of malware traffic with zero false-positive rates, finding all of the malware traffic and missing none of the normal device traffic for an F1 score of 1.00.

The results for the UNSW dataset are similar to the results from the home and lab datasets, and continue to support the hypothesis that devices with high complexity have less accurate models. The iPhone device in Figures 4.38 and 4.39 shows a large behavioral boundary for the single cluster, whereas the boundaries for the less complex devices have smaller ellipsoids around each normal data point as shown in Figures 4.40, 4.41, 4.42, and 4.43.

The overall F1 vs NSR results depicted in Figure 4.44 shows an average of the F1 scores across the seven types of malware traffic for each device. The general trend is for devices of lower complexity to have higher F1 values. The red line on the figure demonstrates that single purpose devices can be separated from general-purpose devices by how accurately they can be modeled with single-purpose devices modeled accuracy above F1=0.6 and general-purpose devices below. Figure 4.45 shows the same relationship of devices' average F1 scores vs NSR complexity with device models that are standardized. This reduces the correlation between F1 and complexity, but improves the F1 score across all of the devices.

5.6 SCADA Discussion

The results from the SCADA dataset were very different than the results of the other datasets examined in sections 4.2, 4.3, and 4.4. Overall the SCADA dataset has less variance, smaller IP complexity, fewer unique flows, and lower NSR complexity. This difference is likely due to the large number of industrial control devices and few general purpose devices. Several of the devices had fewer than 100 flows, such as the printer and the camera.

SCADA Complexity

Figure 4.46 shows the variance for each device on the SCADA network. The camera was the most variant device on the SCADA network, and surprisingly the printer was the least. The results for the camera are likely not accurate as there were only 25 flows from that device. The printer is also likely inaccurately measured with only 36 flows.

As seen in Figure 4.47 there is a clear pattern where LabJack devices all have the same IP complexity score. From this figure, it is clear that all the LabJacks, the metec_gc, and the printer *only* communicate on the local network, while the metec_control, camera, and the metec_test all communicate to devices off of the local area network (LAN).

Figure 4.48 shows the unique flows per device. The metec_control device has approximately three times the number of unique flows as any other device on this network. Overall the devices in the SCADA dataset had a smaller range of flows than any of the other datasets, with the largest span consisting of only 60 unique flows.

The overall NSR complexity for the SCADA dataset is shown in Figure 4.34, which provides a ranking of devices by NSR complexity. This result only shows devices that had at least 100 flows, as several of the devices shown in previous figures did not have the minimum number of flows for NSR analysis. It was unexpected to see that the highest measured device based on NSR complexity was the LabJack_187, and that the metec_control (SCADA controller) has an average NSR measure, as this is the computer that controls all of the SCADA devices.

SCADA Behavior

Figures 4.53 and 4.54 show the Gaussian boundaries for the LabJack187. Due to the overall lower number of flows, there are some probabilistic differences between the number of clusters found in Figure 4.50 and the number of clusters in Figure 4.53, which shows two clusters and 31 points of noise. This leads to an NSR value of 15.5. Despite this relatively high NSR value, the model for the LabJack_187 still performs well against both the C&C and C&C Filedownload malware traffic, with perfect F1 scores for both. The LabJack_187 has the highest complexity of

the dataset, but this is low relative to high devices of the other datasets, leading to an effective model.

Figure 4.55 and Figure 4.56 show the Gaussian boundaries for the metec_control device. It has two clusters and eight points of noise leading to an NSR of 4.0. This model leads to a 0.05% false-positive rate against both attacks. The model for the metec_control correctly identified both types of malware traffic, the C&C and C&C FileDownload, which led to an F1 scores of 0.984 and 0.933.

Figures 4.57 and 4.58 show the model for the metec_gc device. This model has only a single cluster, leading to a single Gaussian. The Gaussian drawn does not match the model well, but as there were no false-negatives found, the two points that fall outside the drawn Gaussian must be within 2σ of the center normal data point. The model performs perfectly against the two types of malware with a zero false positive rate. All malware data points identified were correct, leading to an F1 score of 1.00 for both, again showing that low complexity devices have accurate models.

The NSR and the F1 average for all of the SCADA devices is shown in Figure 4.59. A large majority of the devices have perfect F1 scores of 1.0 against all of the seven attacks. Only three devices have less than a perfect score, the metec_control, LabJack_202, and LabJack_204. Figure 4.60 shows where all the device data were standardized prior to modeling, which actually seems to perform less well than the non-standardized models. However, I suspect its performance was equal to the previous model, and just reflects the probabilistic nature of the model. Standardization does not have the performance gains in the SCADA dataset, likely due to the limited IP and port space, which makes the points closer without standardization.

5.7 Cross Dataset Discussion

It is not possible to compare devices of the same make and model across datasets, as there was very little overlap in specific devices. The home dataset represents the most realistic dataset in terms of the typical usage of IoT devices, as it comes from a house with devices that are consistently interacted with.

The CSU lab, while containing devices that might be found in a home, did not replicate realistic usage. The CSU lab was unoccupied for long periods of time, and devices were not used in the same manner as they would be used in a home. For example, there were several concurrent experiments taking place in the CSU lab, where devices were being unplugged, removed, and rejoined to the network, much more often than a typical home environment.

The UNSW dataset may be more representative of a baseline of traffic for IoT devices. The authors [29] claim that they took captures for 26 weeks, but they do not provide any indication of how much they interacted with the devices during this time. The UNSW data has an average variance that is similar to the CSU lab, and IP complexity near that of the home network.

The SCADA network is clearly different from the other three datasets, as is shown across average variance, average IP complexity, and average NSR. These results were consistent with what I would expect, given a protected network of industrial devices. For example, these devices do not contact the broader Internet, and the traffic is mostly control traffic from the SCADA controller to the end-node devices. This makes for much lower values in variance, IP complexity, and NSR complexity.

An unexpected result that appeared in several of the datasets and across several of the complexity measures was that streaming devices, such as Roku and Amazon devices, were high complexity devices. These devices are small, seemingly simple Linux-based computers that are capable of running various user applications simultaneously. The complexity scores of these devices are similar to other general-purpose devices that run user applications, such as Android-based devices and XBoxes. The discussion in this chapter treated these streaming devices as general-purpose devices.

Complexity Comparison

Variance Variance uses how the aggregates (the packet and bit rates) of the traffic change over time. The results across several datasets support the observation that devices with high variance are often those that produce or consume media of some sort. I believe this is why we see a mix of general-purpose and single-purpose devices with high variance. Laptops and smartphones are

used to consume media, as are streaming devices, and these devices had high variances. Cameras are media producing and show up as top variant devices. The Nest Protect smoke alarm showed up as a highly variant device in both the lab and the UNSW dataset, which was likely due to the low quantity of flows found in each dataset, 29 and 63 flows respectively.

Table 4.2 shows that the variance complexity measure has a mean of 75059, a standard deviation of 106026, and a coefficient of variation (CV) of 1.41. The large variance of the averages from each dataset indicate that this measure of complexity may not show consistent results and may be highly dependent on how much the devices are used.

IP Complexity The IP complexity measure has both advantages and disadvantages. One advantage is that only a single feature, the destination IP address, is required to compute the IP complexity. This method is also completely independent of state of the flow, i.e it does not need to track the number of packets or bytes. Because it inherently tracks the hierarchical structure of IP networks, it is more nuanced than a simple count of IP addresses, and captures the common relationships of devices to their manufacturers and services.

Table 4.2 shows that the IP Complexity metric has an average of 10790, a standard deviation of 9845, and a CV of 0.91. The large variance in the results indicate that like variance, the IP complexity measure may not show consistent results as a general metric of devices complexity and may depend on device use to a large degree.

Unique Flows The unique flows method of measuring complexity captures a mix of both IP complexity and variance. Flows are unique not only if their destination tuples are different, but also if the aggregation features are different. The unique flows metric has an mean of 5536, a standard deviation of 5136 and a CV of 0.93. Similar to variance and IP complexity, unique flows have a high variance across the dataset averages, and may not present consistent results as a general measure of device complexity.

Noise to Signal Ratio Noise to signal ratio, or NSR, can use any of the features from Table 3.1,but this work used only destination IP address and destination port. Using more of the available flow features could result in higher accuracy, which is an interesting direction for future work. The NSR method was the most consistent complexity measure across home, lab, and UNSW datasets. Figures 4.4, 4.19, and 4.34 show that NSR was also the most consistent at ranking general-purpose devices as high complexity and single-purpose devices as low complexity.

Table 4.3 shows the average NSR and F1 score for each dataset. These results shows that a high NSR complexity leads to a low F1 score, and a low NSR leads to high F1 score. The CSU lab dataset had the highest NSR complexity average of 9.596, and correspondingly the lowest average F1 score of 0.879. The SCADA network had the lowest average NSR average of 4.65, and the highest F1 average of 0.975.

Overall, the results of this study indicate that the NSR is a good step towards a standard measure of device complexity and when combined with the Gaussian mixture model anomaly detection method, provides an accurate model for detecting malware traffic, especially in low complexity devices. The results in Table 4.1 demonstrate this consistency with the lowest CV of 0.24 calculated from the four averages of each dataset's NSR complexity.

5.8 Limitations

There were several limitations in this study that should be noted. These limitations can be grouped into two categories, dataset and analysis.

5.8.1 Dataset Limitations

Devices that had less than 100 network flows were eliminated from further analysis. This removed some devices that might have had statistical significance to the study, and reduced the total number of devices contributing to the results.

Additionally, two of the datasets, the CSU lab and the UNSW dataset, were test environments, which may not have been representative of typical device usage.

Another limitation was that there were very few devices that were the same make and model across the datasets, making a direct and specific comparison of devices and datasets outside the scope of this research.

Finally, the attack dataset was generated by malware running on a Raspberry Pi computer, which may present different patterns of traffic than if it were to run on other common IoT devices. In addition, the attack dataset comes from a third party network that is configured with different addresses for devices. This places some limitations on the results using the PartofHorizontalPortScan attack because the IP addresses in the device data will not match the target addresses in this attack.

5.8.2 Analysis Limitations

The scope of this dissertation did not include analysis of NTP traffic. The Network Time Protocol (NTP) is the standard on the internet for obtaining accurate time. NTP is often combined with the widely used public service hosted at pool.ntp.org. This service combines over 4000 NTP servers into a highly redundant and scalable virtual cluster. Pool.ntp.org uses a round-robin DNS query, a query that returns a rotating list of servers for each response. This method produces a relatively random selection of a time servers and introduces a large amount of entropy into the data. This entropy raises the complexity of devices that participate in pool.ntp.org, and results in a less accurate predictive model for the device.

5.9 Enforcement

Building an autonomous enforcement model based on the GMM complexity method would involve analyzing flows from individual devices, and allowing flows that are calculated as normal while blocking flows that are abnormal. There two stages of analysis. The first occurs at the connection setup and the second is an ongoing evaluation of the traffic flow to detect aggregate anomalies.

Connection Stage: The connection stage examines features of the confidence model that are known at connection time, including IP header attributes, such as IP source, IP destination, port, and protocol. The connection stage is only run once prior flow setup.

Aggregation Stage: The aggregation stage gathers flow aggregate information that includes outgoing bytes-per-second and outgoing packets-per-second. The aggregation stage is run continuously while the flow is active. The aggregation stage compares the trained model to the current flow.

5.9.1 Proposed Enforcement Architecture

Traditional networking topologies do not offer the amount of programmability and flexibility needed to implement the proposed enforcement architecture. Therefore the proposed architecture is based on a software-defined networking model. Software-defined networking (SDN) decouples the control and the data plane in routers and switches. This method opens the network to new services, features, and a level of dynamism that was previously not possible. This work leverages the programmability of the network to focus on low NSR complexity devices and dynamically allow, block, rate-limit and route traffic based on whether the flow is normal or anomalous.

The centralized SDN enforcement architecture shown Figure 5.1 consists of three functional components, a switching/routing component, a controller, and a flow confidence engine. The switching and routing component is based on the Openflow specification led by the Open Networking Foundation (ONF) [34]. OpenVSwitch [35] is a software-based network switch that implements OpenFlow.

RYU [41] is an SDN controller that implements OpenFlow and it installs new flows in OpenVSwitch. New flows pushed to RYU by OpenVSwitch are analyzed by the confidence engine. The confidence engine runs the Gaussian behavioral model, and analyzes the current flow connection and returns a normality/anomalous score for each flow. It sends the flow score information back to RYU, which uses the received flow score to add, remove, or modify flow rules based on the

confidence score. RYU then pushes flow rules to the openVswitch flow tables. Device complexity scores dictate to RYU if new flows should be allowed, rate-limited, or dropped.

The architecture in Figure 5.1 allows the network to make extremely granular flow decisions on every flow in the network, including inbound/outbound traffic to the Internet and intra-network device traffic. Based on the behavioral boundary there is no need to isolate an entire device, just the flow that is anomalous.

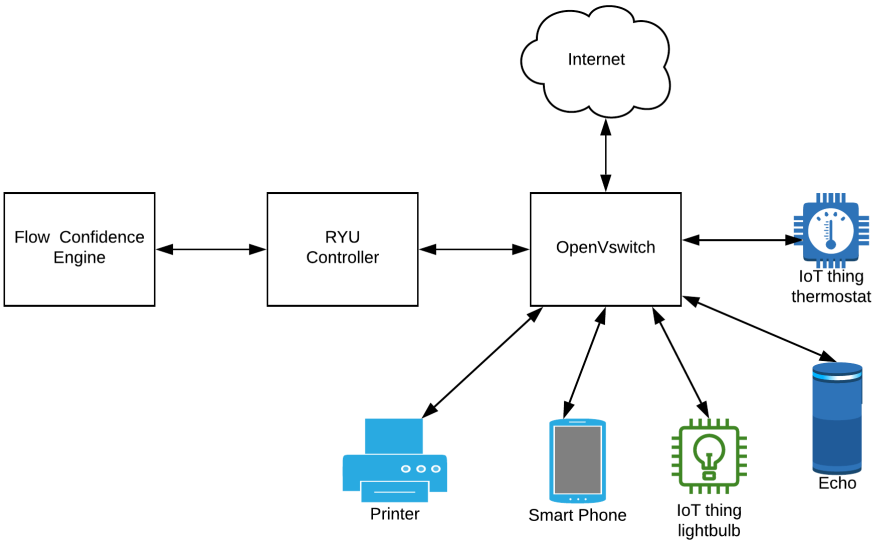


Figure 5.1: Enforcement Architecture

5.9.2 Proposed Enforcement Policies

Even a perfect anomaly detection method cannot perfectly infer the intent behind the anomaly, it can only determine if the new example was an anomaly or not. Before the learning model incorporates anomalies into the learning model some benign intent must be assigned to those anomalies. This could be accomplished in several ways such as signatures of know attacks (non-benign), or known publications of firmware upgrade (benign).

If the intent of the anomaly cannot be automatically deduced through the previously mentioned methods, then complexity can assist in prioritizing the alerts that must goes to the administrator of the network. The policy for these alerts can be guided by the complexity of the devices and the

probability ranking of the anomalous event. For example, a low complexity device that the model has detected a anomaly with high probability should go to the administrator for further analysis, while a high complexity device with a low probability anomaly could be logged.

Chapter 6

Conclusion

6.1 Thesis Summary

Chapter 1 summarized how insecure IoT systems have a history of participating in high impact attacks and represent a significant problem to the Internet. The challenge in securing these devices is two-fold: first, they are easy targets for attackers because they often have undiscovered exploits and vulnerabilities, and second, they cannot run the necessary security software to protect themselves due to computing, ram and power constraints.

Chapter 2 reviewed related research and proposals on using machine learning to identify, categorize, and classify IoT devices based on their network traffic and behavior. This chapter reviewed two popular methods of machine learning, supervised and unsupervised, and their advantages and disadvantages. Supervised methods can be used to accurately identify devices, but require large sets of labeled traffic that is often unavailable. Also, once trained, these classifiers are often unable to recognize new devices. Unsupervised methods are more adaptable and can learn models for unknown devices, but suffer from overgeneralization by indiscriminately modeling all devices in the same way.

In Chapter 3 this work proposed a method of informed autonomous learning that provides the flexibility of unsupervised methods with contextual information based on the complexity of the device. This chapter introduced several methods of measuring the complexity and proposed a new one called noise to signal ratio (NSR). This chapter showed how the NSR metric could be combined with a Gaussian mixture method to build an anomaly detection model closely tied to the complexity of each device.

Chapter 4 showed that anomaly detection models of simple devices perform better than models of more complex devices. These results are supported by over 800 experiments that evaluated how

118 diverse devices across four separate datasets detected seven different types of real malware traffic.

Finally, Chapter 5 discussed how the complexity measure using NSR correlates to the accuracy of modeling across all of the datasets. This chapter also reviewed the results for complexity and behavior in each dataset, and how these results compared across the four datasets. This chapter summarizes the advantages and disadvantages of each complexity measure and showed that the NSR method is the most consistent metric and is independent of device use.

This work makes the following contributions:

1. *Complexity*: I develop a formal measure of device complexity based on network flow traffic called noise to signal ration (NSR). NSR was shown to be an effective standard of measure for complexity, as it shows consistency and low variance when compared to other measures. NSR can be used to categorize and rank devices on the network. I show that devices measured as low complexity are often single-purpose devices, and devices measured as high complexity are typically general-purpose devices. This differentiation could be used by the network to apply security models deferentially to devices of varying capabilities.

2. *Behavior*: I showed that the accuracy of a device's predictive behavioral model is dependent on the complexity of their network traffic. I constructed a Gaussian mixture model (GMM) using NSR and analyzed device models against seven types of malware traffic sources from actual IoT malware infections. I show that the anomaly detection model correctly identifies anomalies and has high F1 scores averaged over 118 devices and 800 total experiments. I show that the complexity measure based on the NSR method is strongly negatively correlated to the accuracy of a device model's ability to detect inliers and outliers, demonstrating that the NSR-based anomaly detection model is particularly accurate for low complexity devices, which are more difficult to secure due to their highly constrained nature.

An immediate and practical use of these contributions could be to guide additional network analysis and inspection by determining where and how to dedicate security resources. Policies

could be developed that would adjust enforcement based on the complexity of the device. For low complexity and accurately modeled devices, algorithms that reduce false negatives might be preferred. For high complexity devices that can run security software, whose models are less accurate, a more liberal network security policy might be chosen. Aspects of this proposal using a one class support vector machine (OCSVM) [42] are discussed in Appendix A.

6.2 Future Work

The complexity measures developed in this dissertation have several further applications that can be explored. This section outlines some of those possibilities.

1. *Create an ensemble anomaly model:* One of the most immediate applications of this research could explore building an ensemble method classifier by combining the GMM and isolation forest algorithms [43]. The GMM and isolation forest algorithms use complimentary methods of finding outliers. GMMs find inliers by combining data to form clusters, while Isolation forest algorithms partition data to find outliers. For GMMs the signal part of the NSR is used to determine the number of clusters. For isolation forest algorithms the noise could be used to determine the amount of contamination the algorithm will expect. This work would need to include a method to calculate an overall consensus between anomaly scores produced by the two algorithms.

2. *Build and test the reference architecture:* The results of this research used models based on actual device and malware traffic, however, to fully demonstrate the efficacy of using this model for IoT devices, future work would need to implement the architecture proposed in 5.9.1. This future work should have a mix of common IoT devices and a few that can be infected with malware. The goal will be to build a fully functioning network and record errors where the model incorrectly classifies traffic.

3. *Expand to the network level:* This work would explore if a complexity metric similar to that of a single device can be expanded to a complete network. If networks such as those found

in the relationship of a subscriber to an Internet Service Provider (ISP) are considered as an atomic entity, then the complexity measure could be used to compare the risk of one individual subscriber network to another. This evaluation could be done in a privacy-preserving way, as it does not require deep packet inspection.

4. *NTP*: The nature of NTP requests as stated in 1.5 makes it difficult to model purely on an destination port and IP address basis, as this introduces too much entropy into the model. Future work in this area could examine symmetry in NTP response and answers to determine if there is invariance in the traffic data that can be modeled.

In conclusion this research proposes several methods to measure the complexity of devices on a network, the most promising of which is called noise to signal (NSR) complexity. The NSR complexity offers a quantifiable way to determine the probabilistic accuracy to which the device can be modeled. The negative correlation between complexity and model accuracy shows that a low complexity device can be more accurately modeled than a higher complexity device. This finding can be used to improve the accuracy of anomaly detection algorithms and allows the field to progress in the development of effective autonomous security control measures.

Although they do not offer a complete resolution of IoT security issues, this research provides a step forward in addressing the highly complex and growing problem of insecure devices by empowering the network to be an active part of the solution.

Bibliography

[1] Kevin Ashton et al. That internet of things, thing. *RFID journal*, 22(7):97–114, 2009.

[2] Sandip Ray, Yier Jin, and Arijit Raychowdhury. The changing computing paradigm with internet of things: A tutorial introduction. *IEEE Design & Test*, 33(2):76–96, 2016.

[3] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[4] OCF Security Working Group. Open connectivity foundation (ocf) specification part 2: Security specification. Standard, Open Connectivity Foundation, 2019.

[5] Scarfone K Smith M Fagan M, Megas K. Foundational cybersecurity activities for iot device manufacturers. report, National Institute of Standards and Technology, 2019.

[6] CTA-C2 Working Group. The c2 consensus on iot device security baseline capabilities. Technical report, Consumer Technology Association, 2019.

[7] ENISA. Baseline security for iot. report, 2017.

[8] JSOF. "overview- ripple20.", 2020.

[9] New mirai variant expands arsenal, exploits cve-2020-10173, 2020.

[10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[11] Robert F Ling. On the theory and construction of k-clusters. *The computer journal*, 15(4):326–332, 1972.

[12] various. Device provisioning protocol specification 1.1. report, 2020.

[13] D. Romascanu E. Lear, R. Droms. Manufacturer usage description specification. rfc, March 2019.

[14] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. Iot sentinel: Automated device-type identification for security enforcement in iot. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2177–2184. IEEE, 2017.

[15] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. Behavioral fingerprinting of iot devices. In *Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security*, pages 41–50. ACM, 2018.

[16] Samuel Marchal, Markus Miettinen, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and N Asokan. Audi: Toward autonomous iot device-type identification using periodic communication. *IEEE Journal on Selected Areas in Communications*, 37(6):1402–1412, 2019.

[17] Jorge Ortiz, Catherine Crawford, and Franck Le. Devicemein: Network device behavior modeling for identifying unknown iot devices. In *IoTDI'19, Montreal, QC, Canada*. ACM, 2019.

[18] Jingjing Ren, Daniel J Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In *Proceedings of the Internet Measurement Conference*, pages 267–279. ACM, 2019.

[19] Ibrahim Alrashdi, Ali Alqazzaz, Esam Aloufi, Raed Alharthi, Mohamed Zohdy, and Hua Ming. Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0305–0310. IEEE, 2019.

[20] Mahmudul Hasan, Md Milon Islam, Md Ishrak Islam Zarif, and MMA Hashem. Attack and anomaly detection in iot sensors in iot sites using machine learning approaches. *Internet of Things*, 7:100059, 2019.

[21] Aristotle and Jonathan Barnes. *Aristotle's Posterior Analytics*. Oxford: Clarendon Press, 1976.

[22] Andrei N Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 369–376, 1963.

[23] Jorma Rissanen. *Stochastic complexity in statistical inquiry*. World Scientific, 1989.

[24] HA Ceccatto and Bernardo A Huberman. The complexity of hierarchical systems. *Physica Scripta*, 37(1):145, 1988.

[25] William Bialek, Ilya Nemenman, and Naftali Tishby. Predictability, complexity, and learning. *Neural computation*, 13(11):2409–2463, 2001.

[26] Mark Allman, Vern Paxson, and Jeff Terrell. A brief history of scanning. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 77–82, 2007.

[27] Kyle Haefner and Indrakshi Ray. Complexiot: Behavior-based trust for iot networks. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 56–65. IEEE, 2019.

[28] Kyle Haefner and Indrakshi Ray. Trust and verify: A complexity-based iotbehavioral enforcement method. In *14th IFIP WG 11.11 International Conference on Trust Management*. IFIPTM, 2020.

[29] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 2018.

[30] METEC. Methane emissions test and evaluation center, 2012.

[31] Maria Jose Erquiaga Agustin Parmisano, Sebastian Garcia, 2020.

[32] cisco. Network analyzer, 2020.

[33] wfa.org. User space access point, 2020.

[34] OpenNetworking.org. Openflow switch erata, open networking foundation, onf ts-001, 2012.

[35] OpenNetworking.org. Open virtual switch (ovs): Production quality, multilayer open virtual switch, 2019.

[36] F. Loi A. Radford C. Wijenayake A. Vishwanath A. Sivanathan, H. Habibi Gharakheili and V. Sivaraman. Classifying iot devices in smart environments using network traffic characteristics, 2018.

[37] labjack.com. Measurement and automation devices, 2020.

[38] Brian S Everitt. A finite mixture model for the clustering of mixed-mode data. *Statistics & probability letters*, 6(5):305–309, 1988.

[39] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[40] G.E.P. Box and G.M. Jenkins. Time series analysis forecasting and control. *XF2006175558*, 3, 01 1970.

[41] ryusdn.org. Ryu sdn framework, 2019.

[42] Larry M Manevitz and Malik Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.

[43] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

Appendix A

Complexity Tuning of a One Class Support Vector Machine(OCSVM)

A.1 Introduction

This section provides another method for using complexity to improve modeling of devices. This section shows work done to tune the search of a One Class Support Vector Machine (OCSVM) [42] to improve false positive rates for devices measured to be of low complexity. This is important as established in the main body of this document, low complexity devices have a well defined behavior footprint, and any error that leads to blocking of legitimate traffic will have a larger impact on the device's functionality as it will affect a larger percentage of the normal traffic than a false positive error would on a highly complex device.

A.2 Methodology

Novelty detection is a form of outlier detection where the training set is considered untainted by outliers i.e. only positive samples. New observations are classified and determined to fall within the decision boundary are inliers and observations outside the decision boundary are outliers. To derive a behavior for a device we employ the OCSVM algorithm using the Radial Basis Function (RBF) kernel. Outlier flows detected during the training phase are recorded, and form the set of flows we call significant flows.

Definition A.2.1. Significant Flow: A *significant flow* is one that is marked as an outlier by the OCSVM during training. This set of flows plus the decision boundary forms the behavior boundary of the device.

Definition A.2.2. Device Behavioral Boundary: *Device behavioral boundary* is the set of all unique significant flows and the decision boundary found during training.

A.2.1 Device IP Complexity

This research examines how devices form connections. Simply counting the number of unique IP addresses that a device connects with fails to take into account the inherent service-oriented hierarchical structure of the IPv4 address space, where companies and services are often part of similar subnets. To account for this I propose a complexity measure based on a simple ratio of *IP spread* to *IP depth*. IP spread is the number of unique source and destination IP addresses that interact with the device. IP depth is the number of IP addresses that belong to the same higher level octets.

Definition A.2.3. IP Tree, IP Branch, IP Leaf, IP Spread An *IP tree* is a unique first order octet which comprises the root of the tree. An *IP Branch* is a second or third order octet that has one or more fourth order octets under it. An *IP leaf* is a unique fourth order octet. *IP Spread* is the sum of total unique IP addresses that interact with a device.

Device IP Spread (IP_{Spread})

$$IP_{Spread} = \sum IP_{trees} \tag{A.1}$$

Device IP Depth (IP_{Depth})

$$IP_{Depth} = \frac{\sum IP_{leaf}}{\sum IP_{branch}} \tag{A.2}$$

Device IP Complexity (d_{ip})

$$d_{ip} = \frac{IP_{Spread}}{IP_{Depth}} \tag{A.3}$$

To calculate IP spread and depth we build unordered trees of each IP address where the first order octet is the root and lower octets are children. Then we can calculate how many trees, branches and leaf nodes each IoT device contacts. A large number of IP trees with few branches indicates a large IP spread. A small number of IP trees that have many branches and leaves indicates a large IP depth. IP depth/spread is used as one measure of a device's complexity. Devices belonging to a single ecosystem such as Google Home have a small number of broad trees as they connect to

mostly Google's networks dedicated to these types of devices. Other devices such as laptops and smart phones make connections to many unique destinations thus leading to a large number of thin trees each having fewer branches and leaves. Figure 4.2 shows the total IP complexity of each device.

A.2.2 Novelty Detection Tuning Using Device Complexity

To test the efficacy of the discovered model each device was analyzed against seven common forms of malware listed in 3.6. This malware is part of a dataset published by Stratosphere Laboratory [31] and contains 20 captures where malware was executed on IoT devices, and 3 captures for benign IoT devices traffic.

The OCSVM using the RBF kernel is governed by the two hyper-parameters, ν (nu) and γ (gamma). Gamma sets the radius of the RBF kernel by determining the influence of each example of the decision boundary and ν sets the upper bound on fraction of errors during training and the lower bound on the fraction of support vectors used. For the purposes of this work we set this using the 'scale' option of Sci-kit learn which uses the following equation to determine gamma.

$$\gamma = \frac{1}{n\_features * x.var()} \tag{A.4}$$

To establish that the complexity measurement is a statistically relevant one we take the linear regression of the number of outliers found by the OCSVM using the default values of $\nu = 0.5$ and gamma as calculated in equation A.4. The correlation of outliers to anomalies can be seen in Figure A.1.

This research examines three methods to establish ν for devices; static ν set uniformly across all devices; a dynamic ν set per device, and a ν tuned to the complexity of the device.

A.2.3 Static Hyper-Parameter ν

For the static method an average ν is found and applied uniformly across all the device models. To find the average ν, each device was modeled using OCSVM with ν varied over the range of

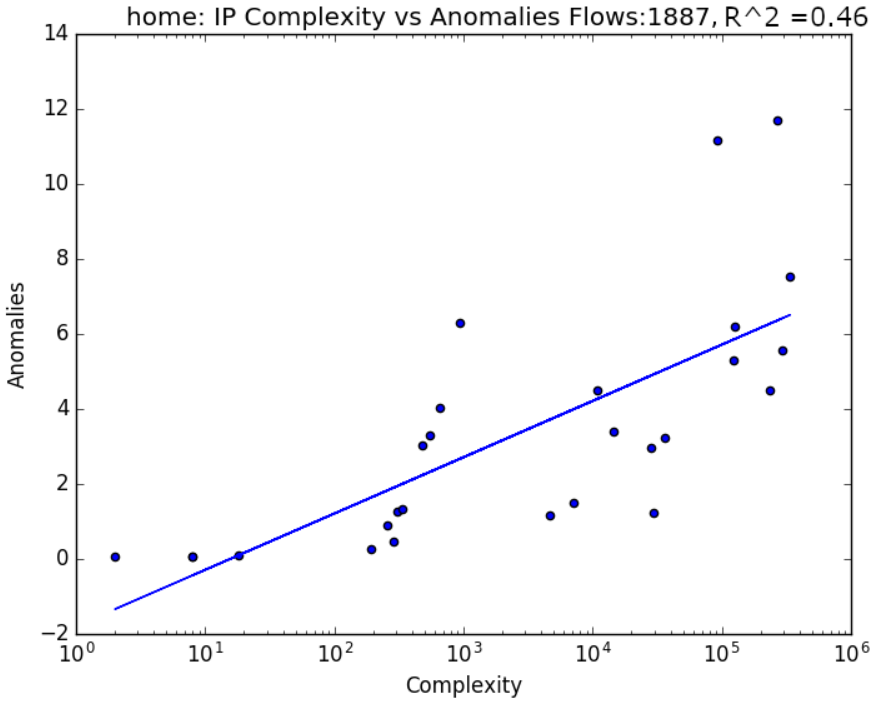


Figure A.1: Complexity Vs Anomalies

0.00001 to 0.5. The ν for each device that had the best F1 score was saved and the mean ν value was calculated across all the devices. This average ν was then used to train the model for each device and test for anomalies. This gives a baseline model where there is a balance between precision and recall and where the hyper-parameter ν is set to the same value for each device.

A.2.4 Dynamic Hyper-Parameter ν

The dynamic method finds the best ν for each device and that ν is applied individually to each model. To find the ν value for individual devices, each device was modeled using OCSVM with ν varied over the range of 0.00001 to 0.5. The ν for each device that had the highest F1 score was then used to train the model for that device and test for anomalies. This gives results that balance precision and recall and a model that is tuned per device.

A.2.5 Complexity-Tuned Dynamic ν

To tune the model based on complexity a value for ν is found that minimizes false positives for low complex devices. To find a ν value that is tuned to the complexity of the device, each device was modeled using OCSVM with ν varied over the range of 0.00001 to 0.5. The ν for each device that had the highest F_β scores where $\beta = \hat{A}_{DC}$, where $\hat{A}_{DC}$ is the normalized value (between 0 and 1) of the aggregate device complexity as defined in section 3.4.1. This search prioritizes minimizing false negatives on low complexity devices as seen in equation A.5.

$$F_\beta = (1 + \beta^2)\frac{precision * recall}{(\beta^2 * precision) + recall} \tag{A.5}$$

A.3 Home Results

This section shows the results for the tuned OCSVM using the seven malware traffic flows.

Figure **??** show the home dataset against the C&C malware attack plotted as false positive rate vs. complexity score. Optimization is shown from left to right, untuned, search, and tuned-search.

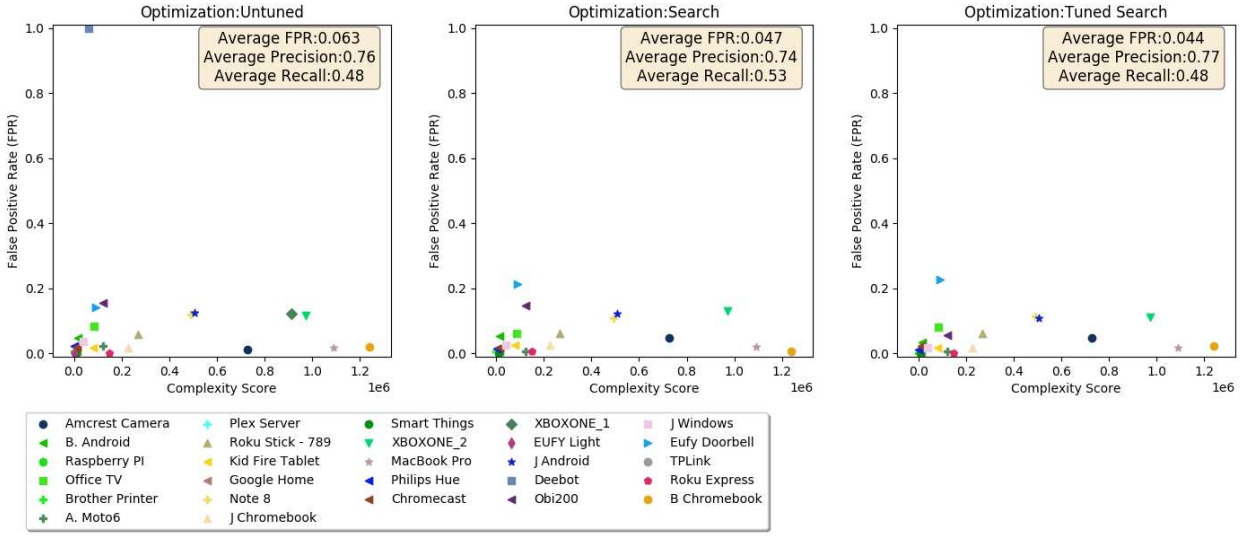


Figure A.2: Home: C&C: Flows

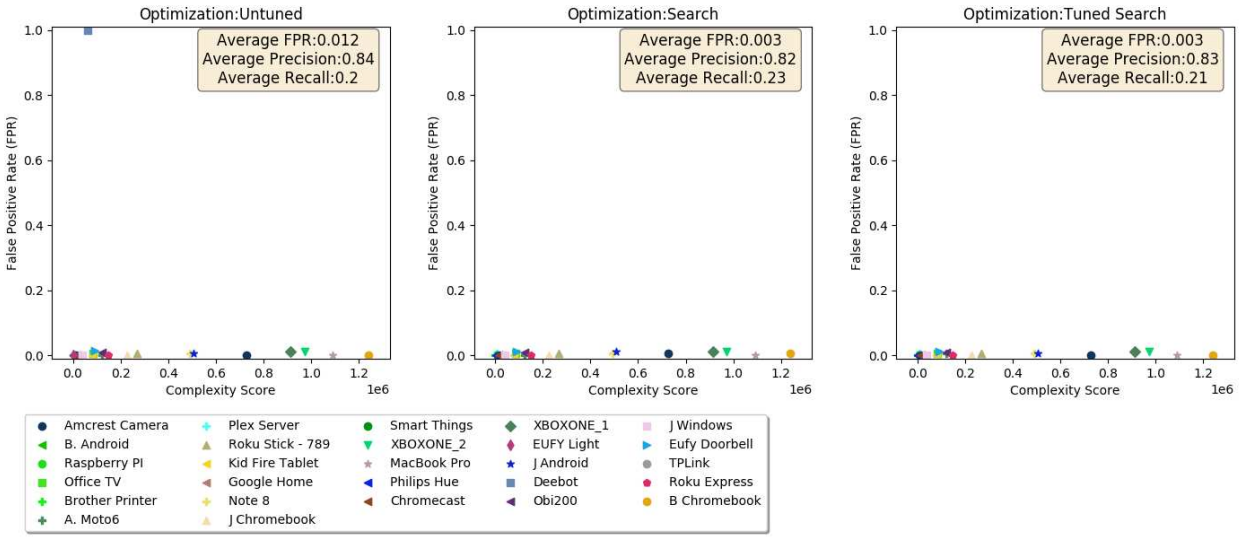


Figure A.3: Home: C&C Heartbeat: Flows

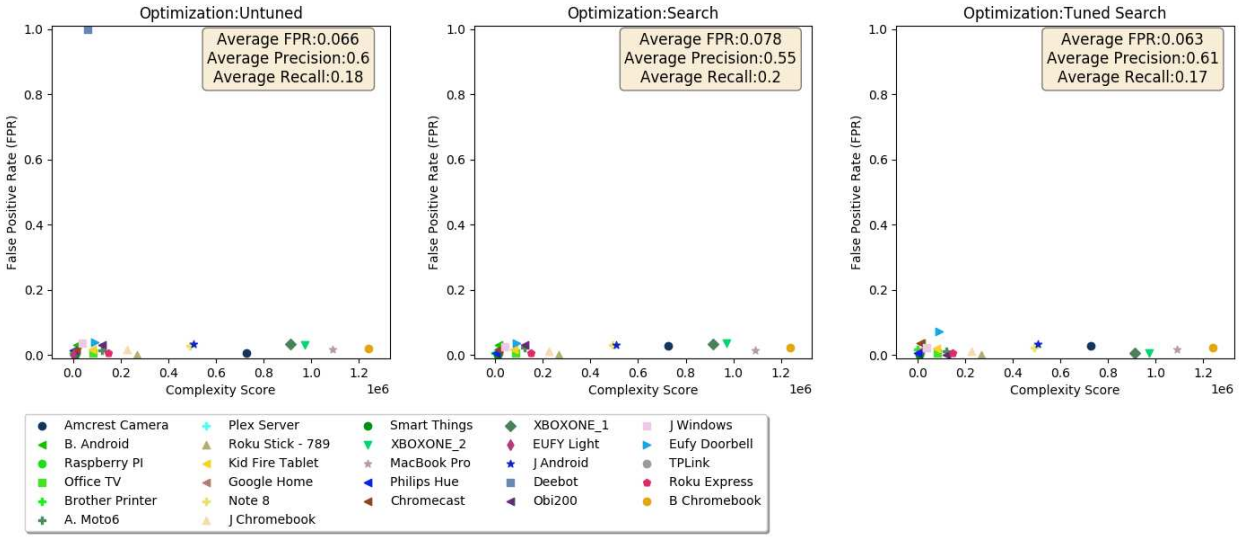


Figure A.4: Home: C&C FileDownload: Flows

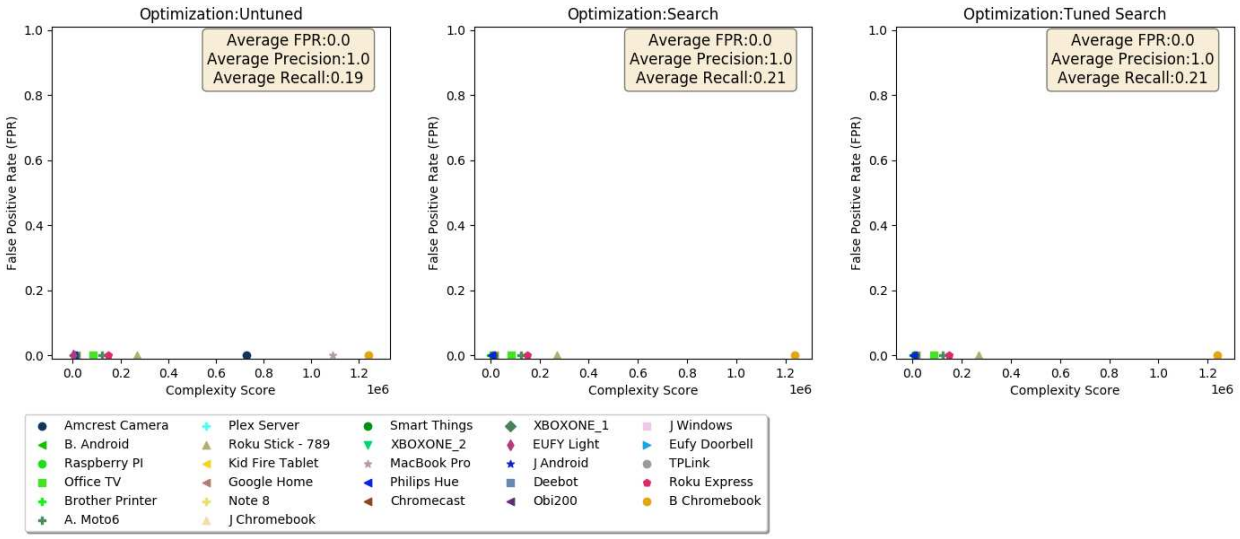


Figure A.5: Home: DDoS: Flows

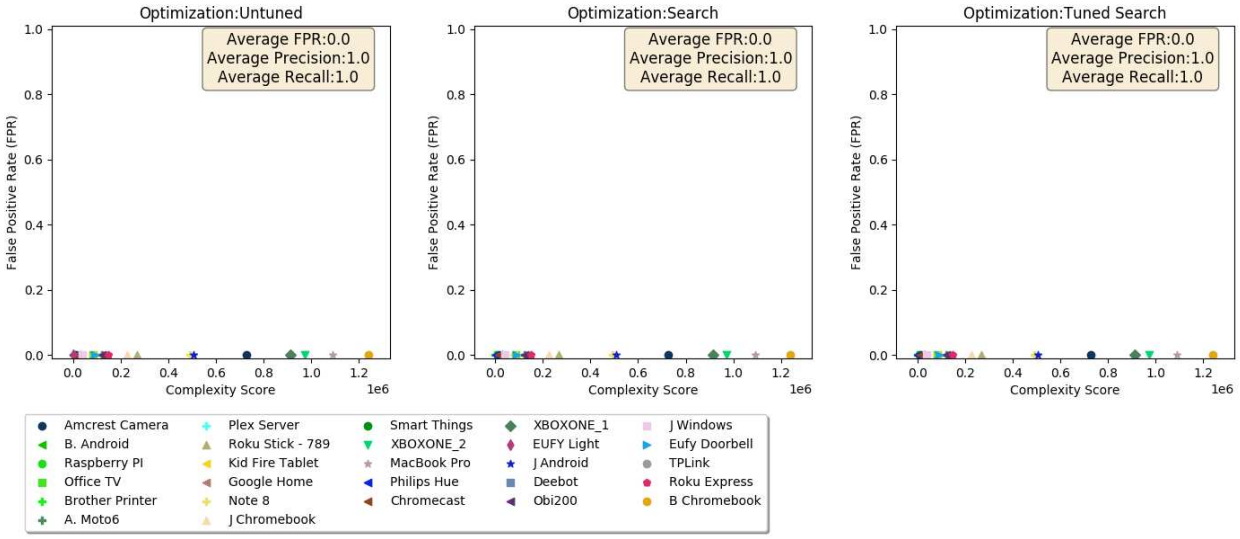


Figure A.6: Home: Okiru: Flows

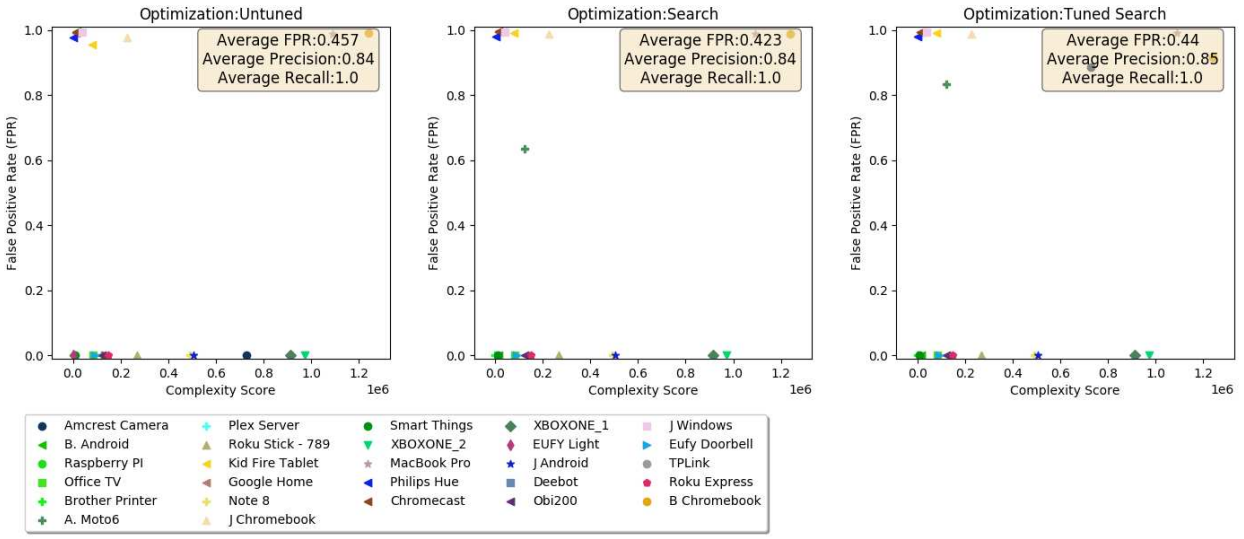


Figure A.7: Home: Horizontal Port Scan: Flows

A.4 Lab Results

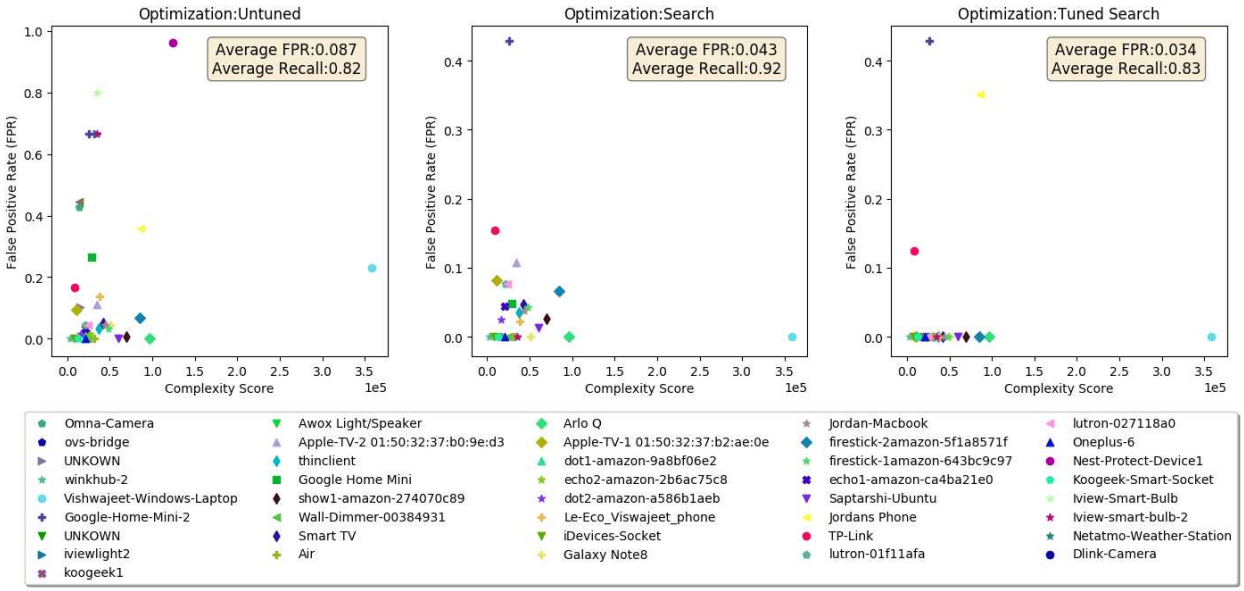


Figure A.8: Lab: C&C: Flows

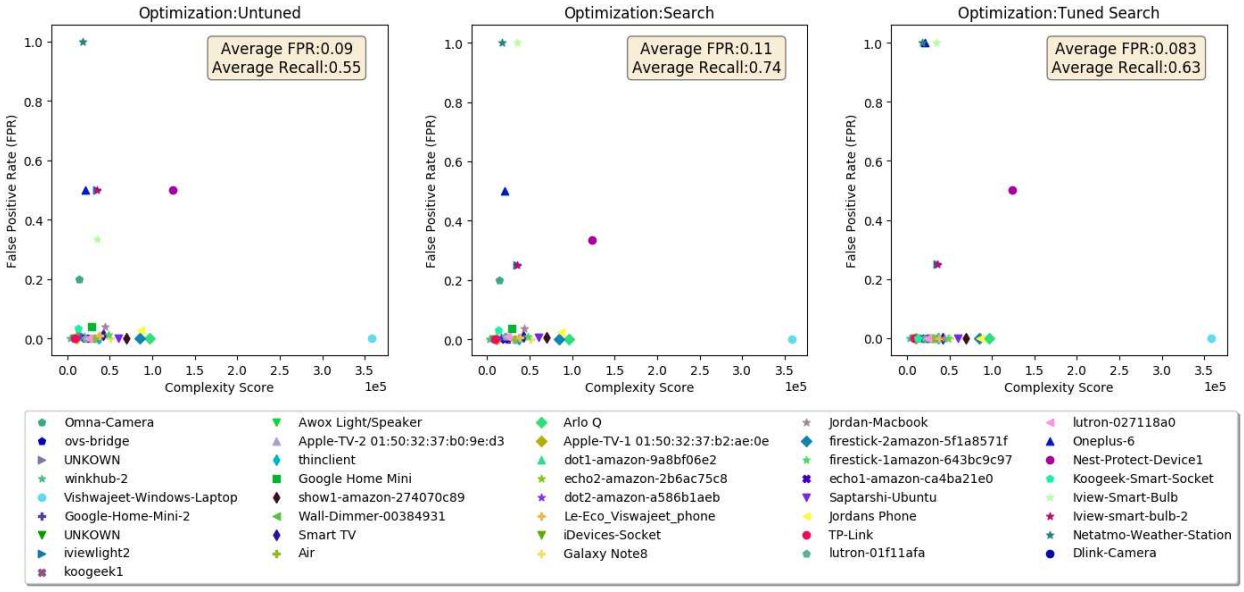


Figure A.9: Lab: C&C Heartbeat: Flows

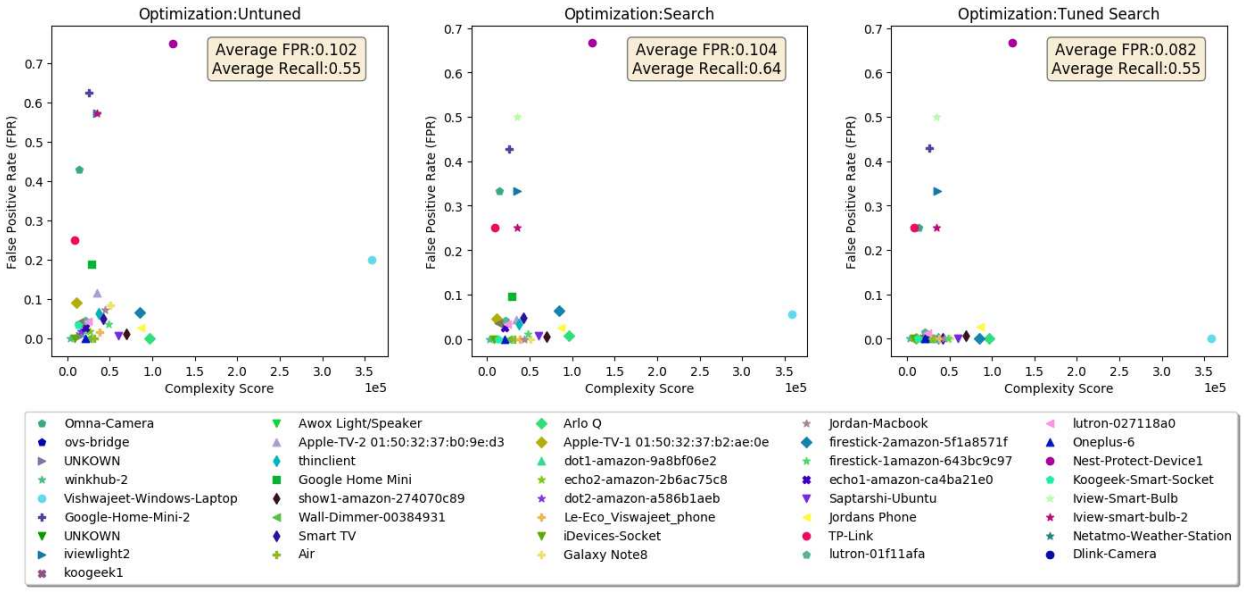


Figure A.10: Lab: C&C FileDownload: Flows

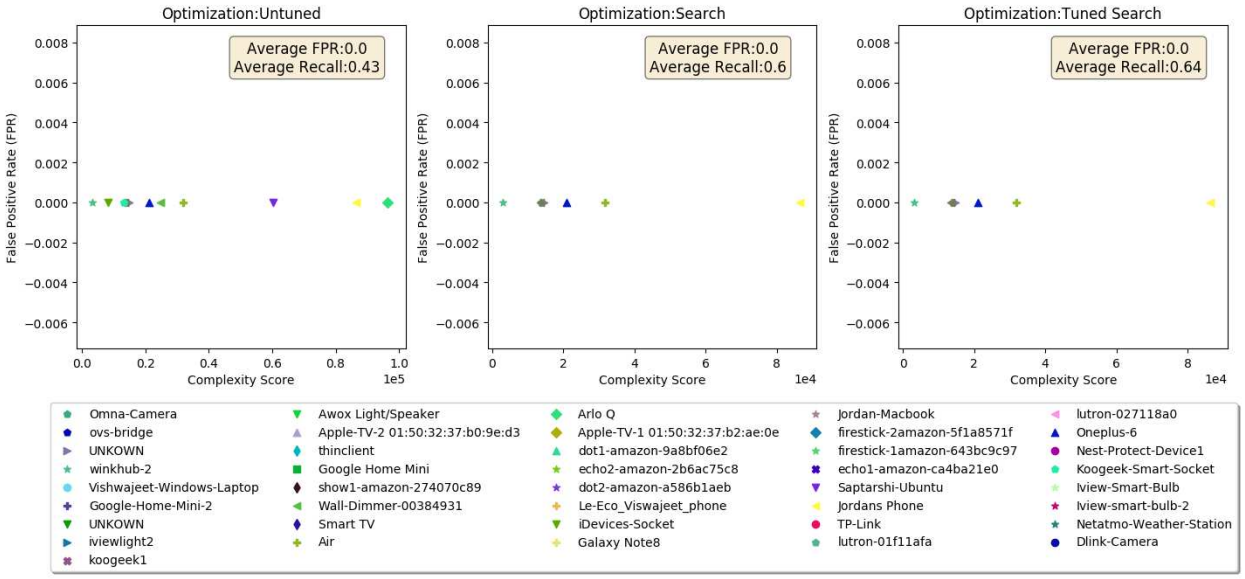


Figure A.11: Lab :DDoS: Flows

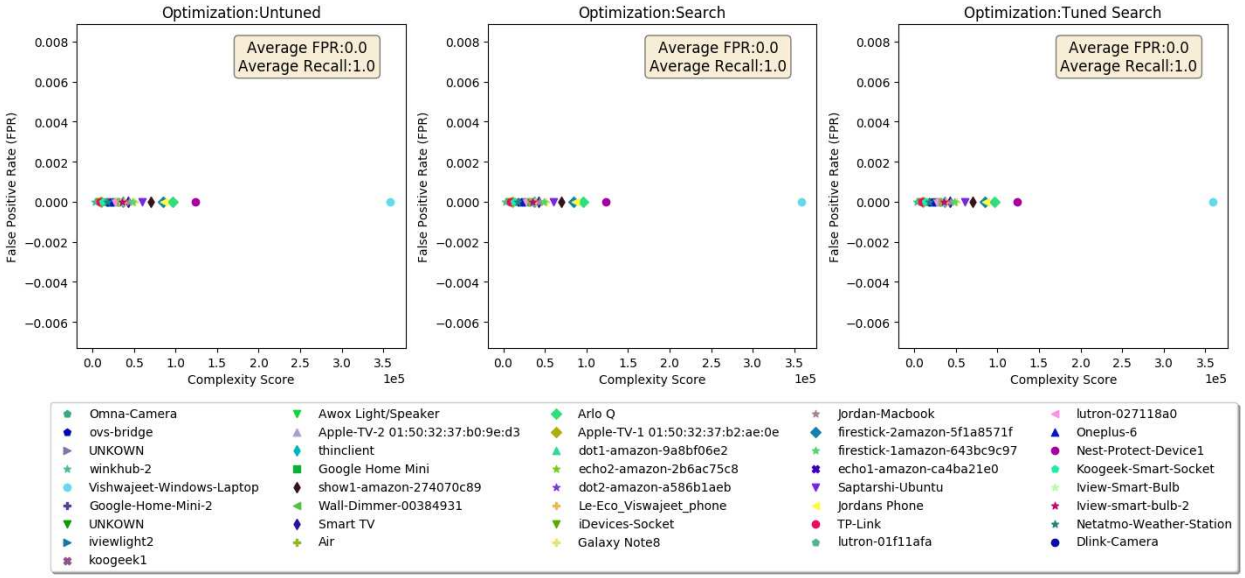


Figure A.12: Lab: Okiru: Flows

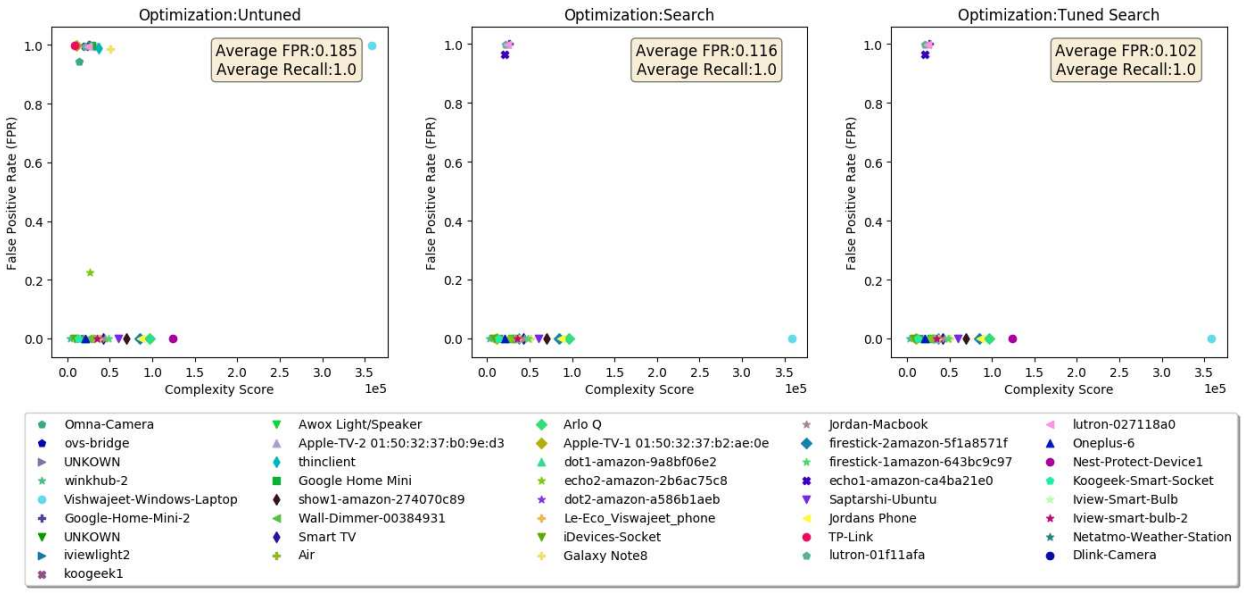


Figure A.13: Lab: Horizontal Port Scan: Flows

A.5 UNSW Results

The following Figures [A.14,A.15,A.16,A.18,A.19] show how each device in the UNSW dataset against the five does against the five types of malware traffic when using a OCSVM and tuning the hyperparameters for lower false positive rates on devices that are of lower complexity.

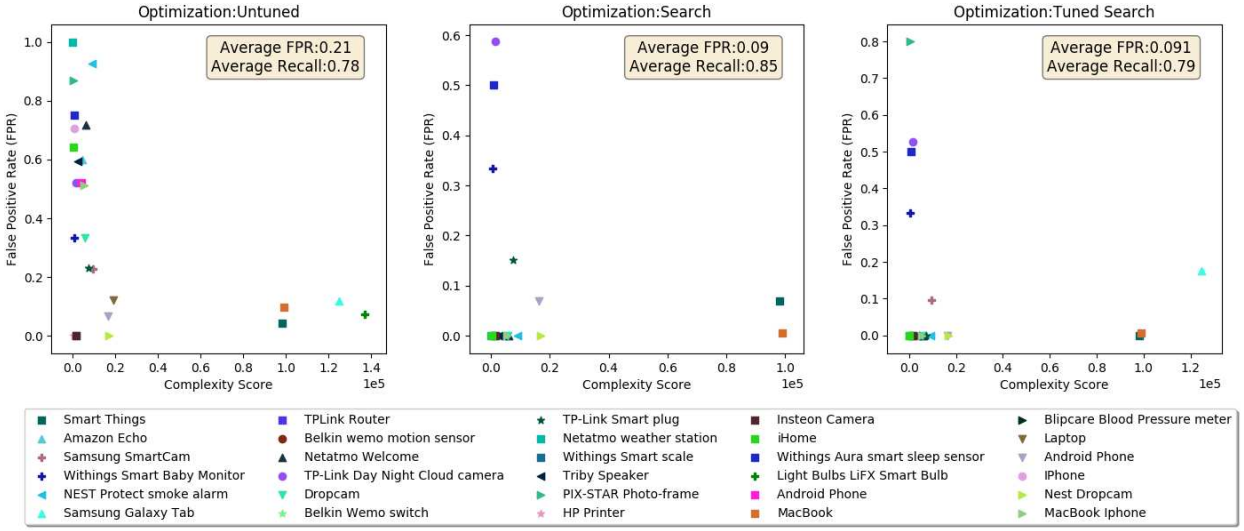


Figure A.14: UNSW: C&C: Flows

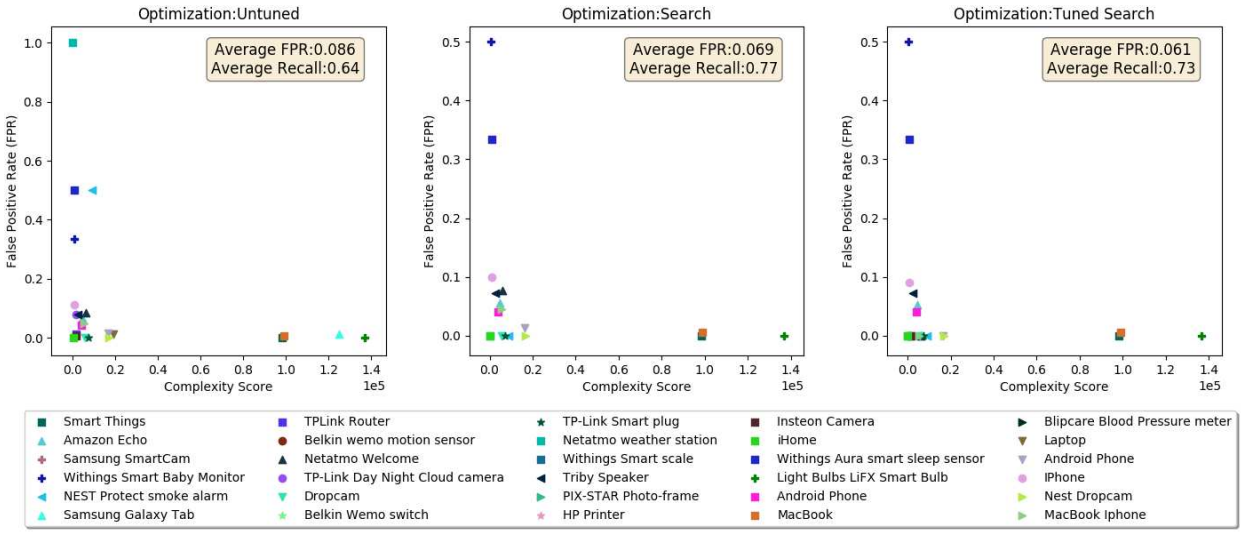


Figure A.15: UNSW: C&C Heartbeat: Flows

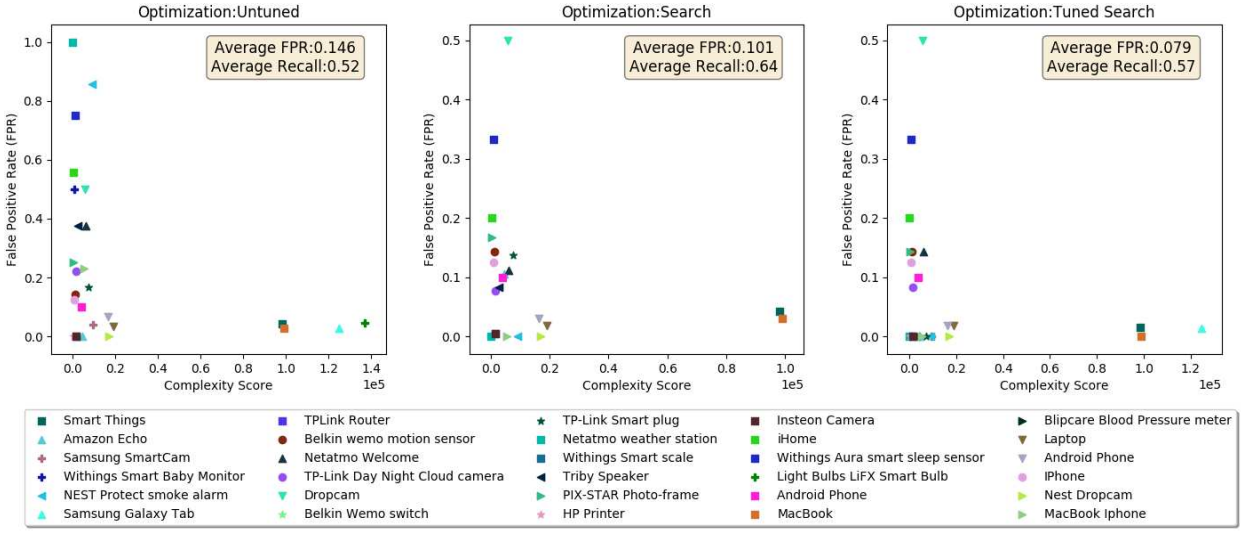


Figure A.16: UNSW: C&C FileDownload: Flows

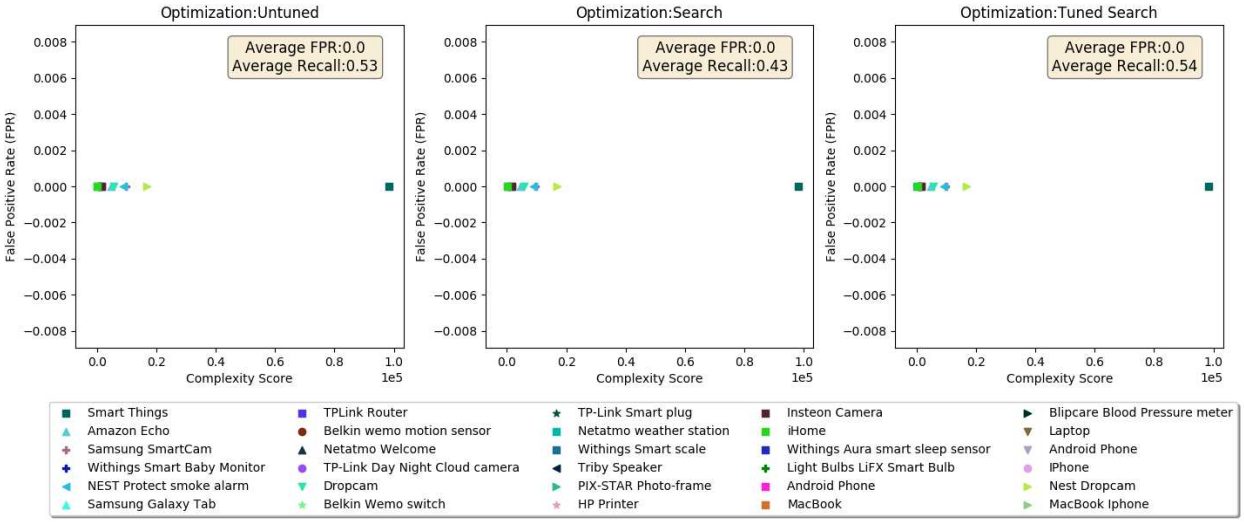


Figure A.17: UNSW: DDoS: Flows

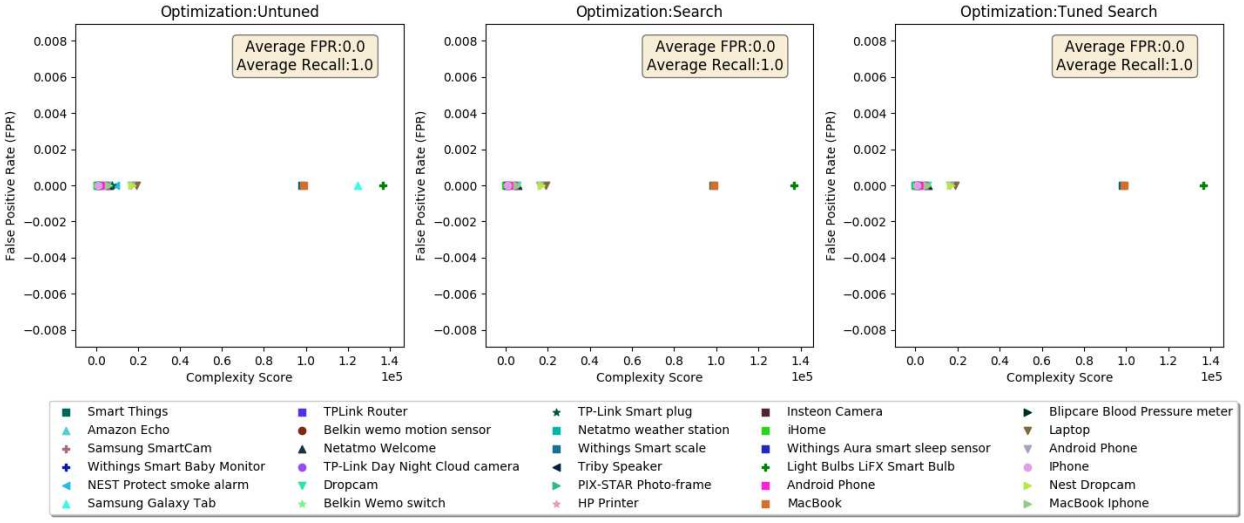


Figure A.18: UNSW: Okiru: Flows

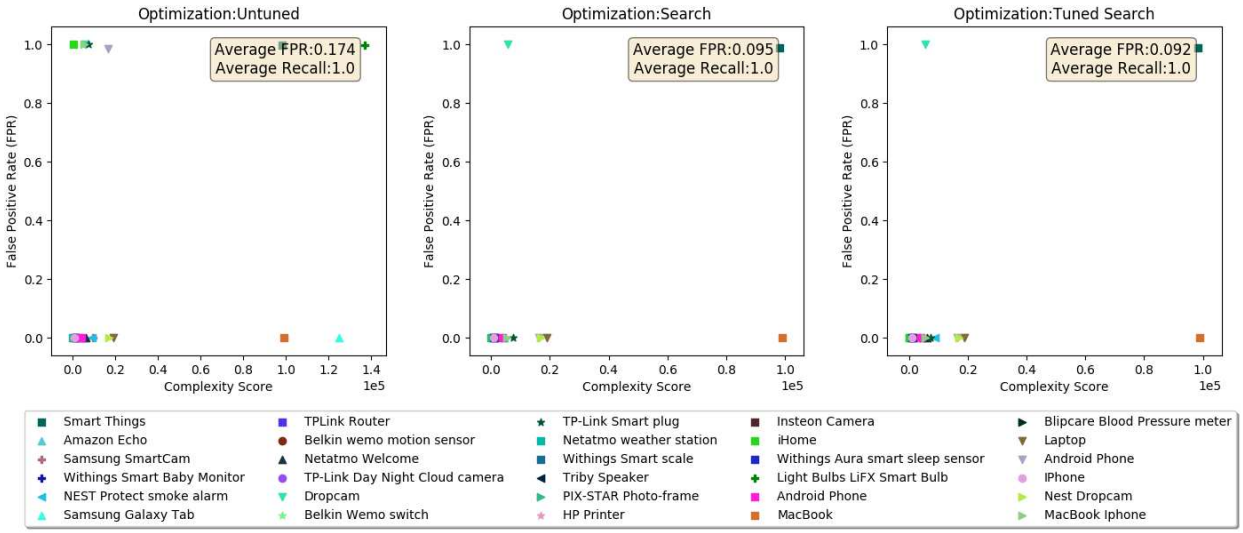


Figure A.19: UNSW: Horizontal Port Scan: Flows

A.6 SCADA Results

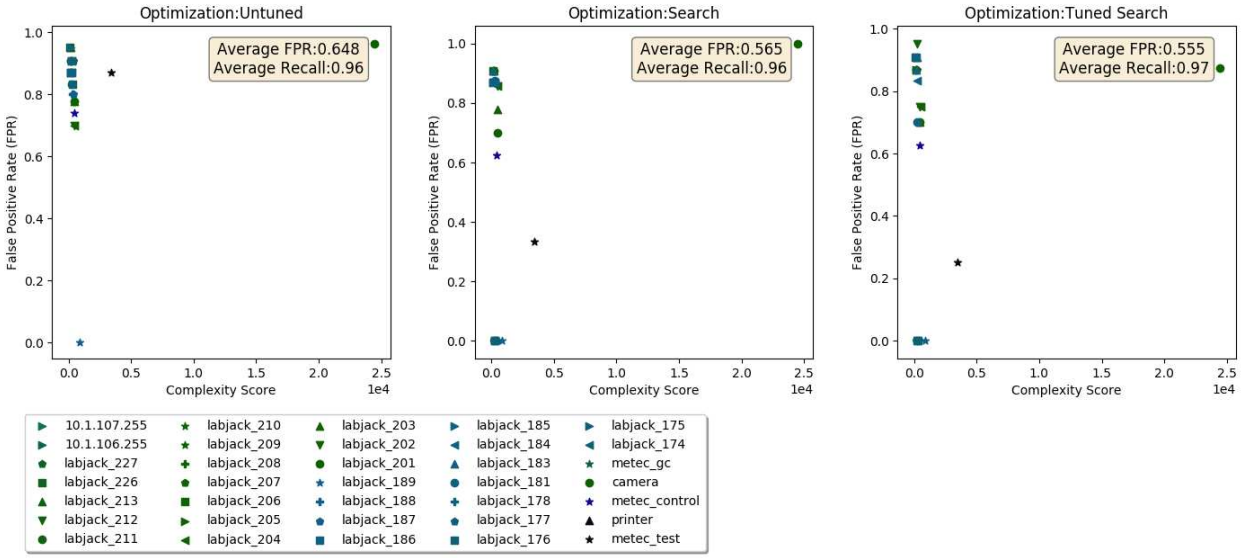


Figure A.20: SCADA: C&C: Flows

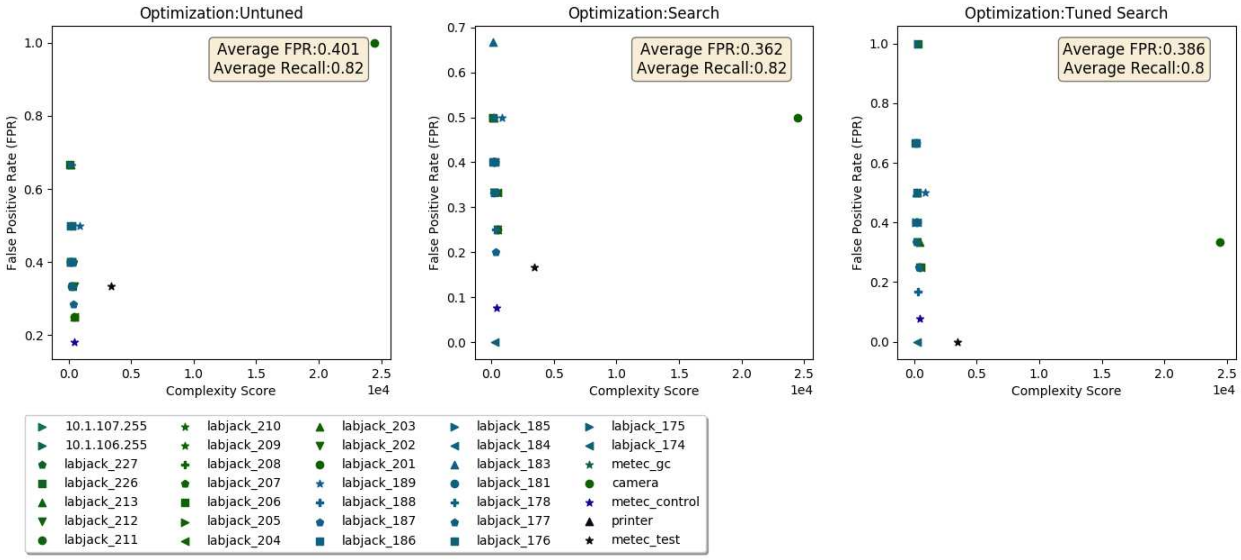


Figure A.21: SCADA: C&C Heartbeat: Flows

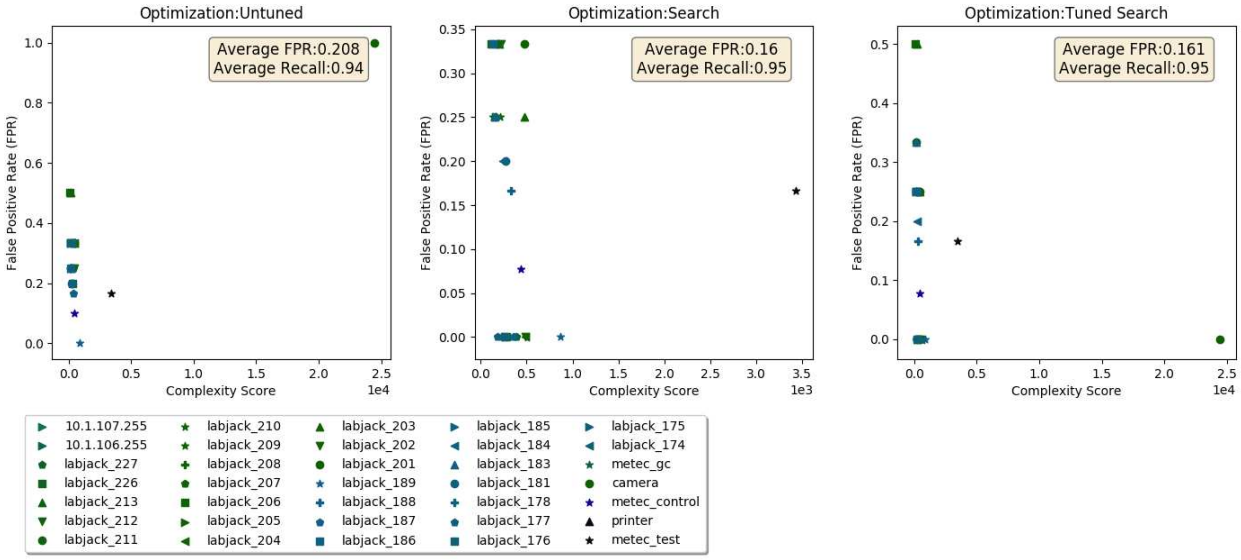


Figure A.22: SCADA: C&C FileDownload: Flows

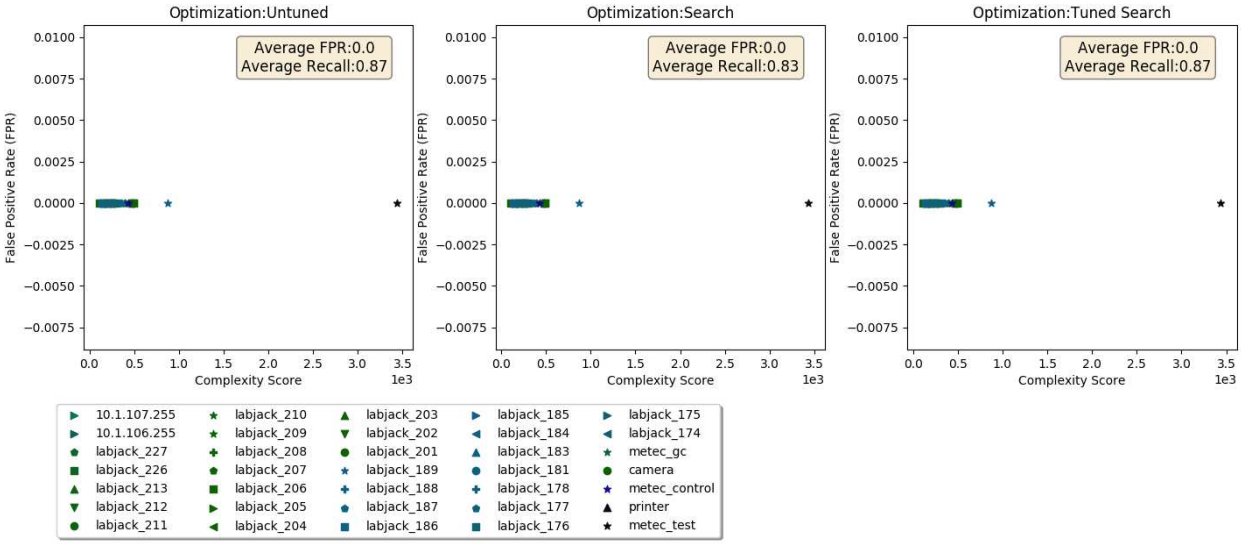


Figure A.23: SCADA: DDoS: Flows

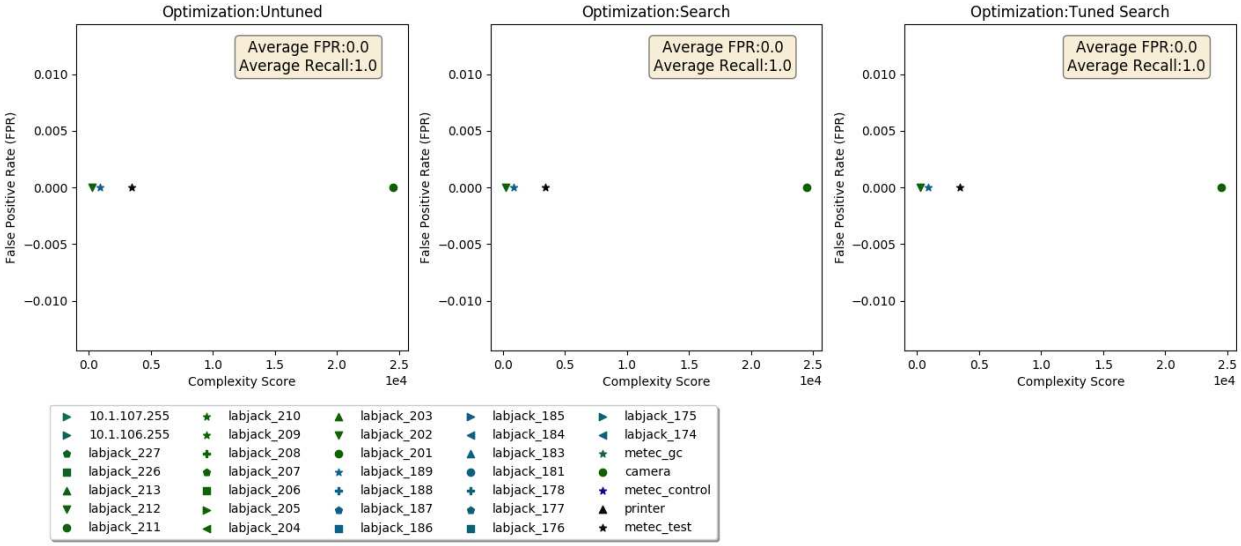


Figure A.24: SCADA: Okiru: Flows

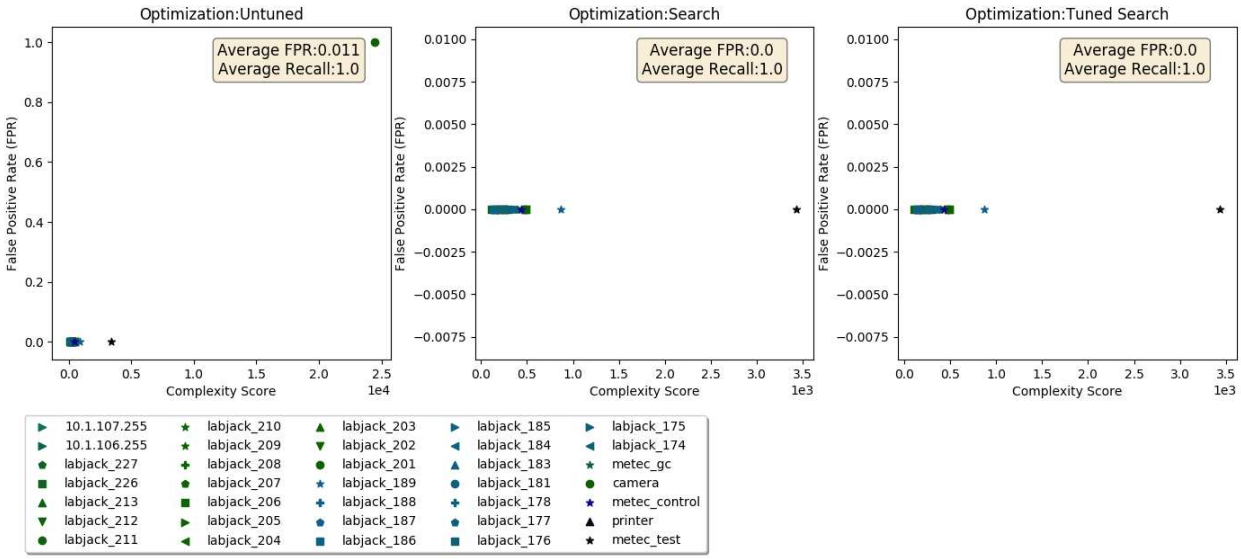


Figure A.25: SCADA: Horizontal Port Scan: Flows

Appendix B

Home

This Appendix shows the Gaussian Boundary for each device in the home dataset against all seven attacks listed in 3.6.

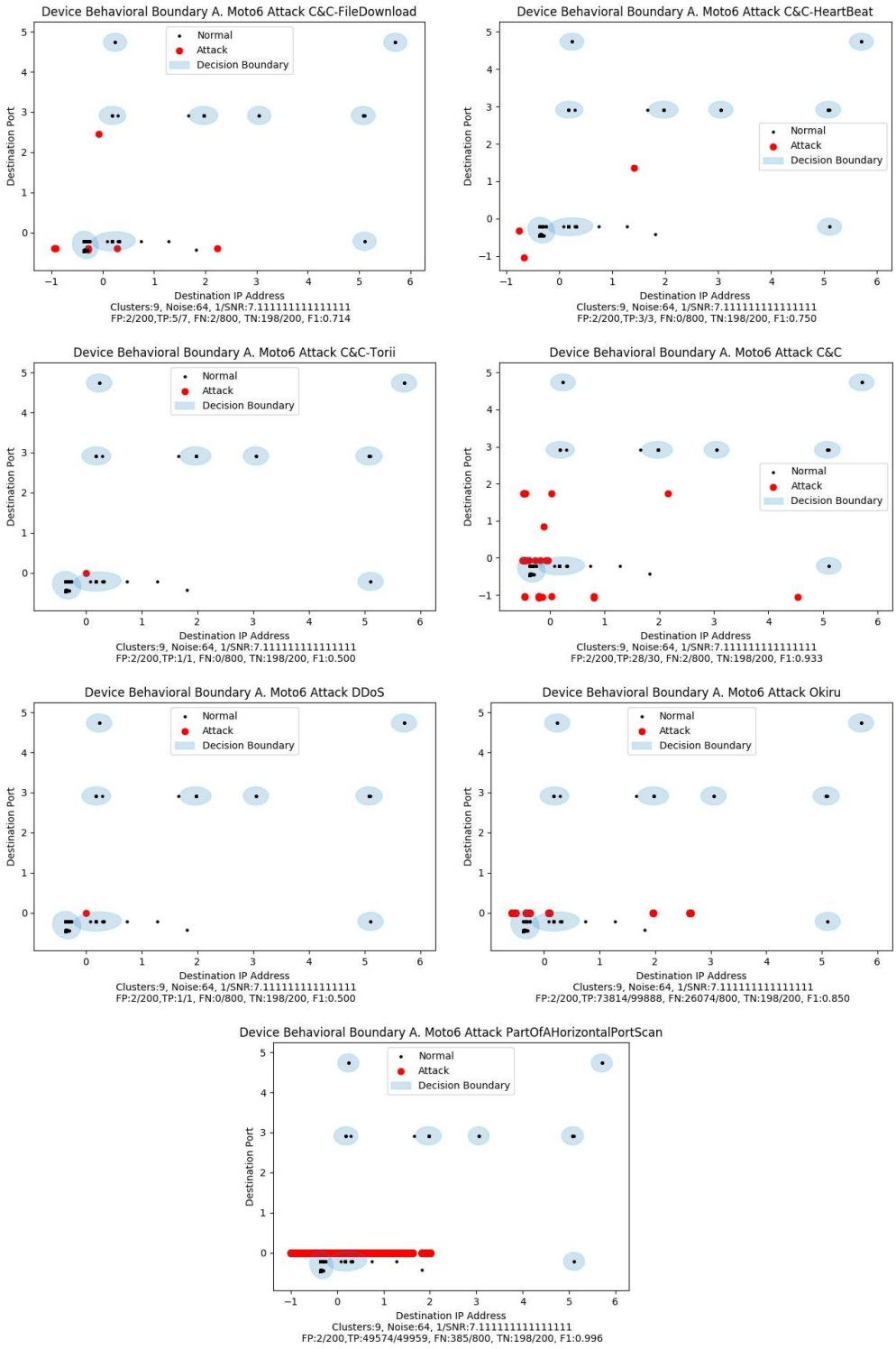


Figure B.1: A-Moto6: HOME

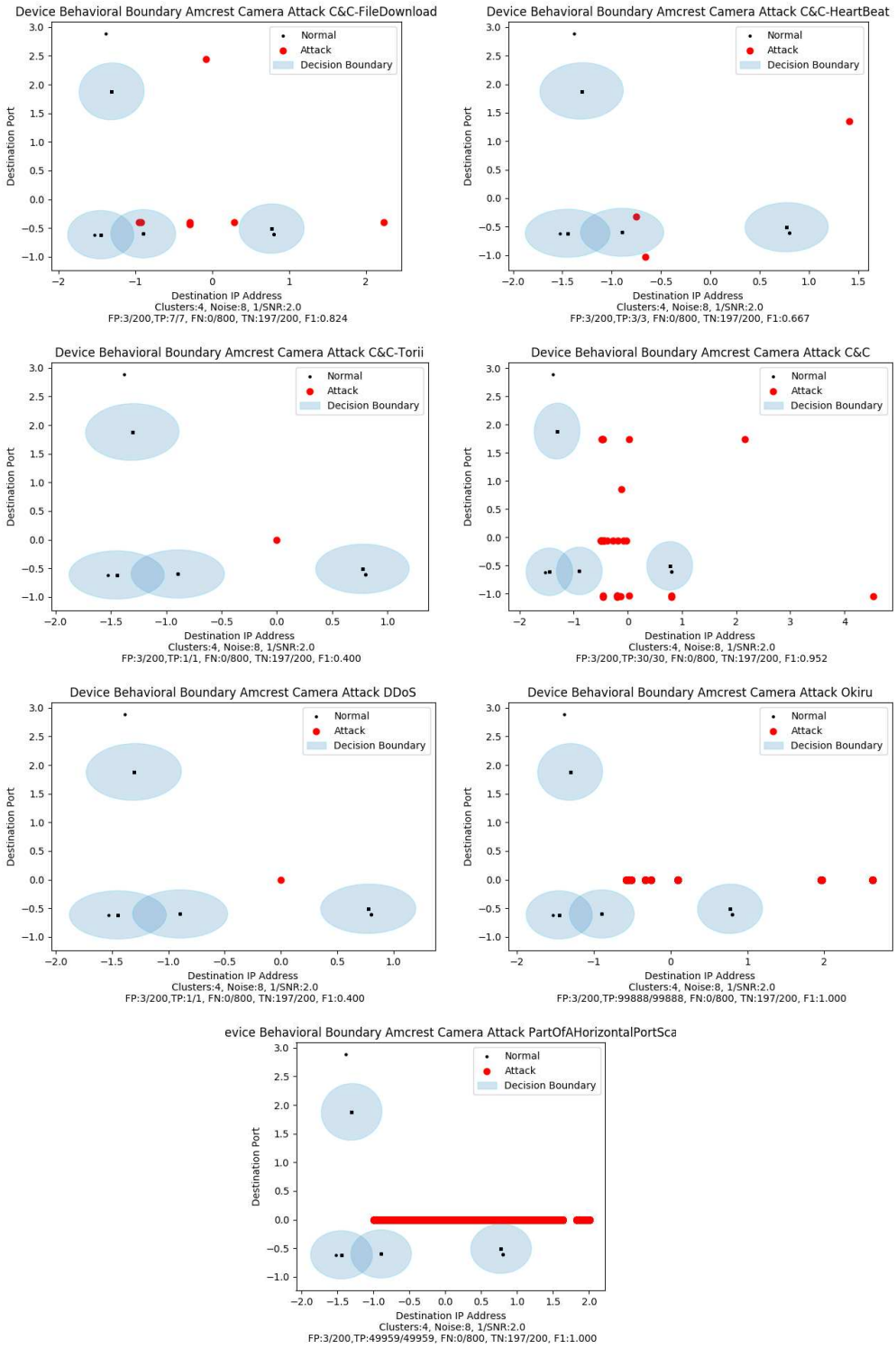


Figure B.2: Amcrest-Camera: HOME

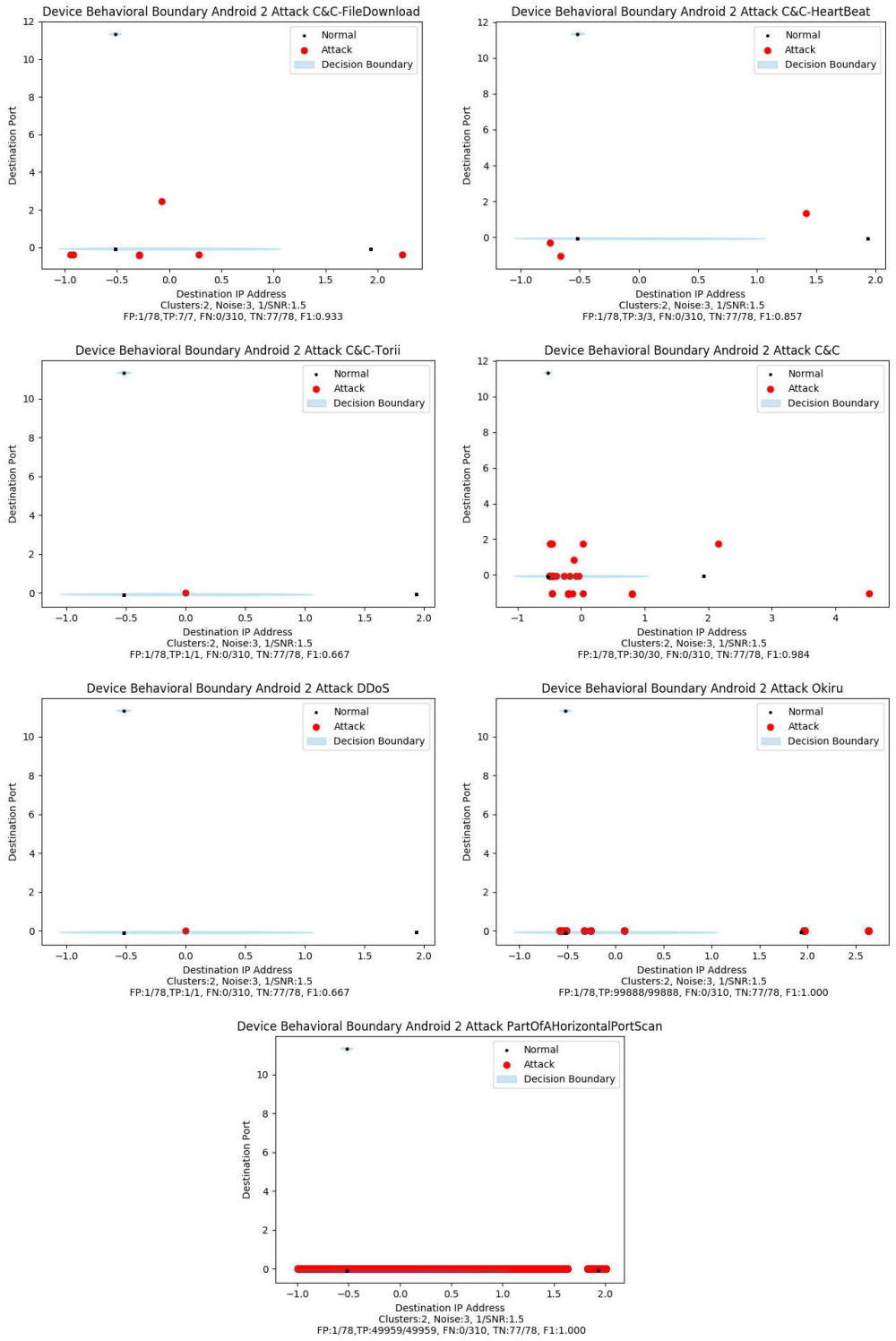


Figure B.3: Android-2: HOME

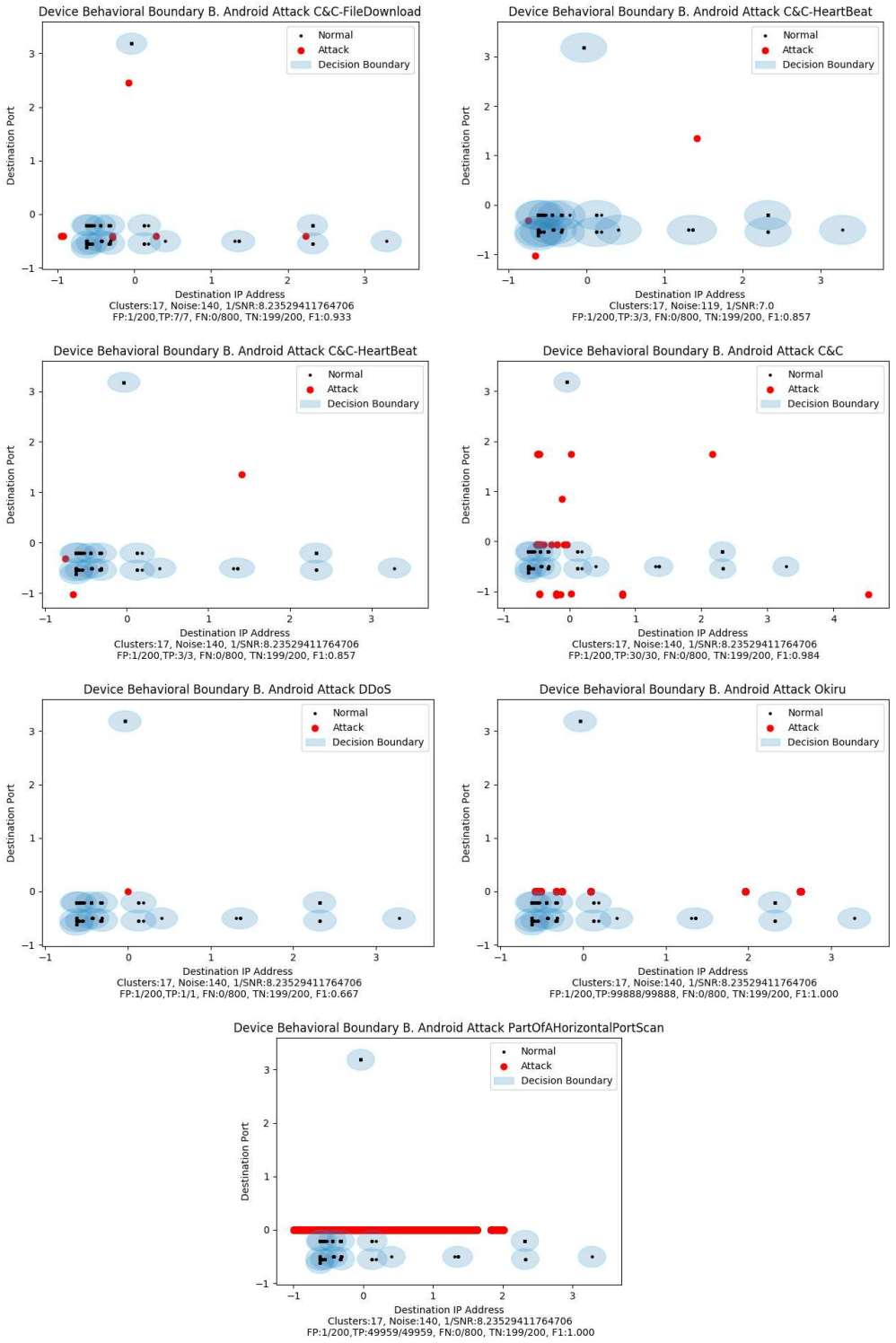


Figure B.4: B-Android: HOME

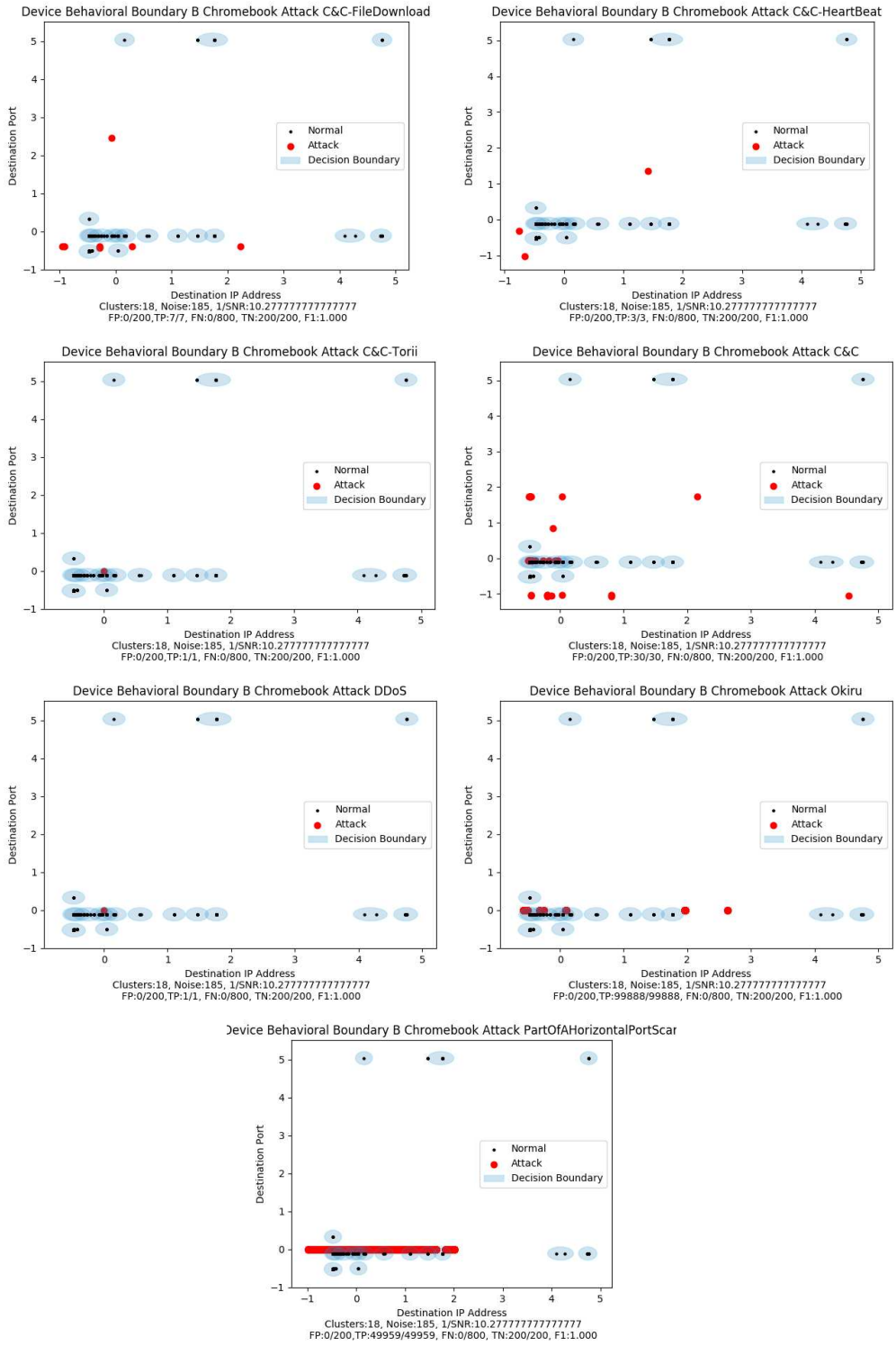


Figure B.5: B-Chromebook: HOME

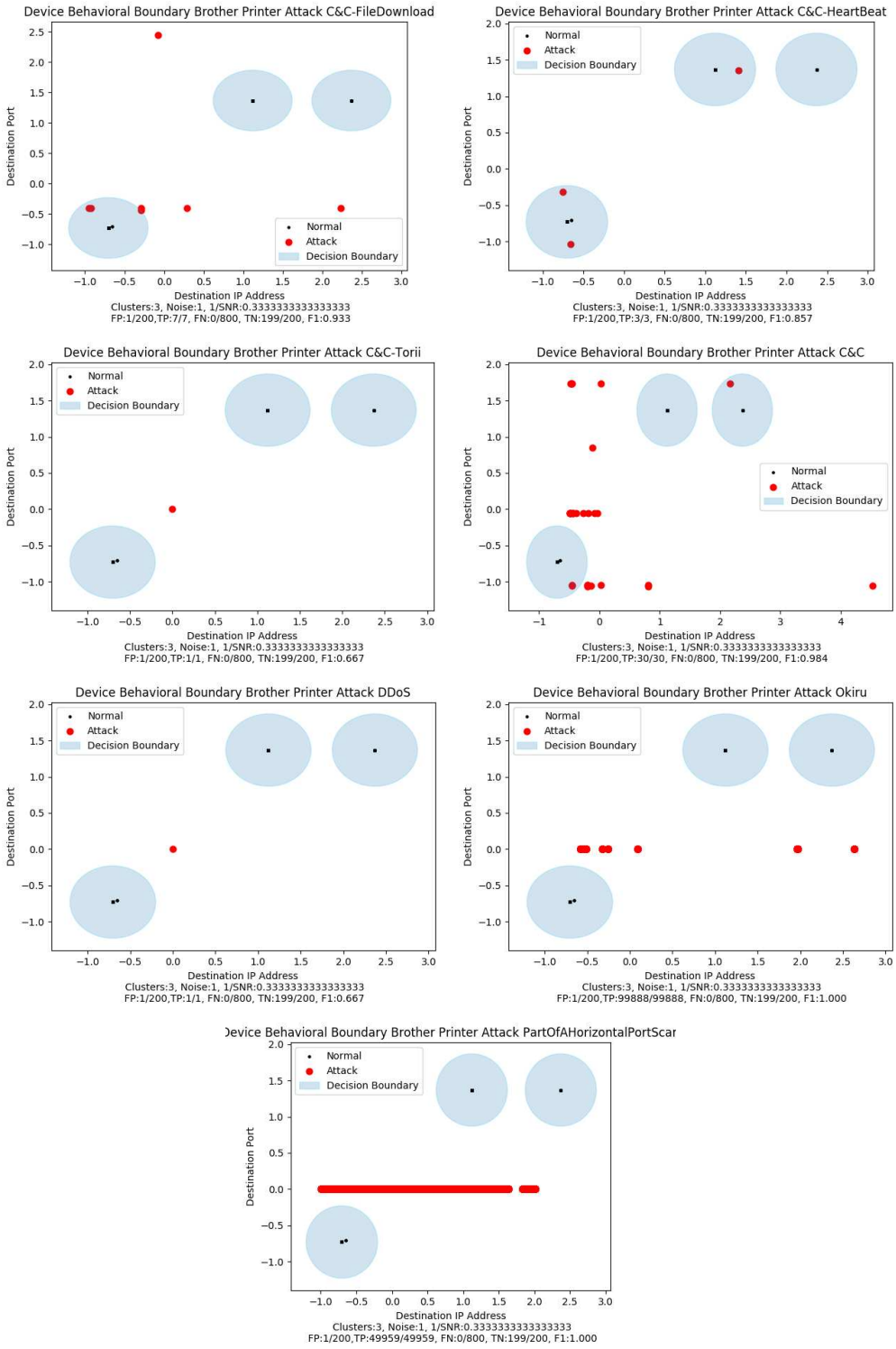


Figure B.6: Brother-Printer: HOME

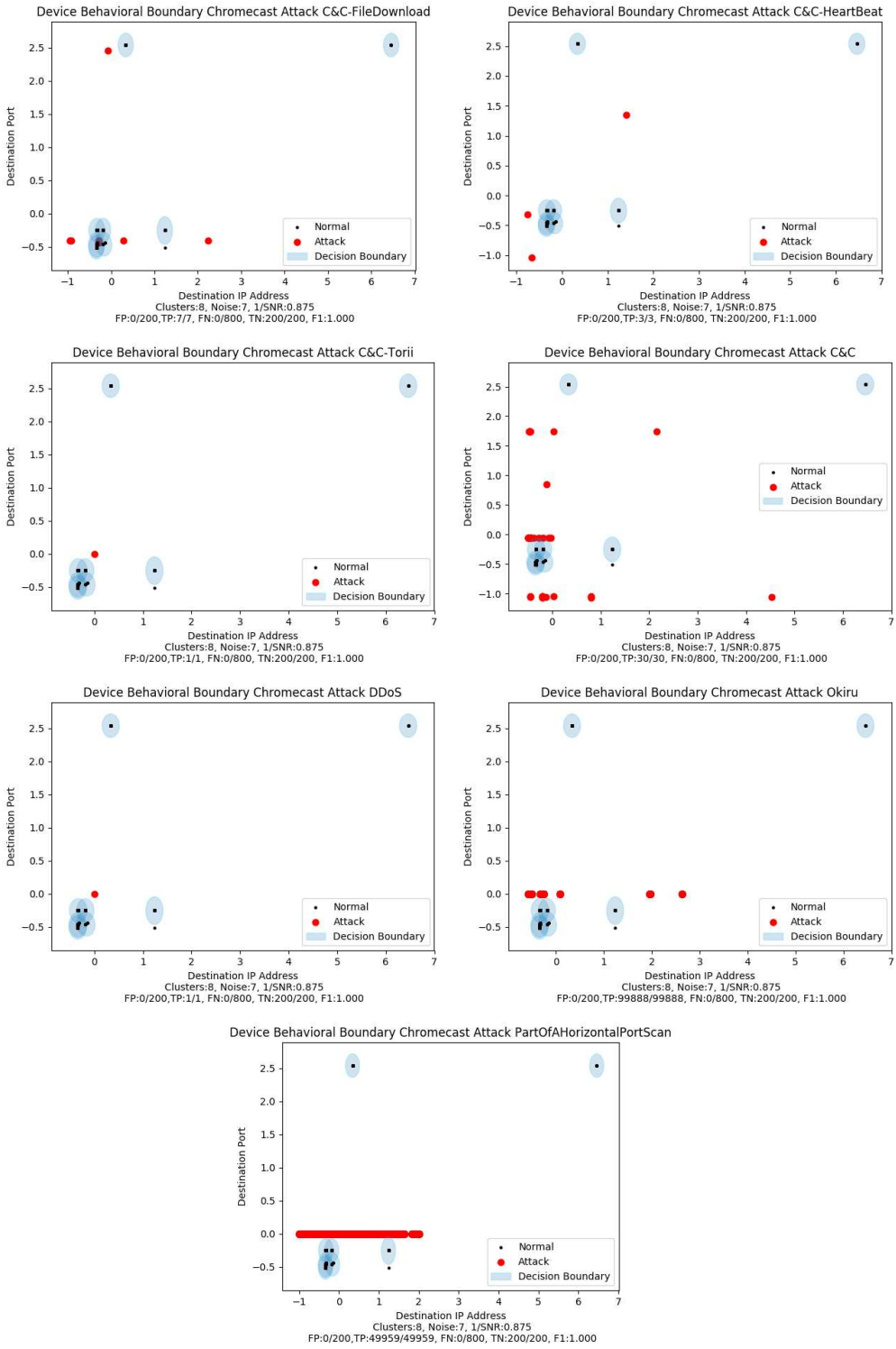


Figure B.7: Chromecast: HOME

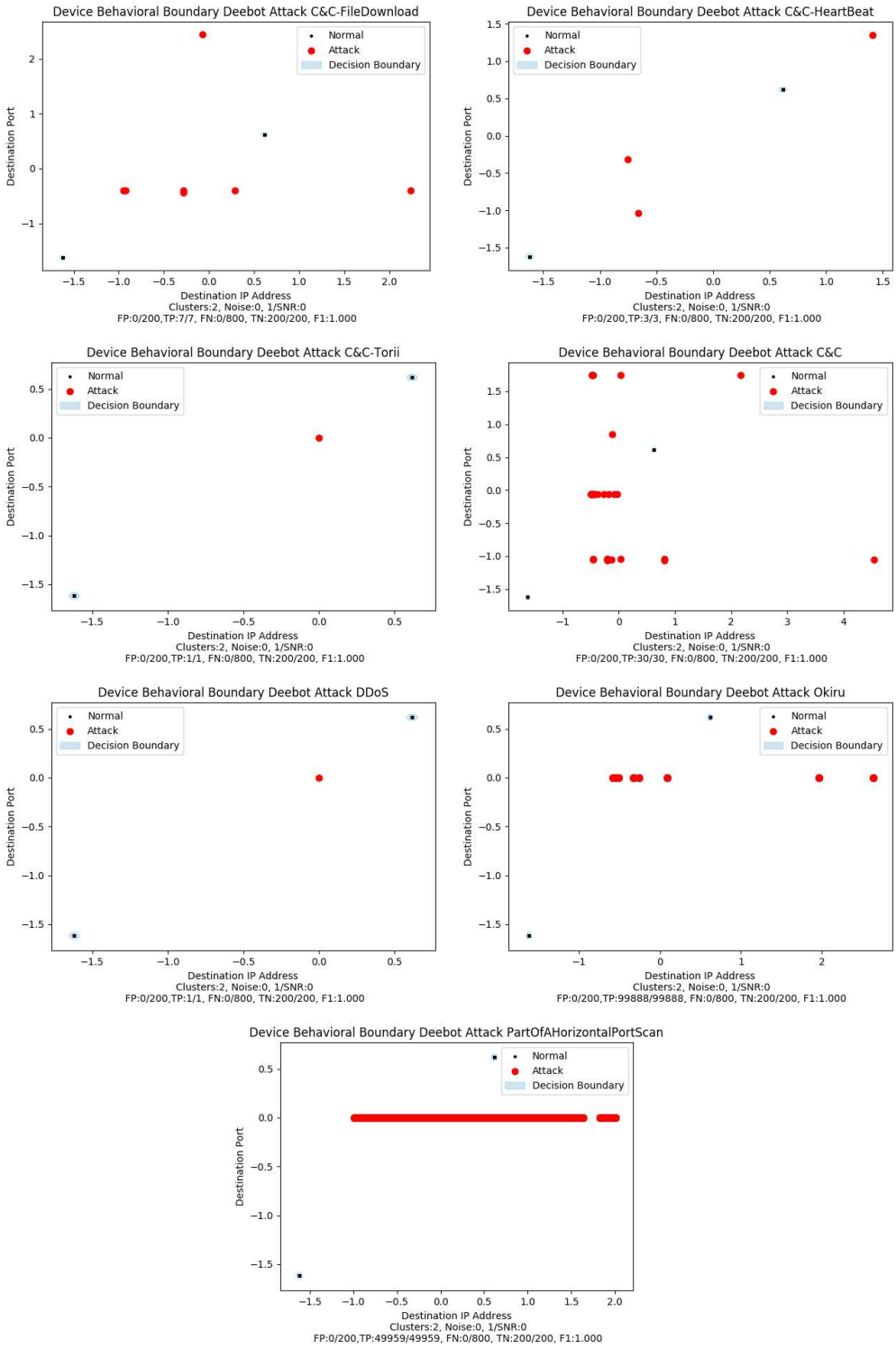


Figure B.8: Deebot: HOME

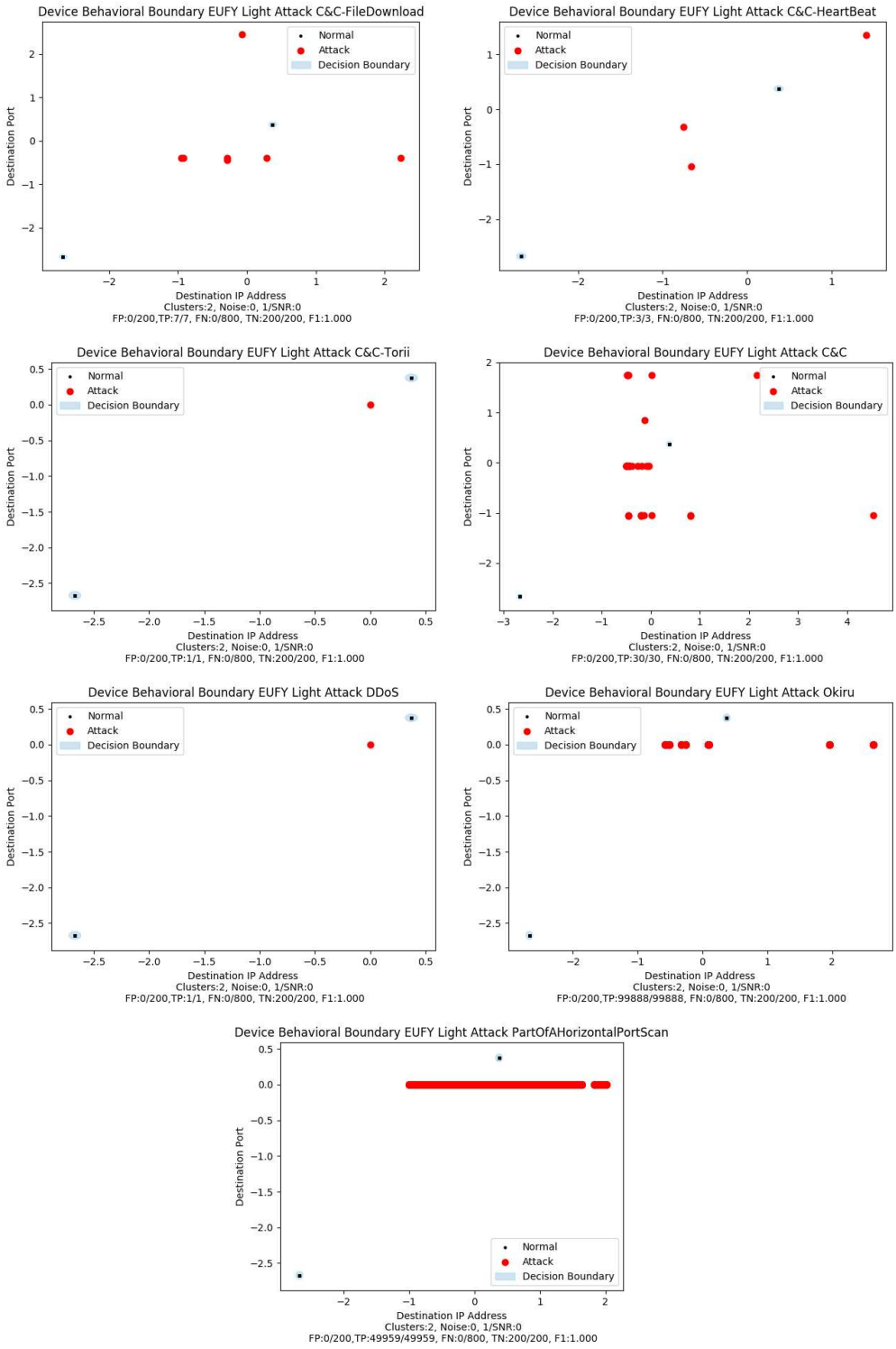


Figure B.9: EUFY-Light: HOME

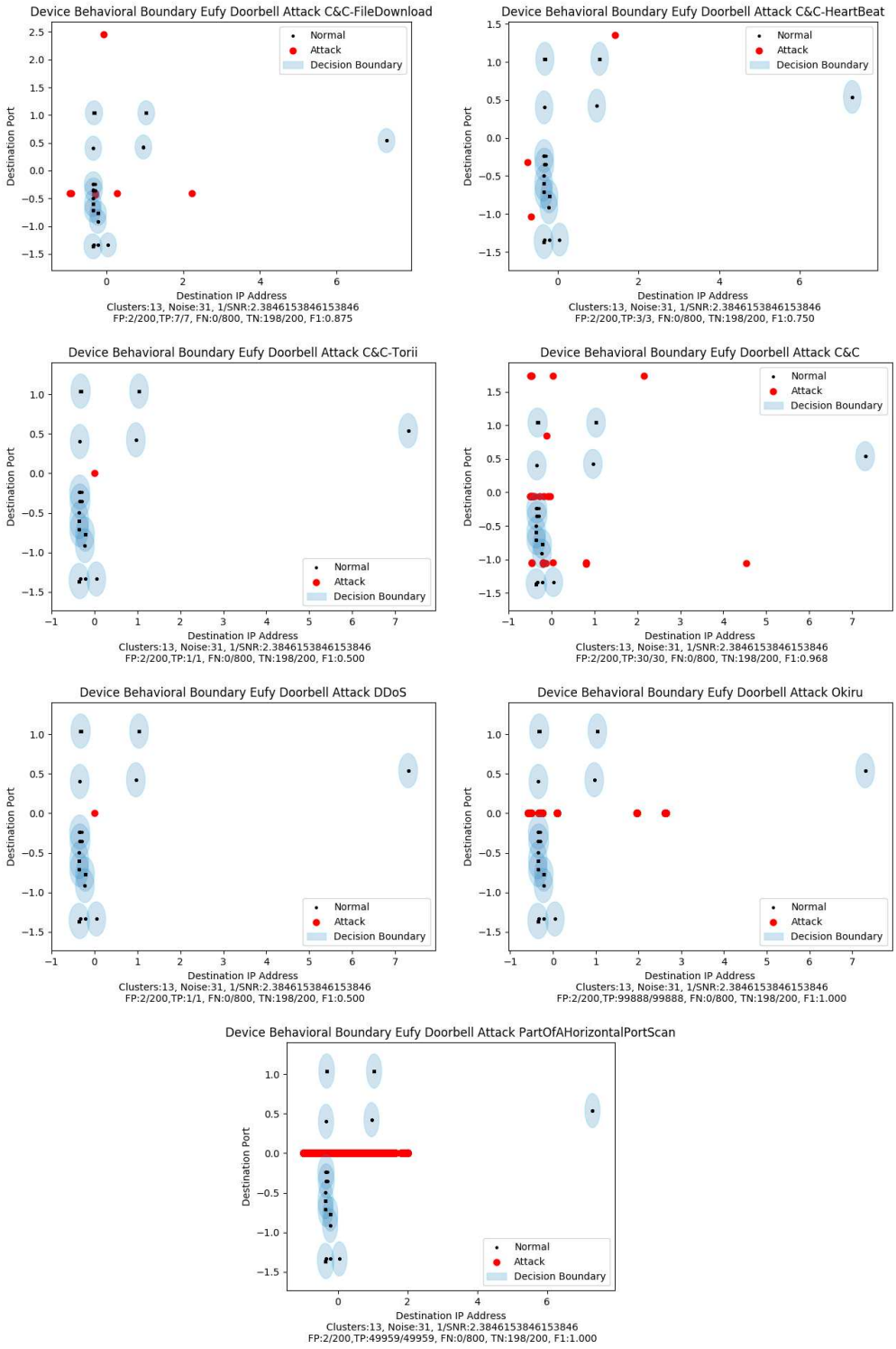


Figure B.10: Eufy-Doorbell: HOME

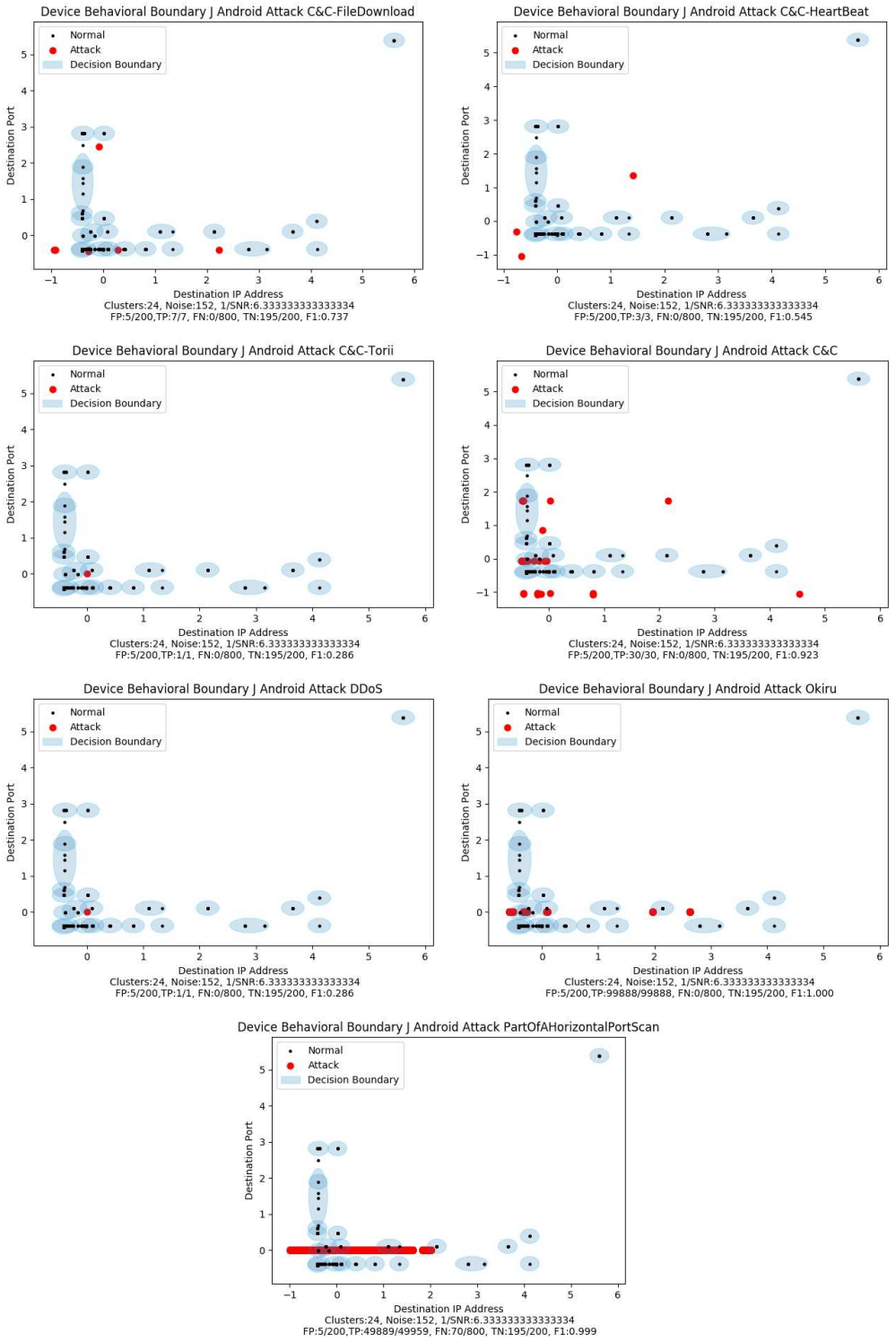


Figure B.11: J-Android: HOME

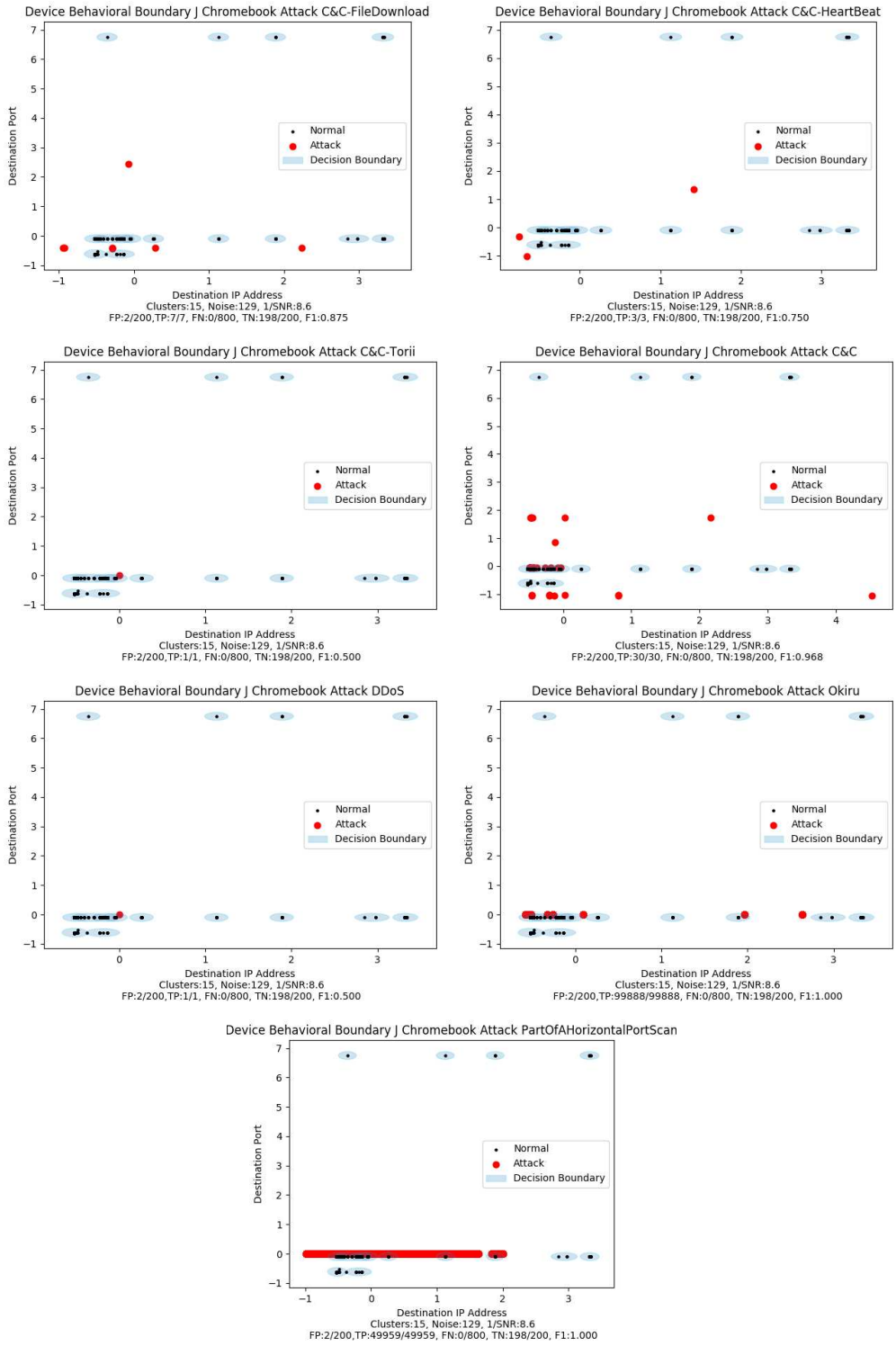


Figure B.12: J-Chromebook: HOME

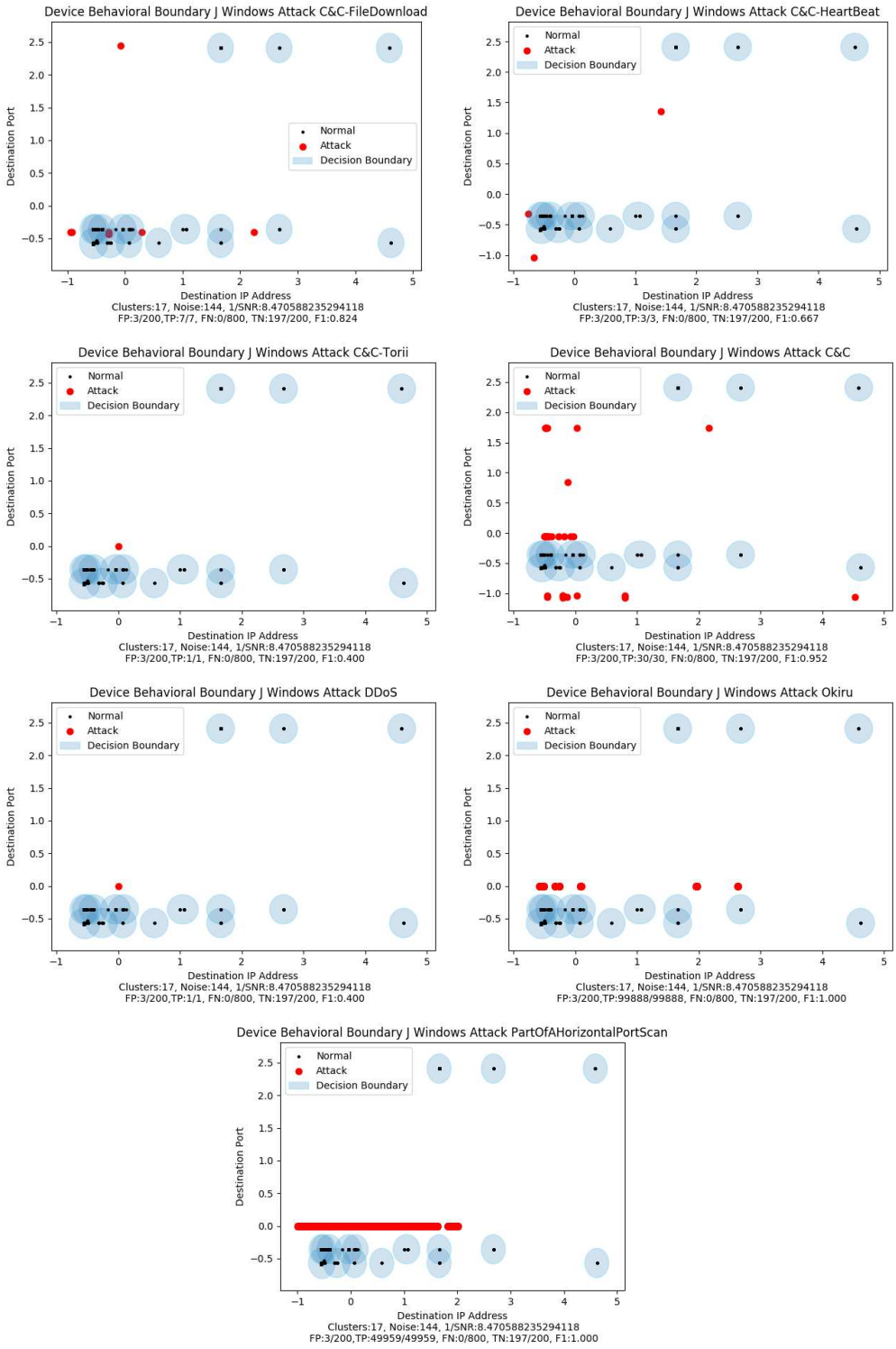


Figure B.13: J-Windows: HOME

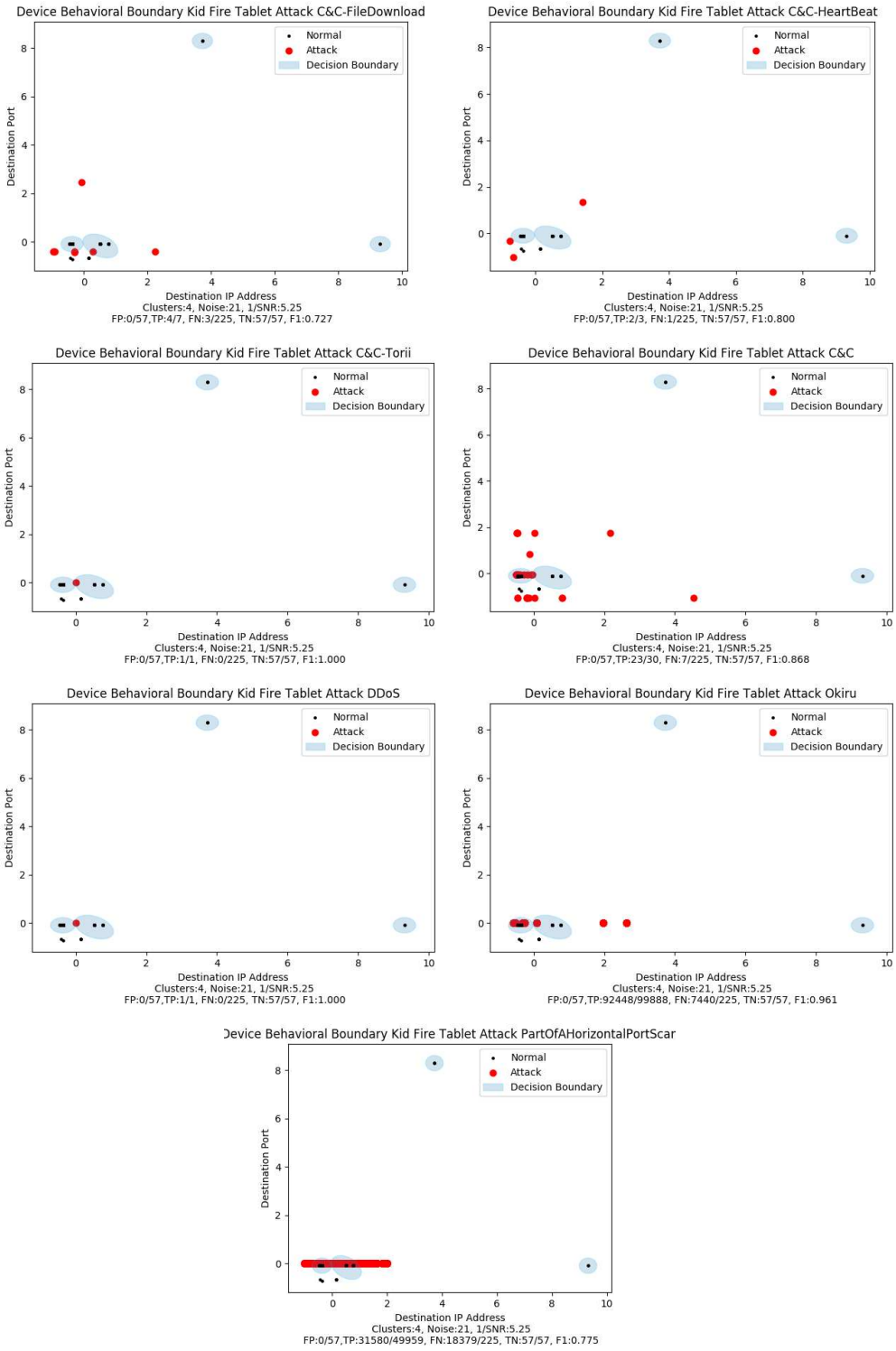


Figure B.14: Kid-Fire-Tablet: HOME

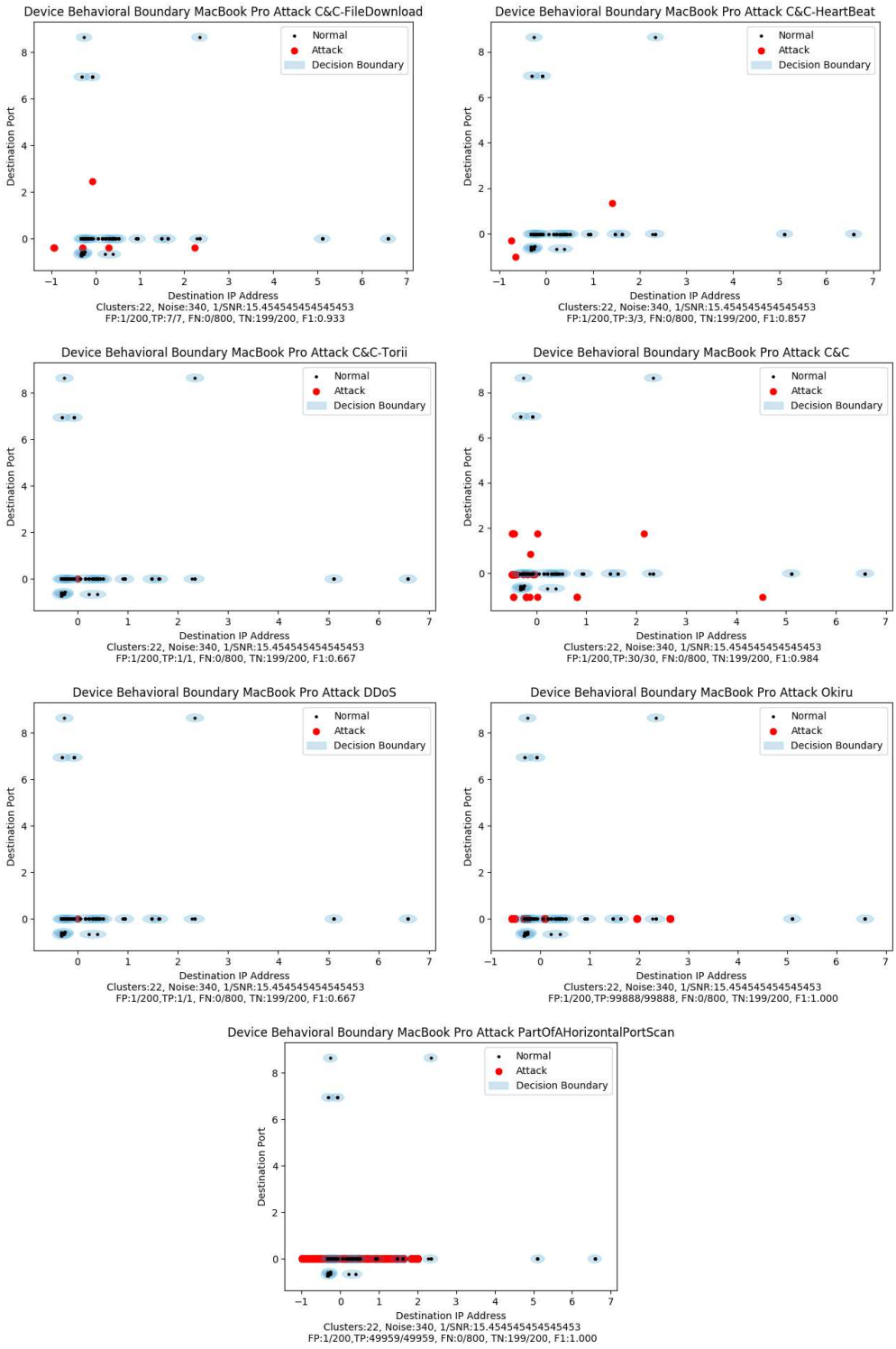


Figure B.15: MacBook-Pro: HOME

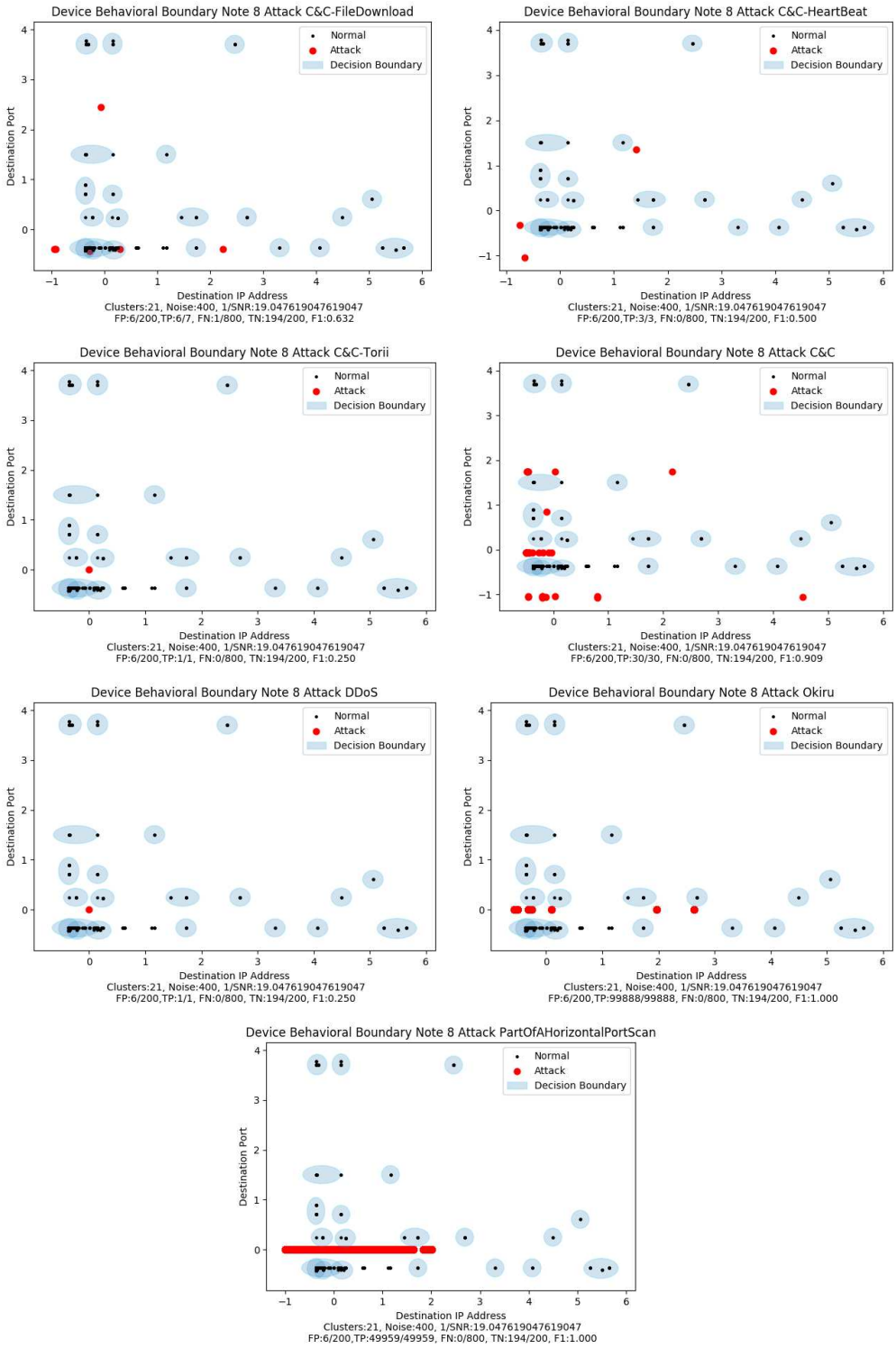


Figure B.16: Note-8: HOME

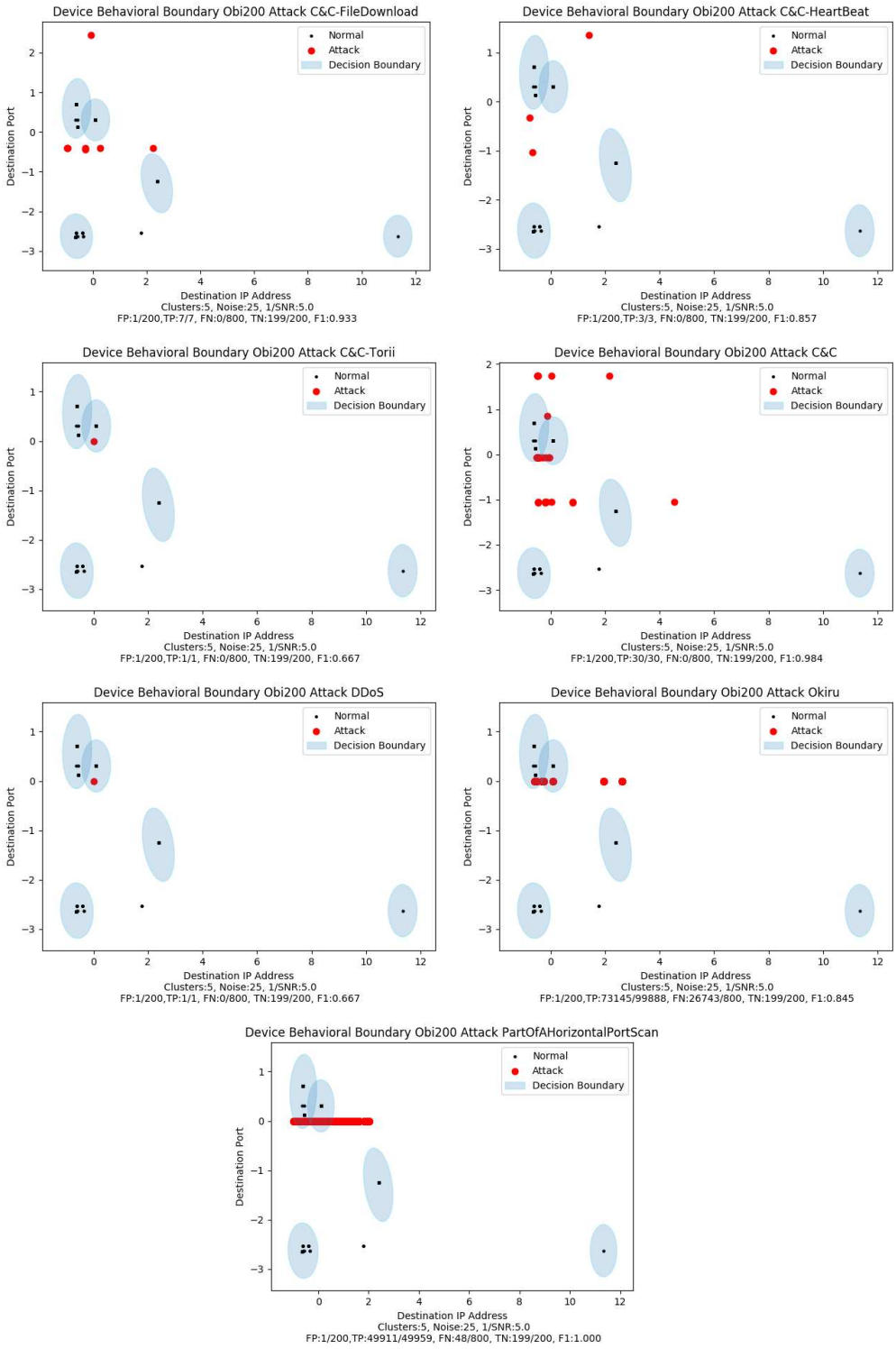


Figure B.17: Obi200: HOME

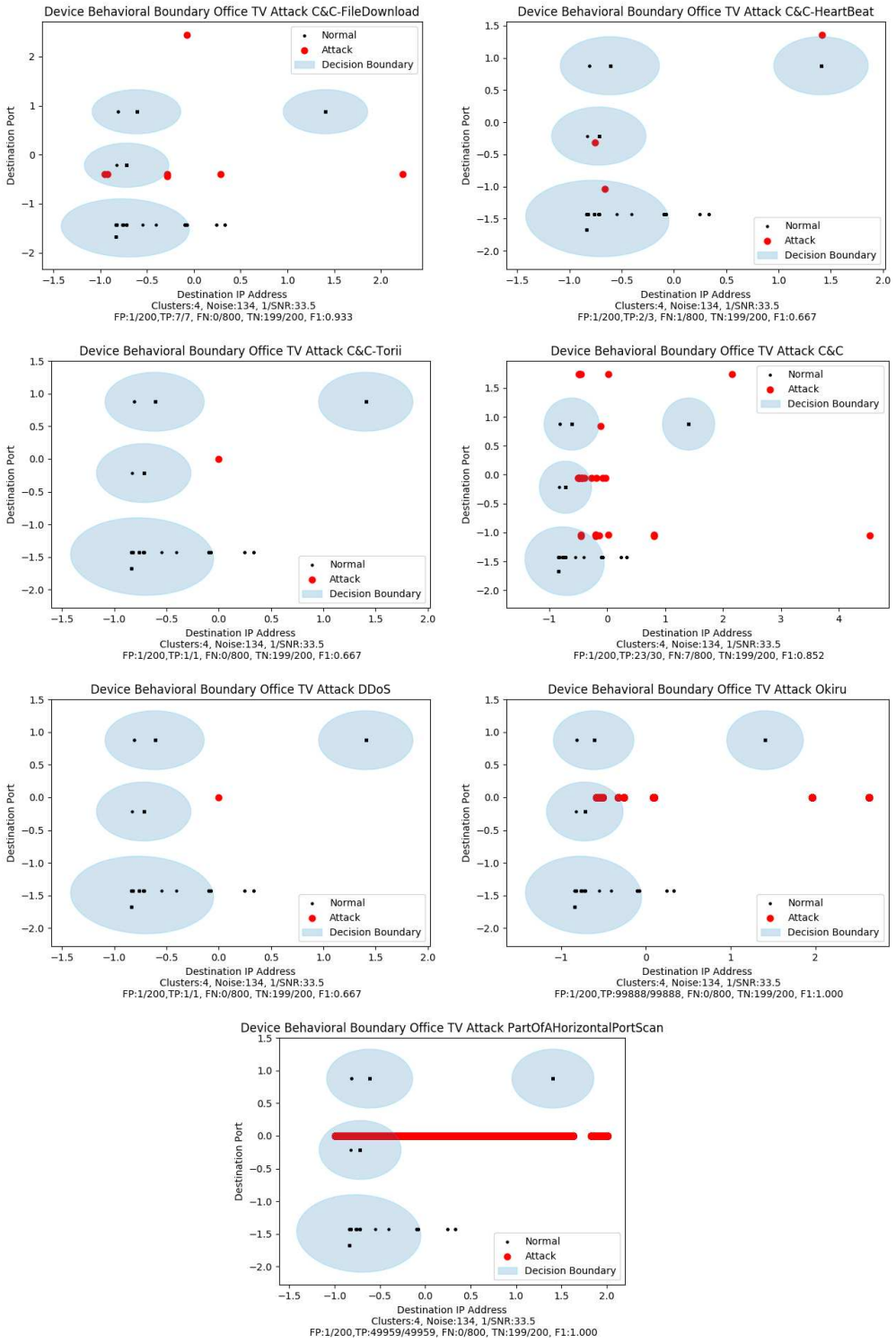


Figure B.18: Office-TV: HOME

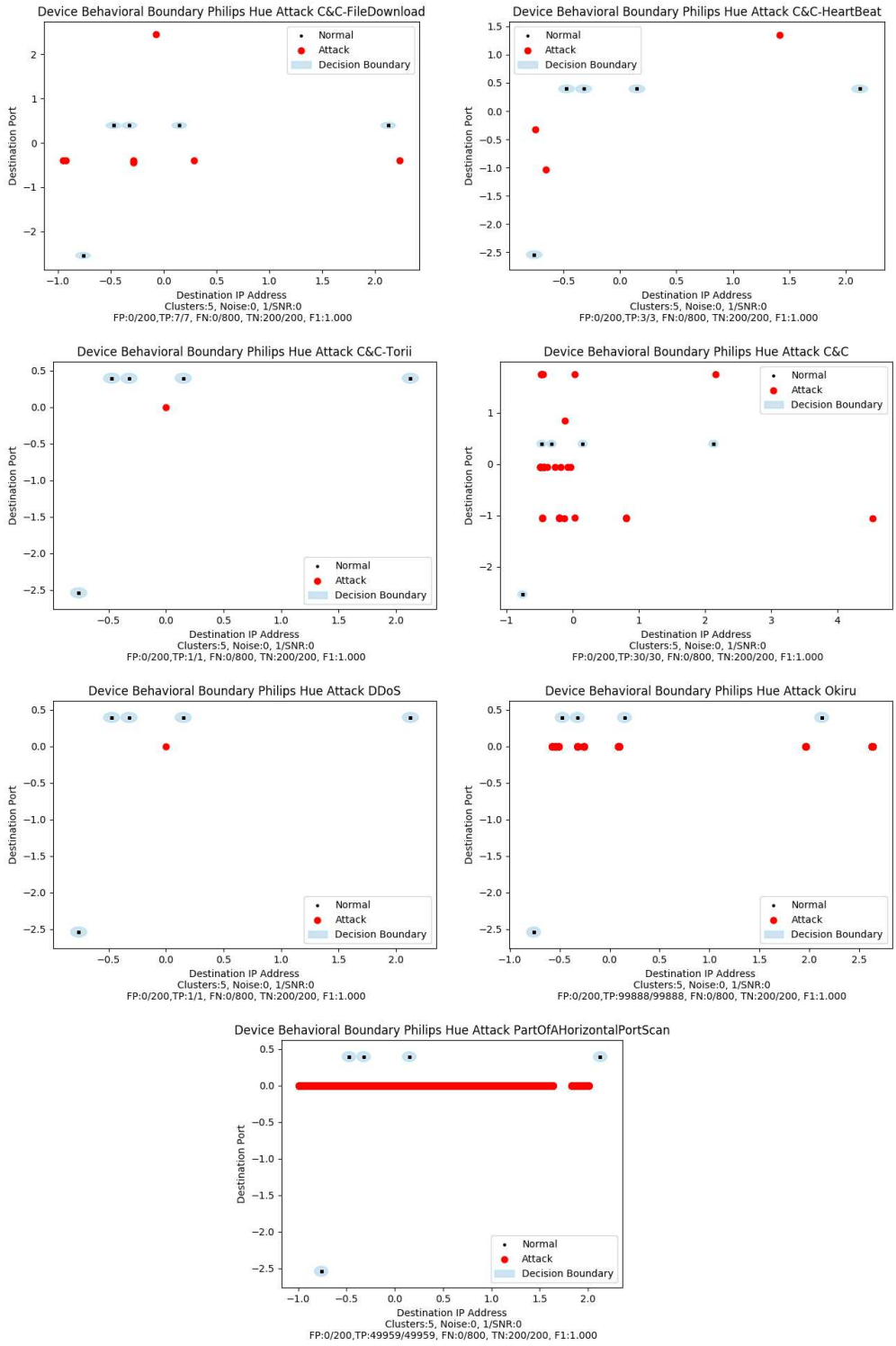


Figure B.19: Philips-Hue: HOME

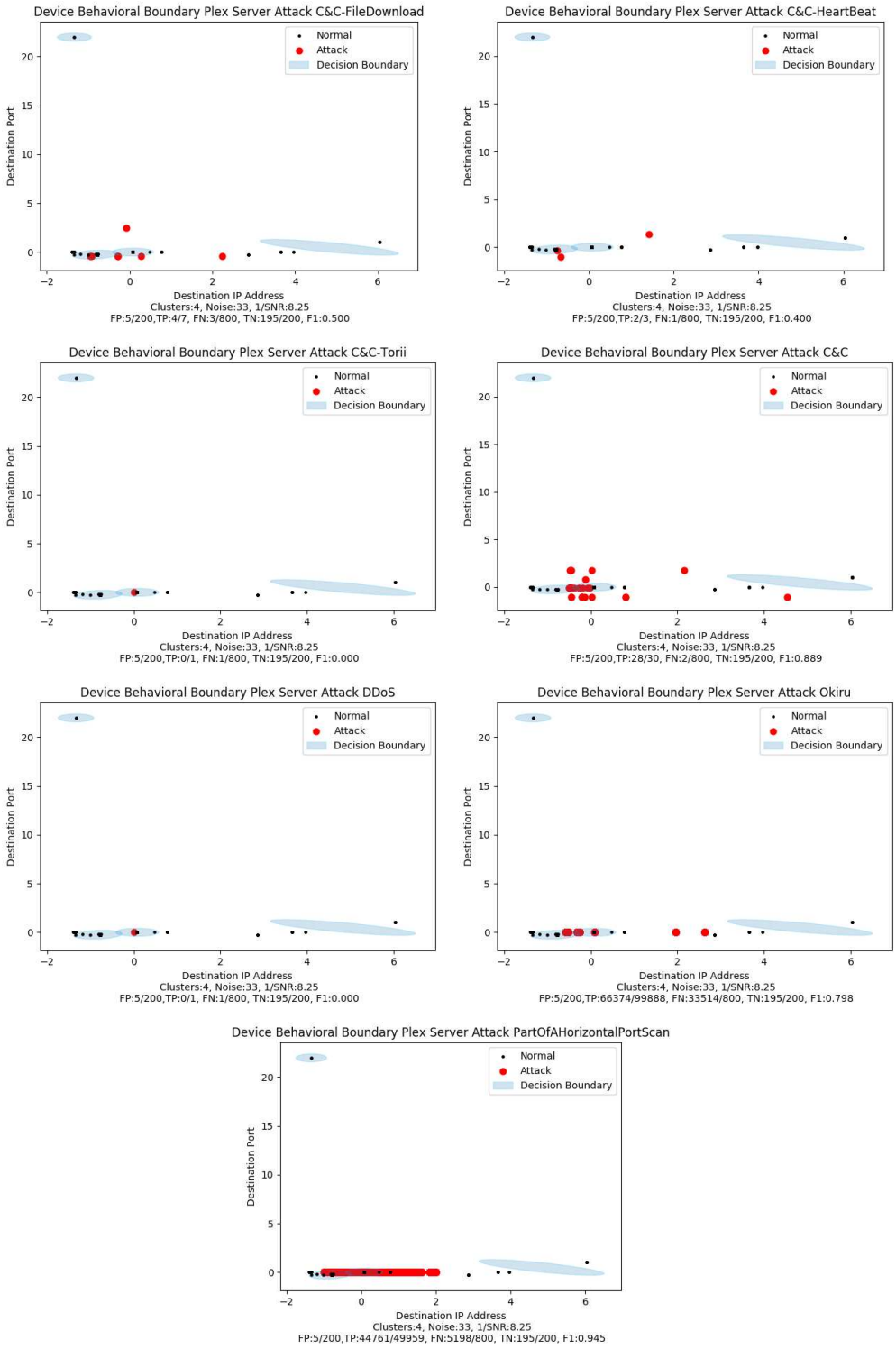


Figure B.20: Plex-Server: HOME

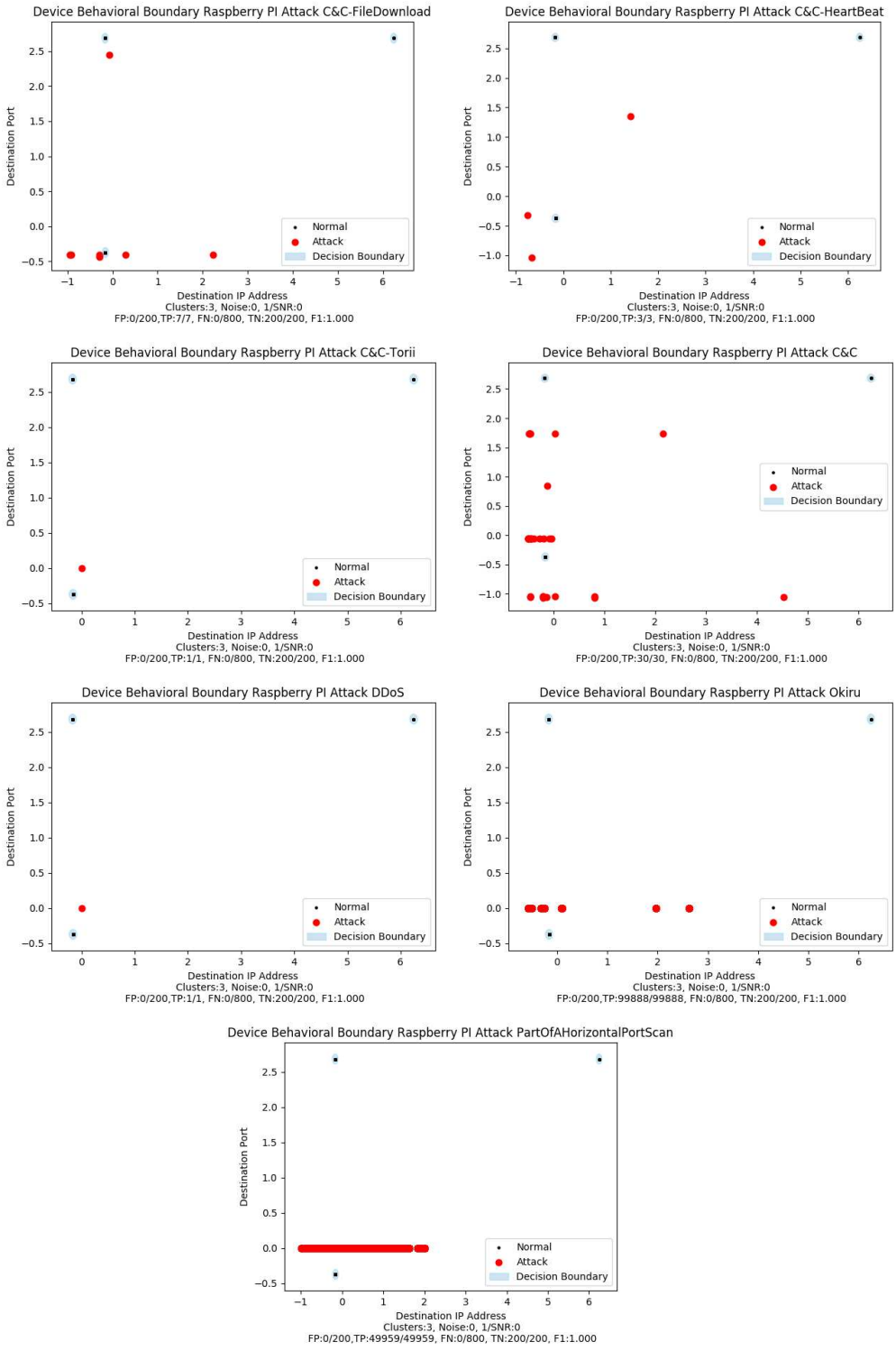


Figure B.21: Raspberry-PI: HOME

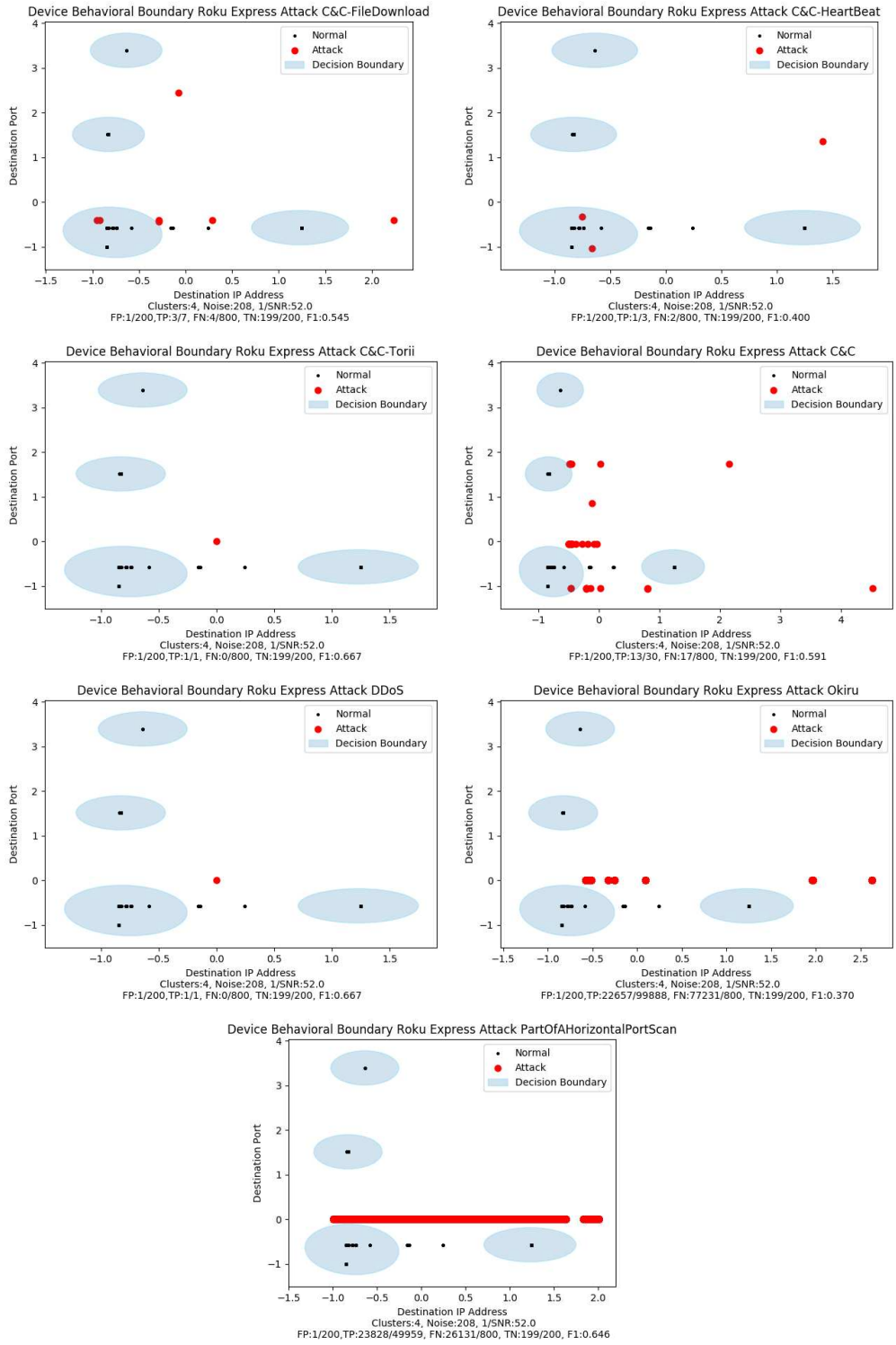


Figure B.22: Roku-Express: HOME

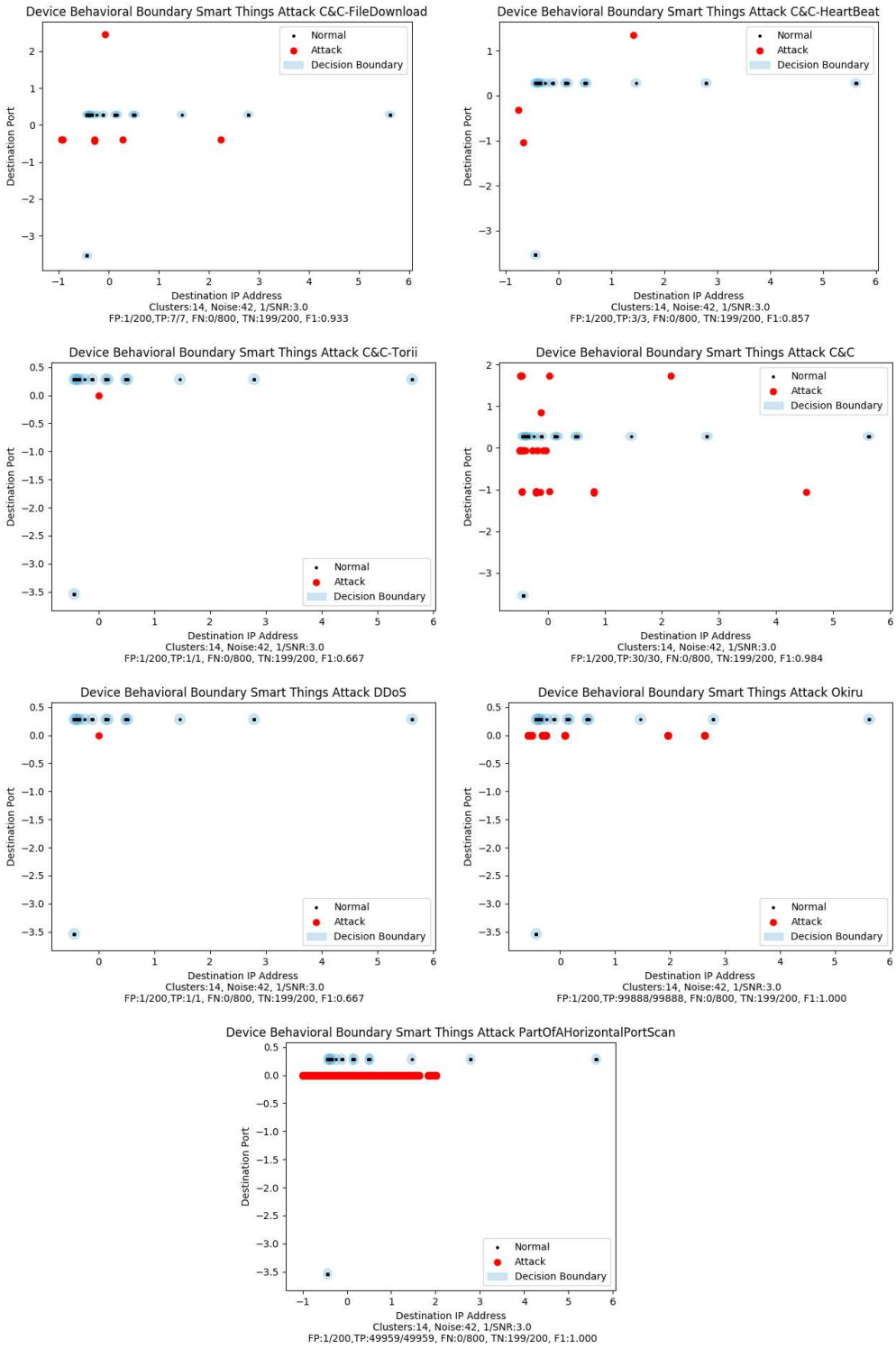


Figure B.23: Smart-Things: HOME

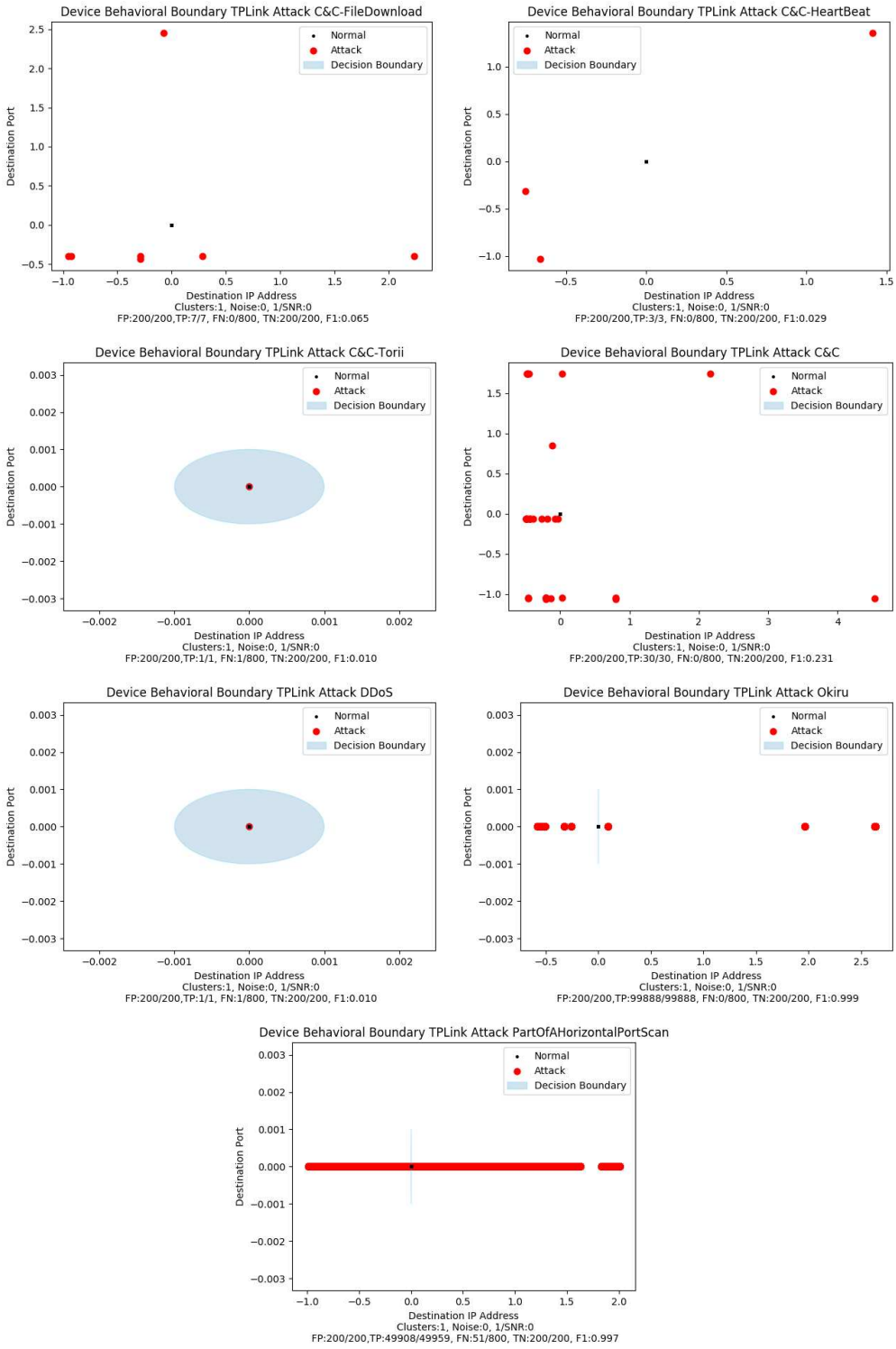


Figure B.24: TPLink: HOME

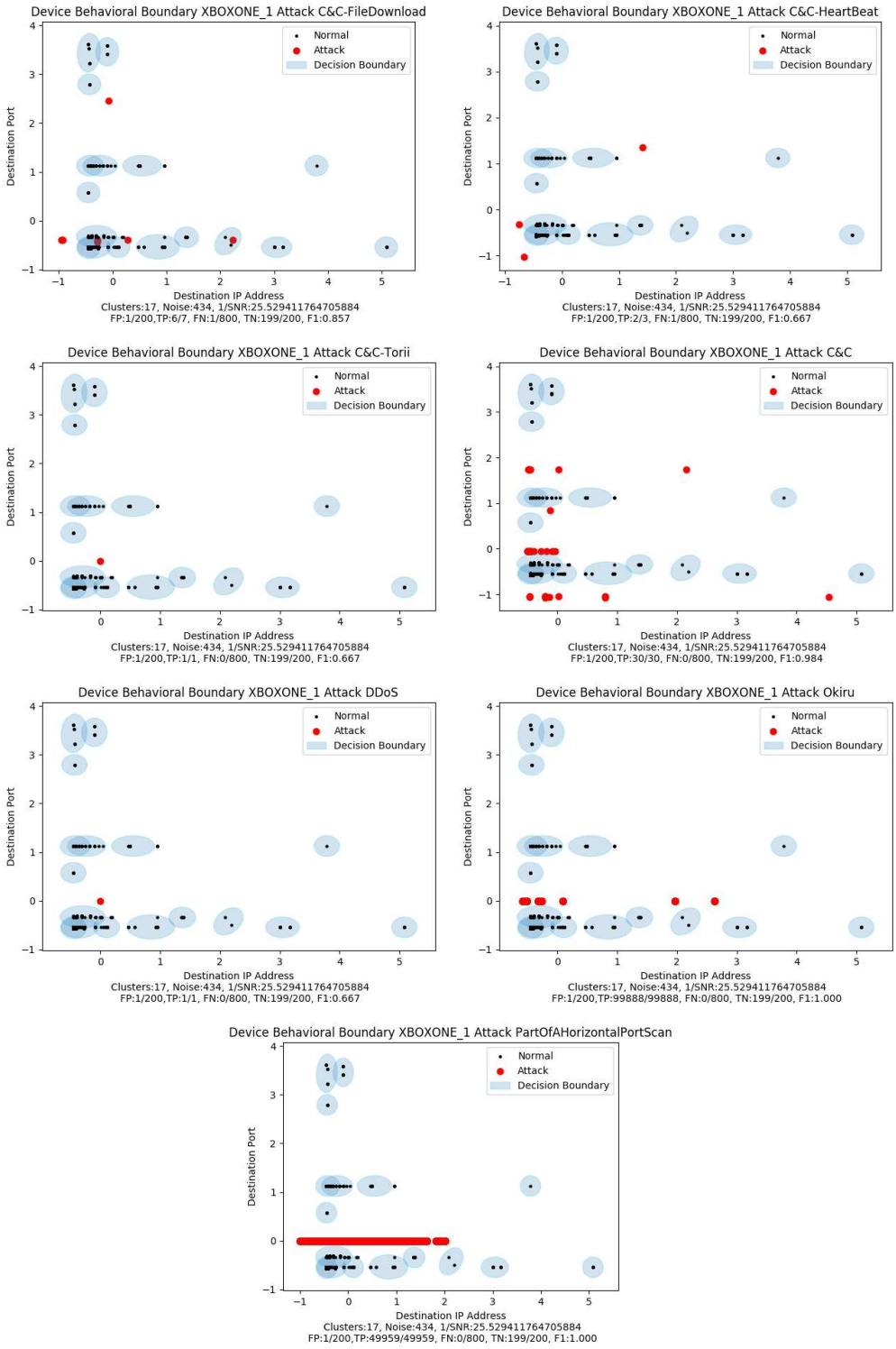


Figure B.25: XBOXONE-1: HOME

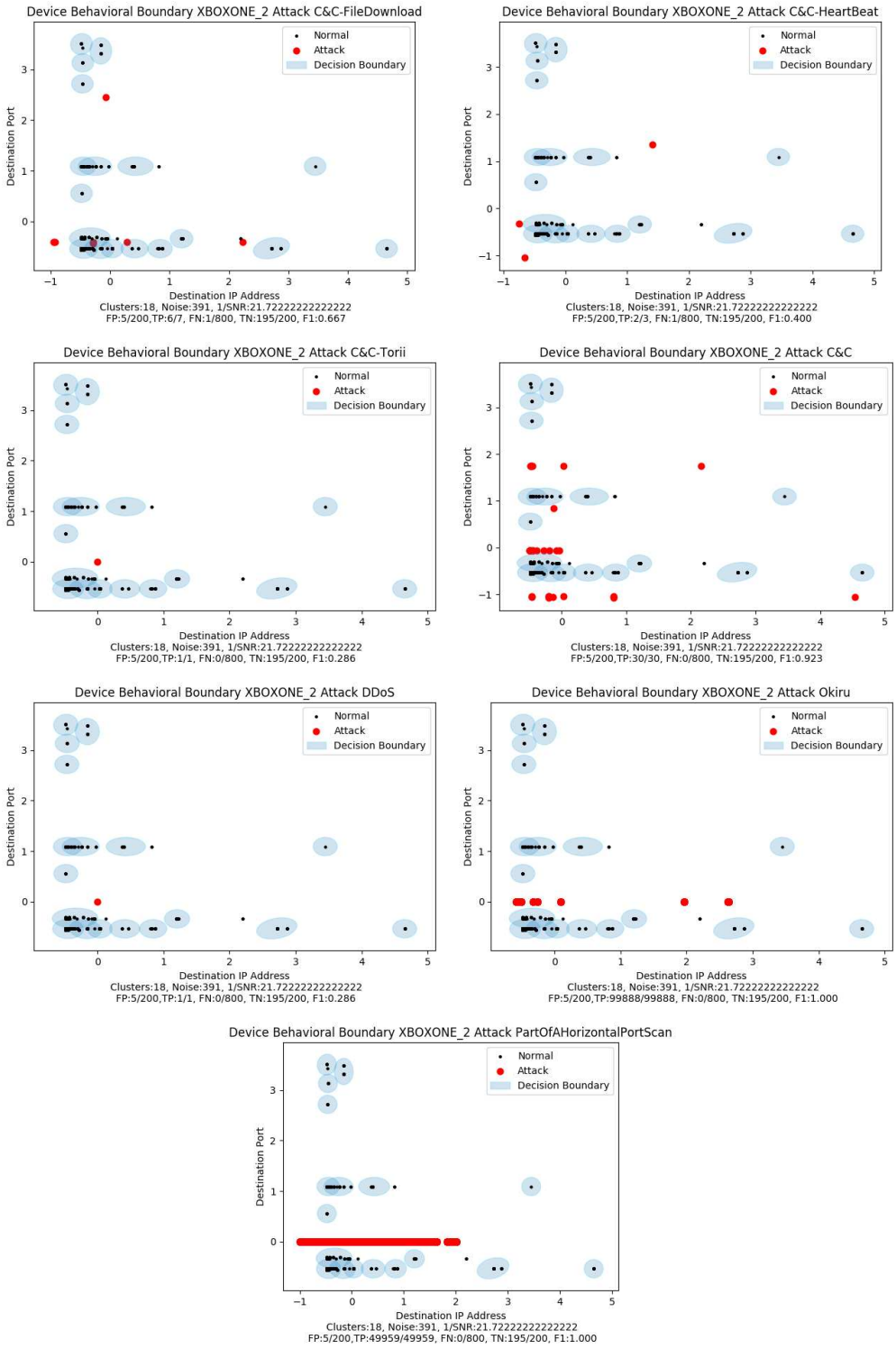


Figure B.26: XBOXONE-2: HOME

Appendix C

Lab Appendix

This Appendix shows the Gaussian Boundary for each device in the lab dataset against all seven attacks listed in 3.6.

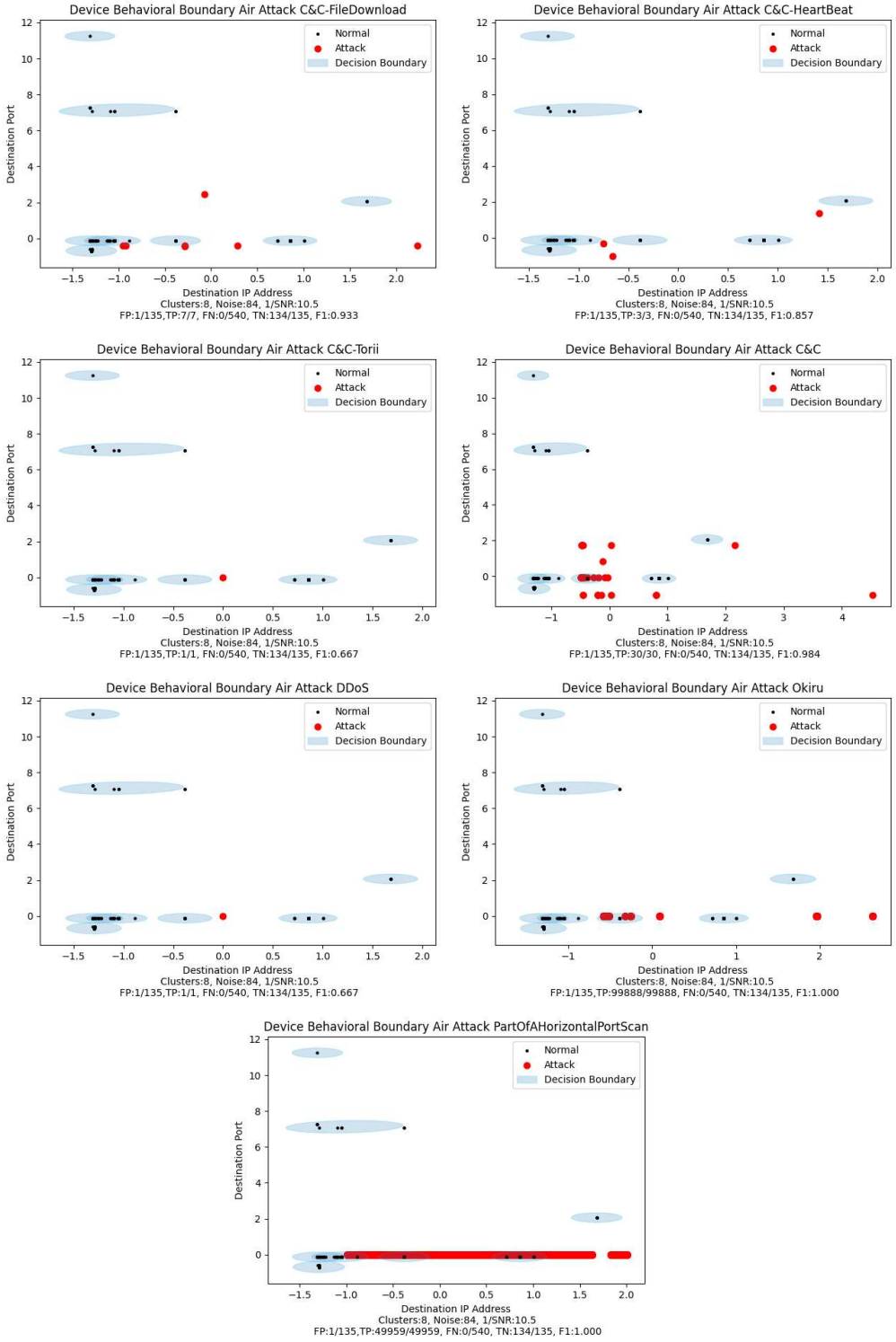


Figure C.1: Air: LAB

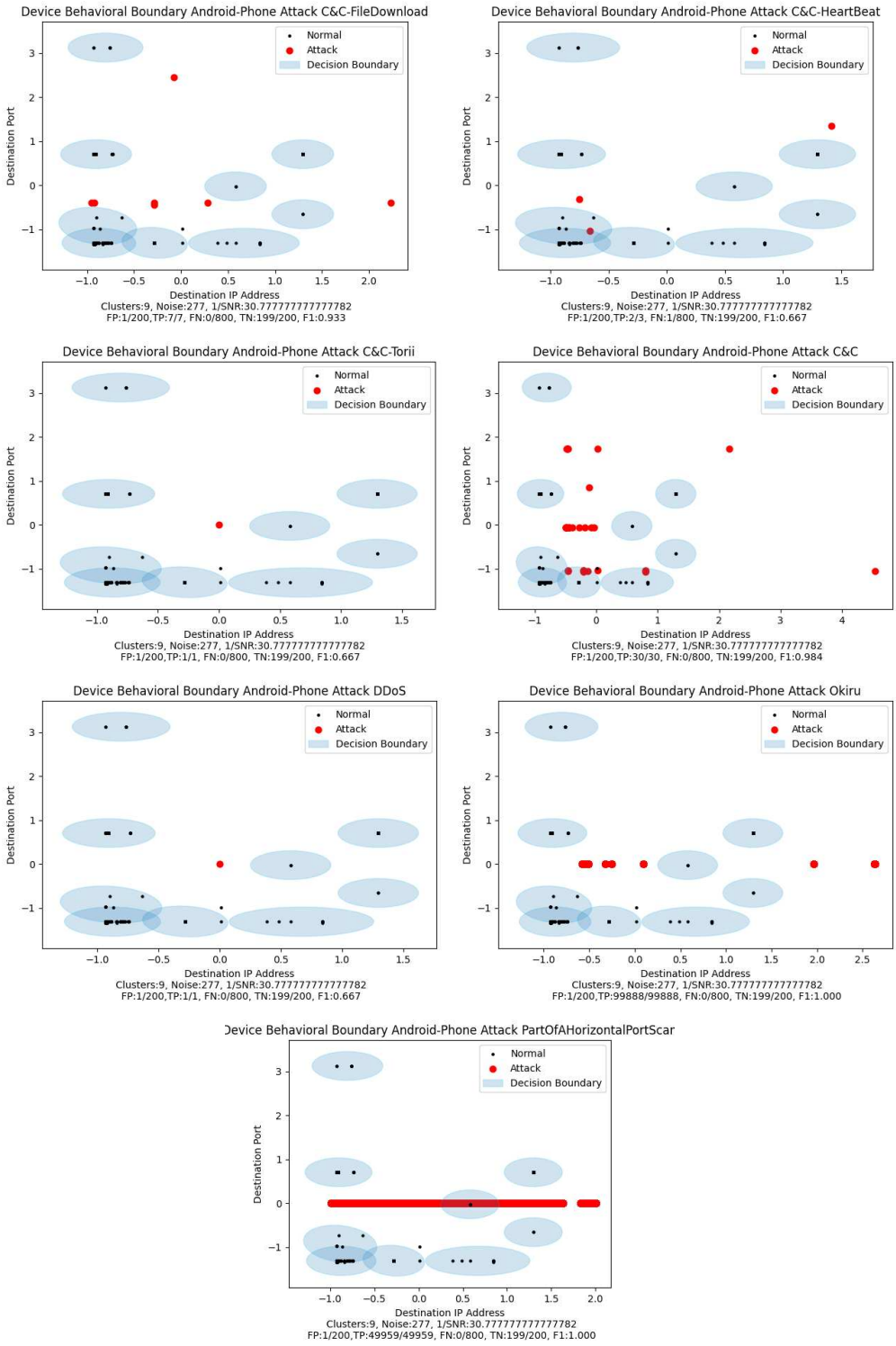


Figure C.2: Android-Phone: LAB

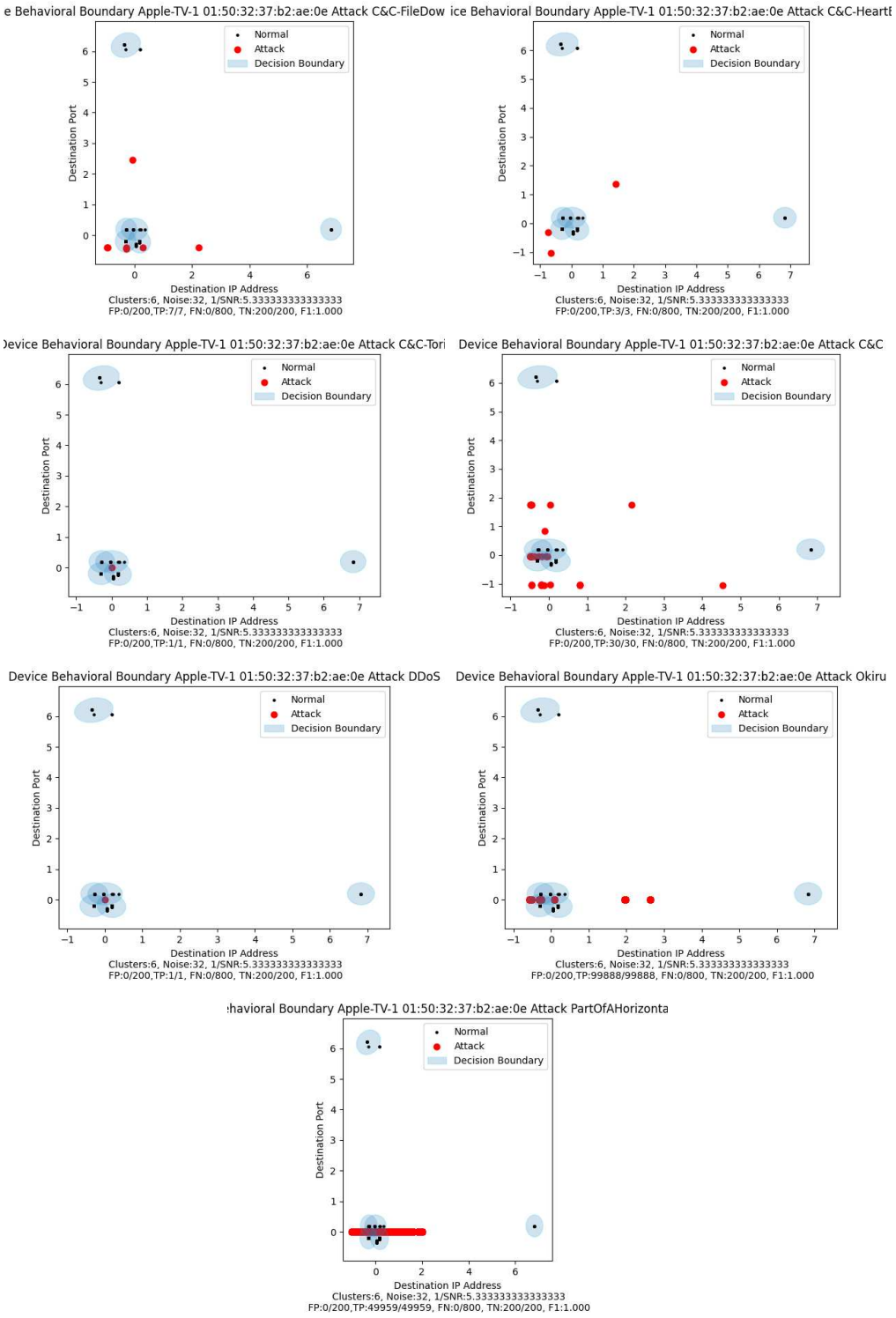


Figure C.3: Apple-TV-1: LAB

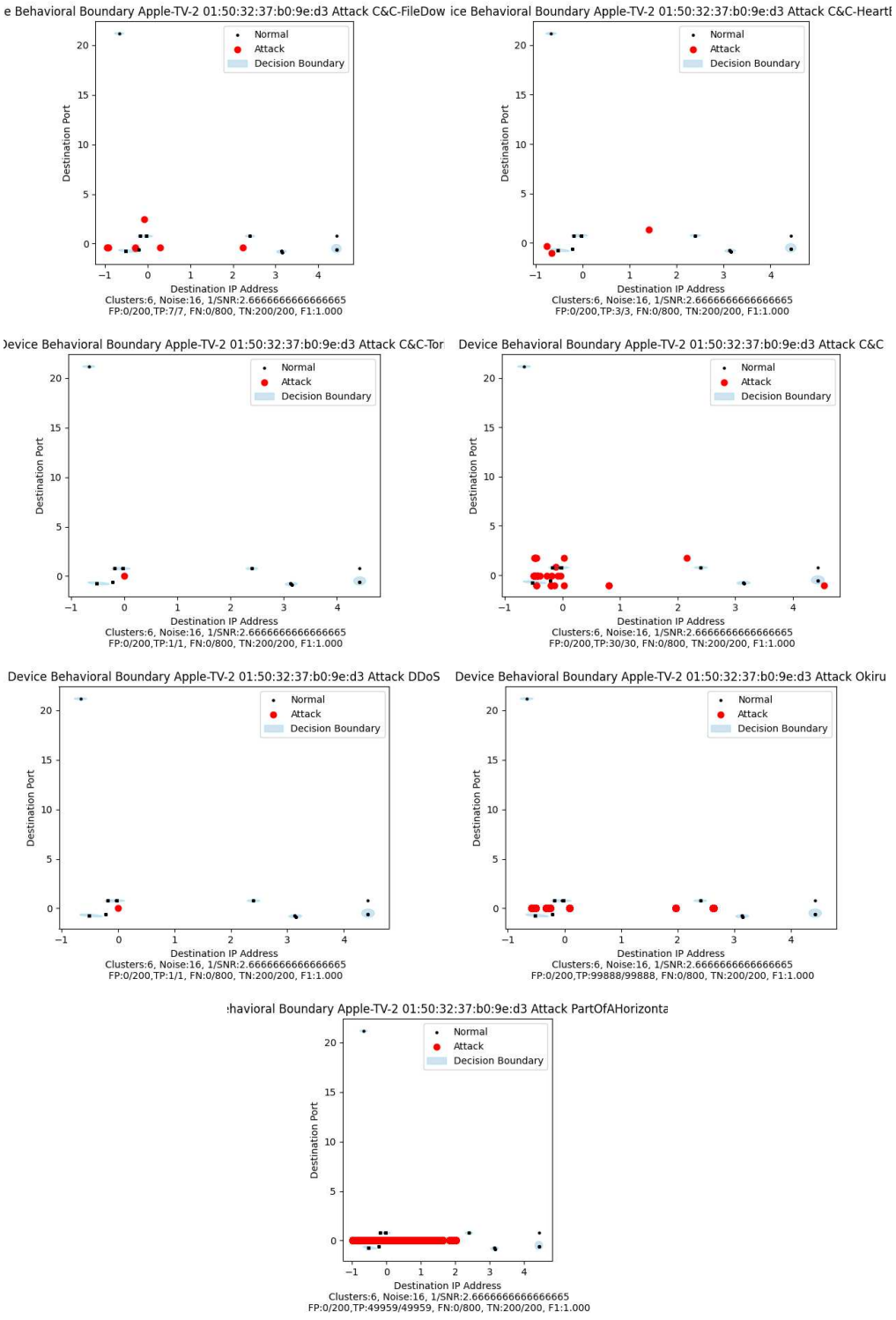


Figure C.4: Apple-TV-2: LAB

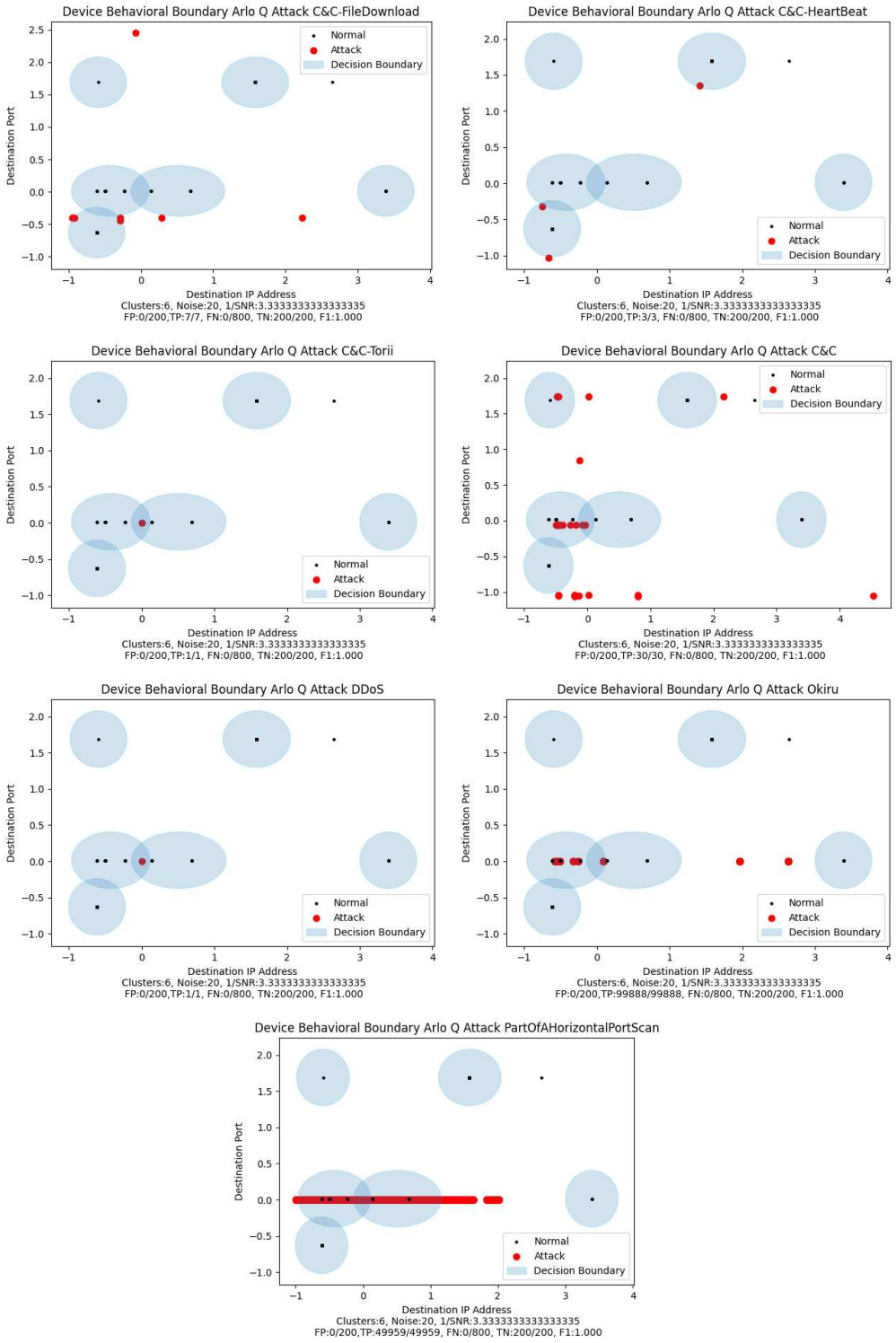


Figure C.5: Arlo-Q: LAB

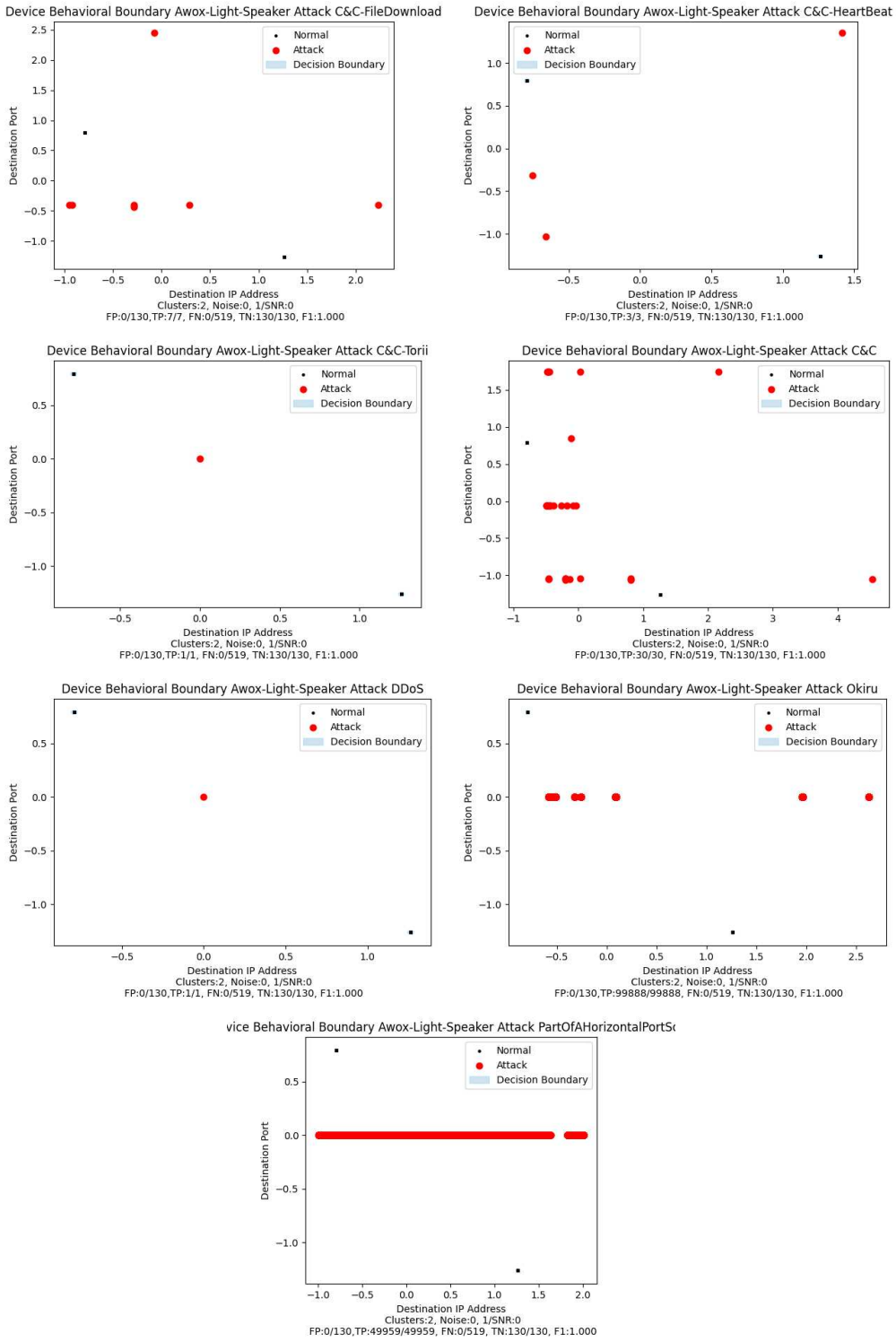


Figure C.6: Awox-Light-Speaker: LAB

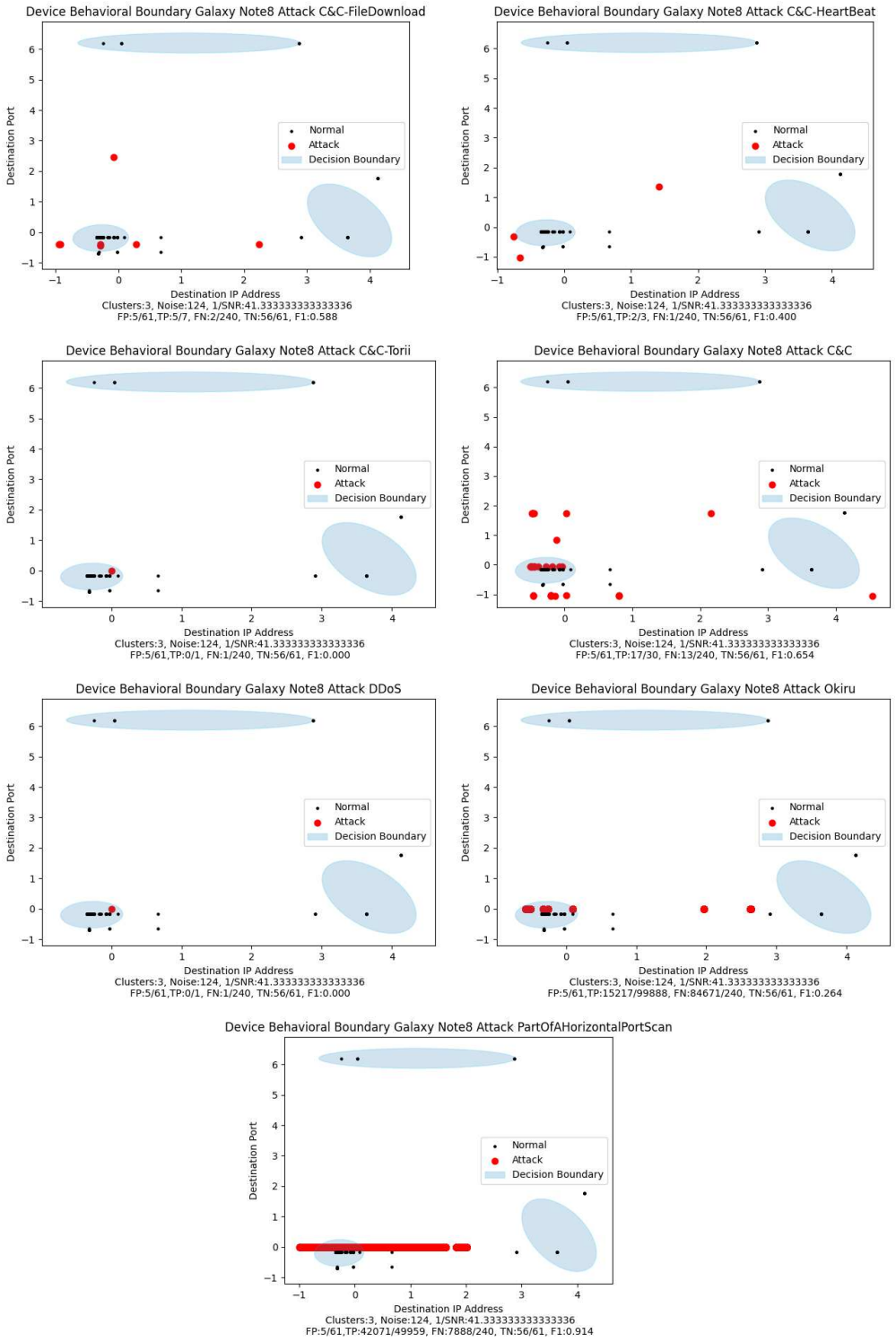


Figure C.7: Galaxy-Note8: LAB

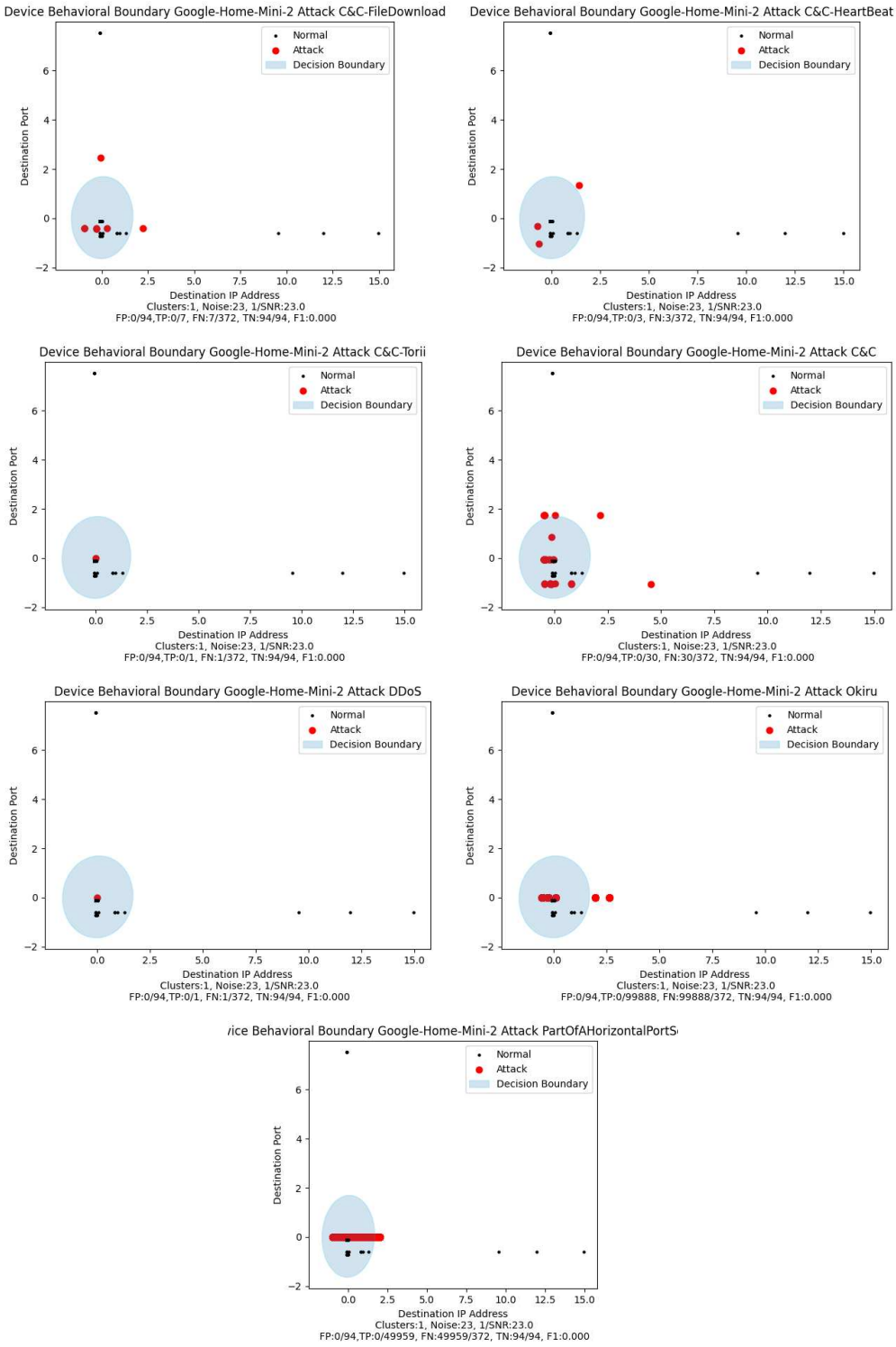


Figure C.8: Google-Home-Mini-2: LAB

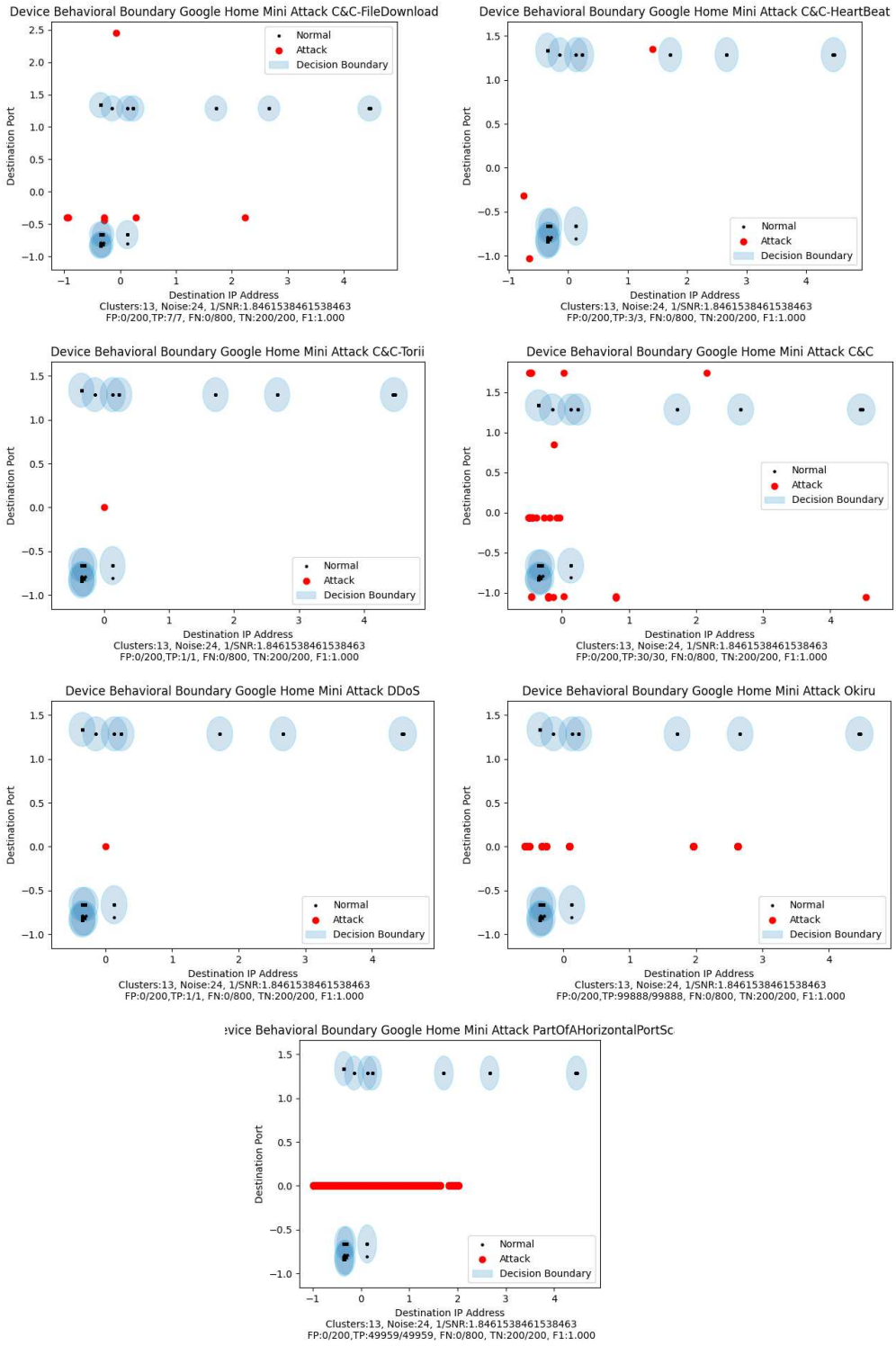


Figure C.9: Google-Home-Mini: LAB

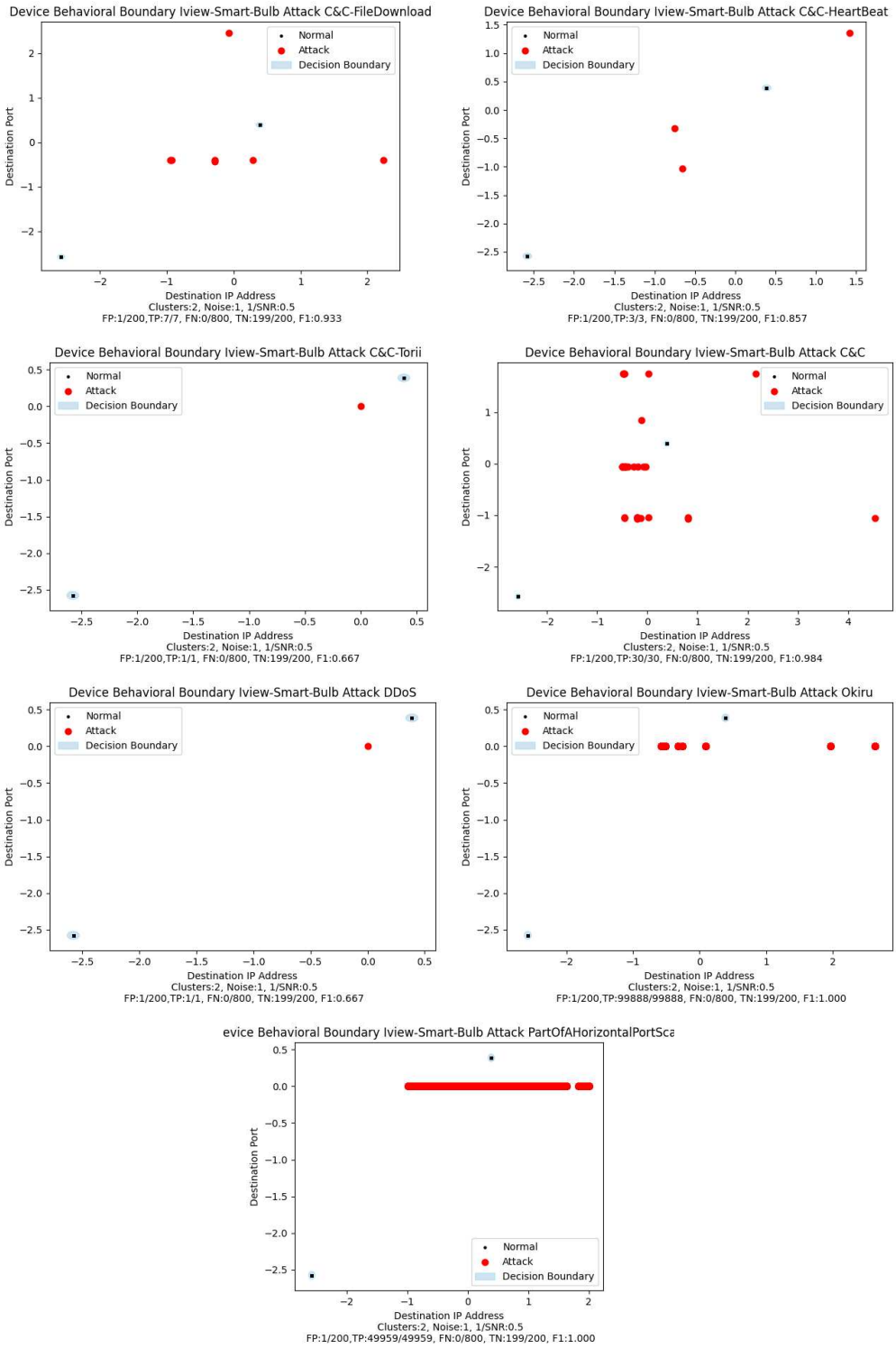


Figure C.10: Iview-Smart-Bulb: LAB

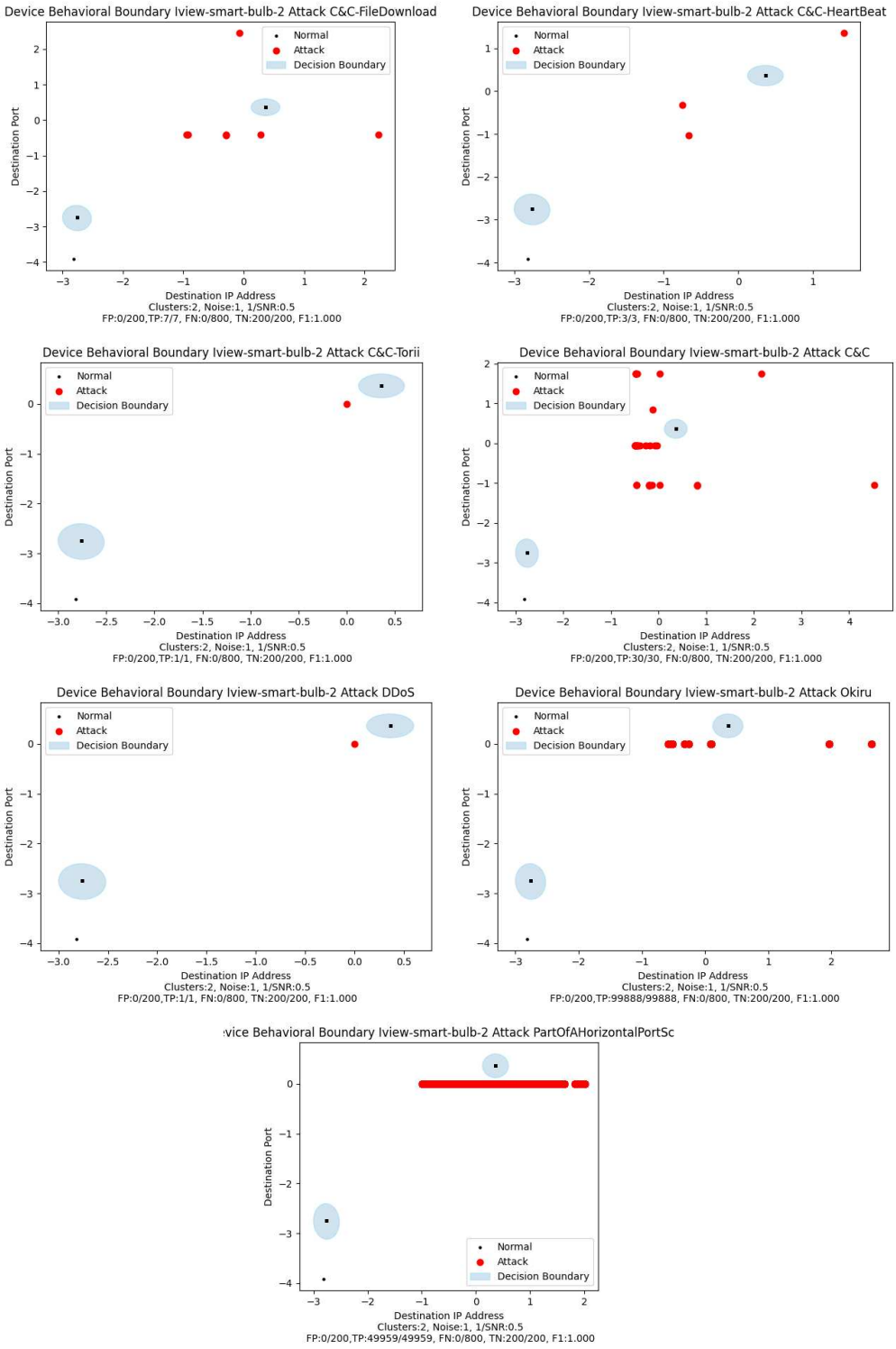


Figure C.11: Iview-smart-bulb-2: LAB

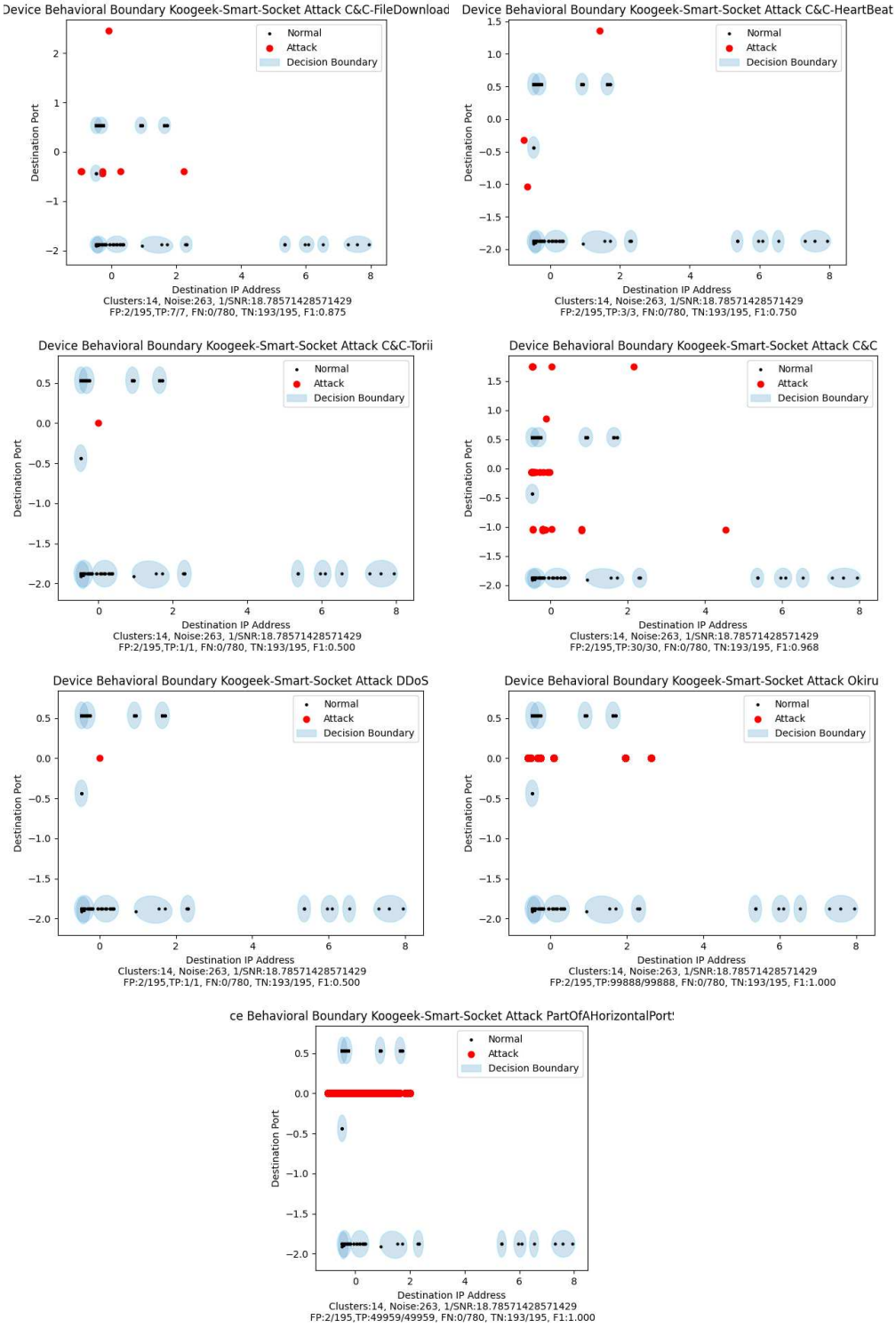


Figure C.12: Koogeek-Smart-Socket: LAB

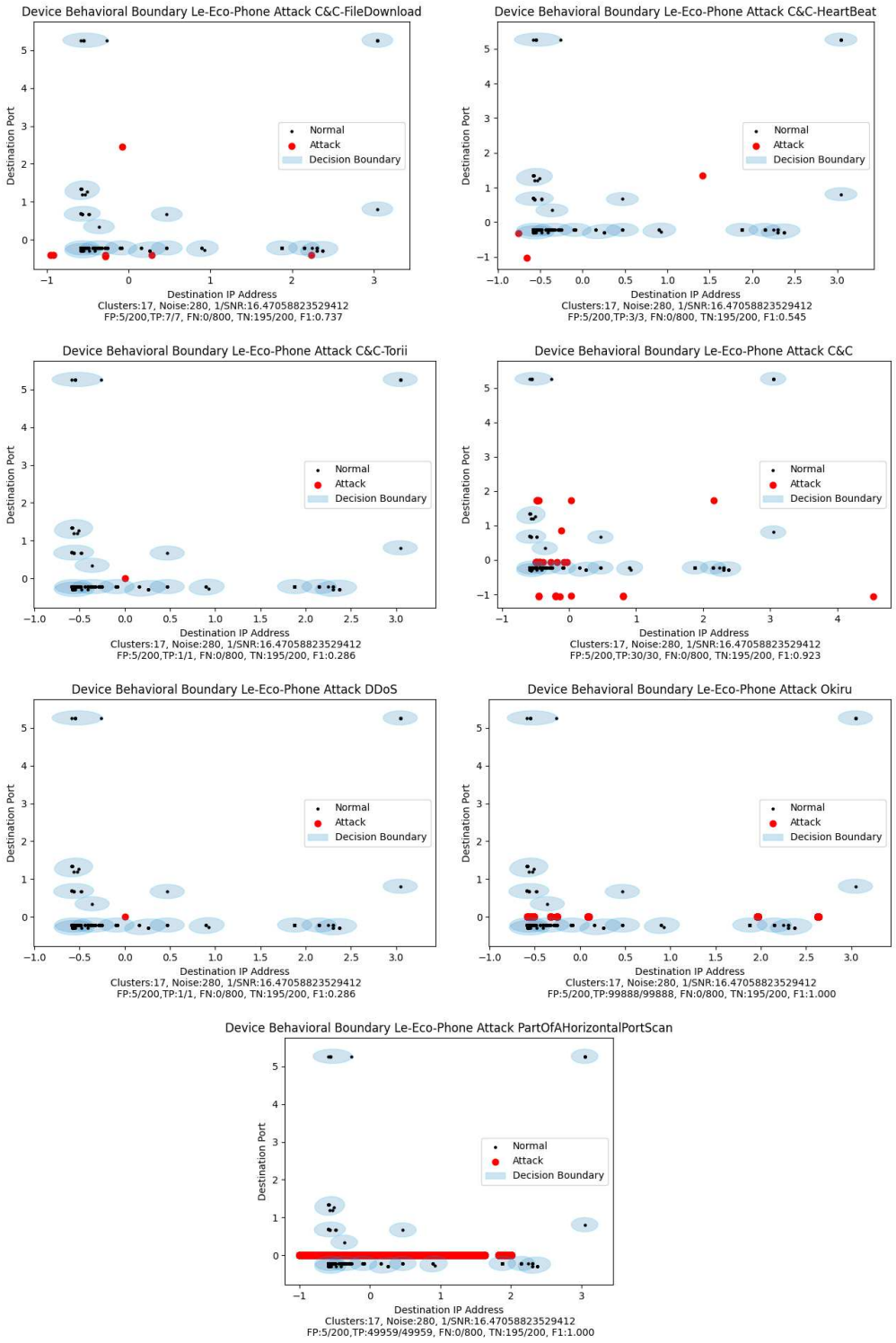


Figure C.13: Le-Eco-Phone: LAB

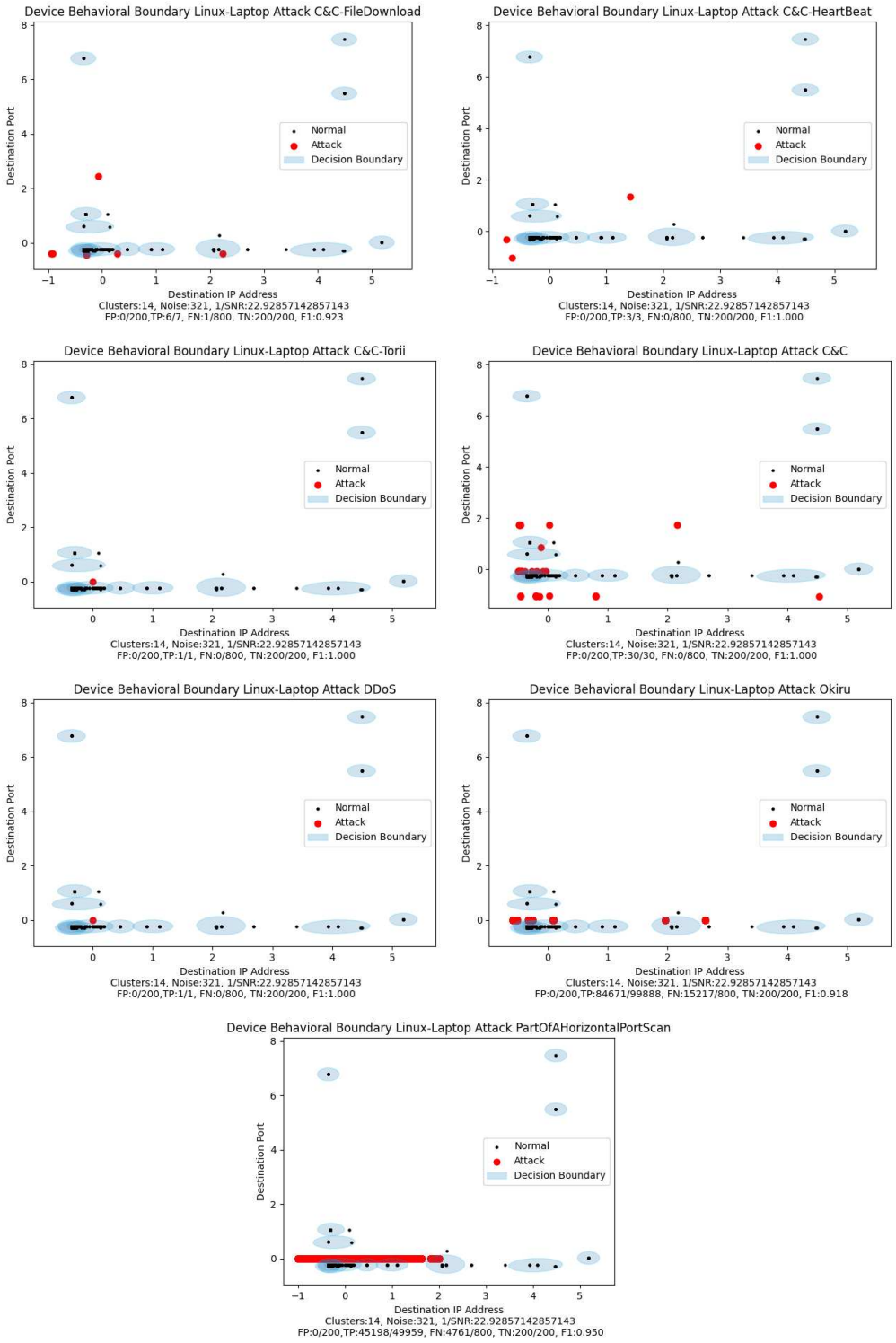


Figure C.14: Linux-Laptop: LAB

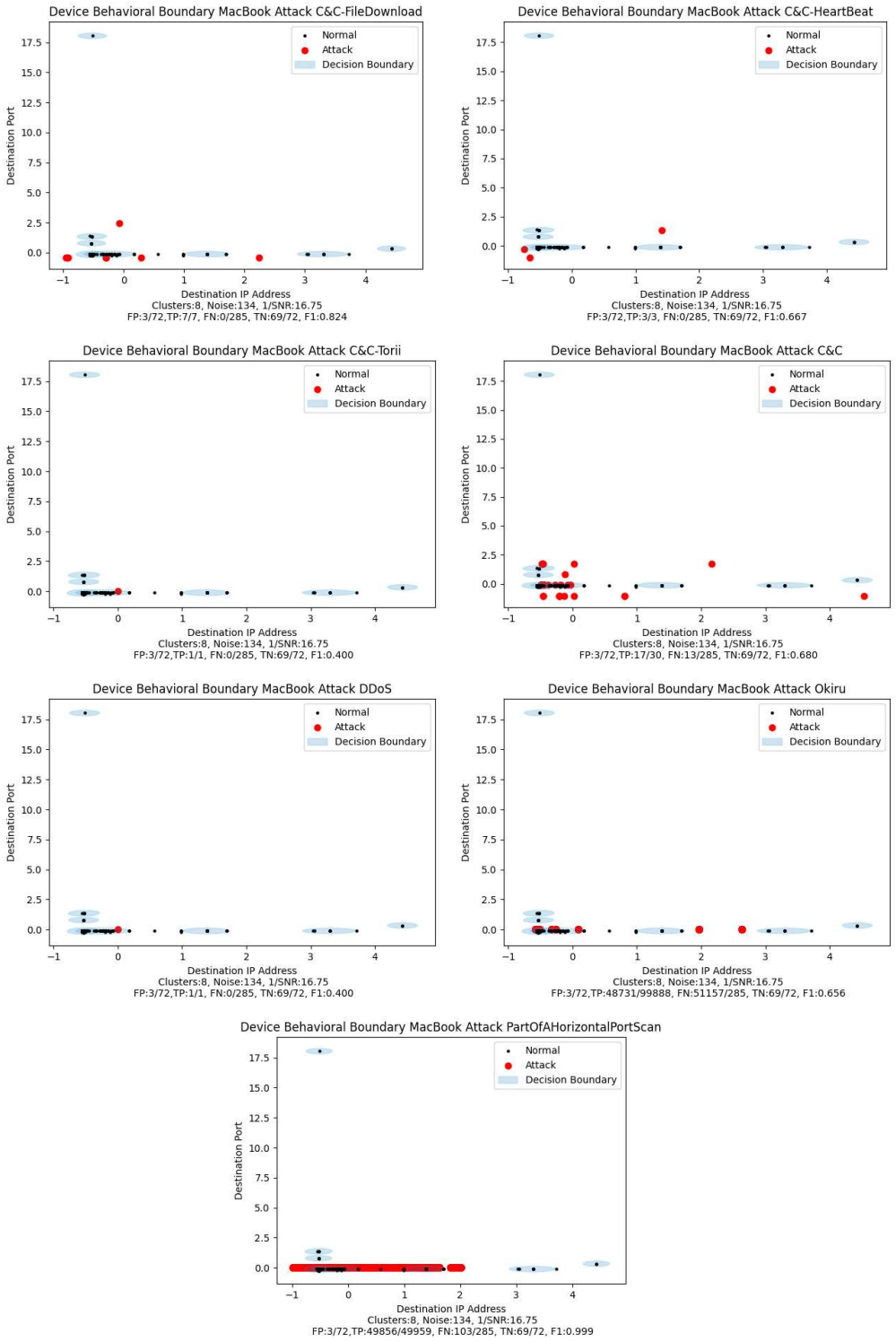


Figure C.15: MacBook: LAB

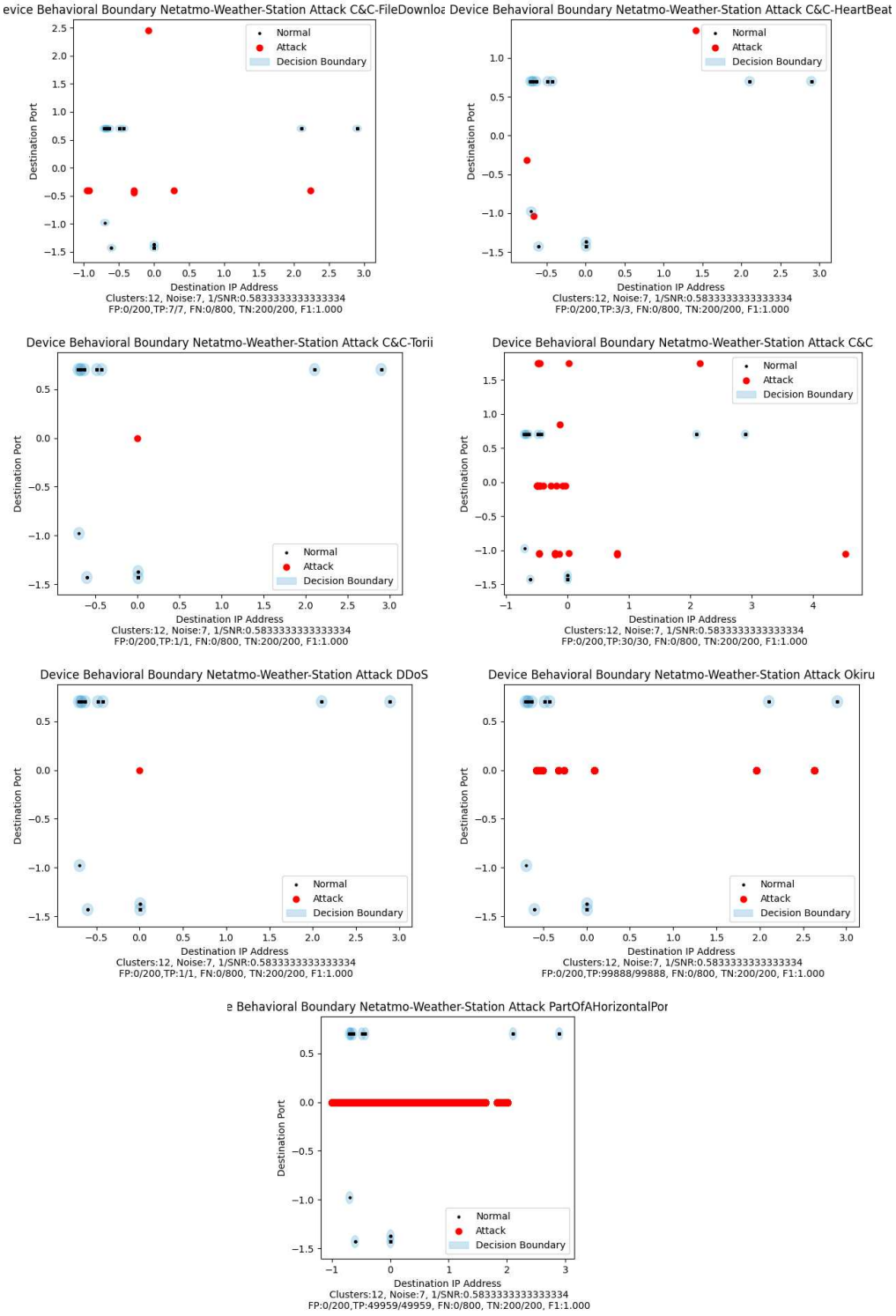


Figure C.16: Netatmo-Weather-Station: LAB

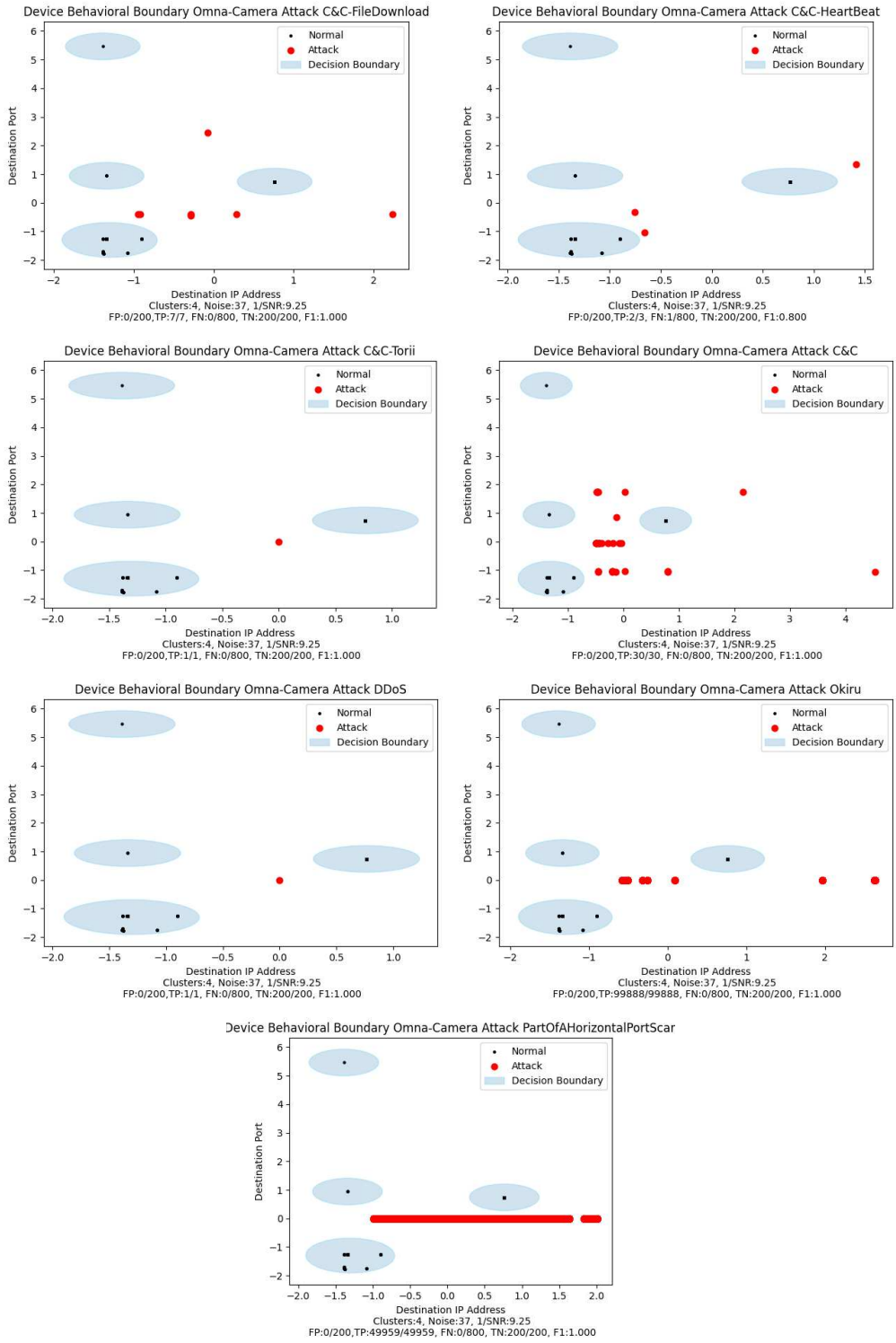


Figure C.17: Omna-Camera: LAB

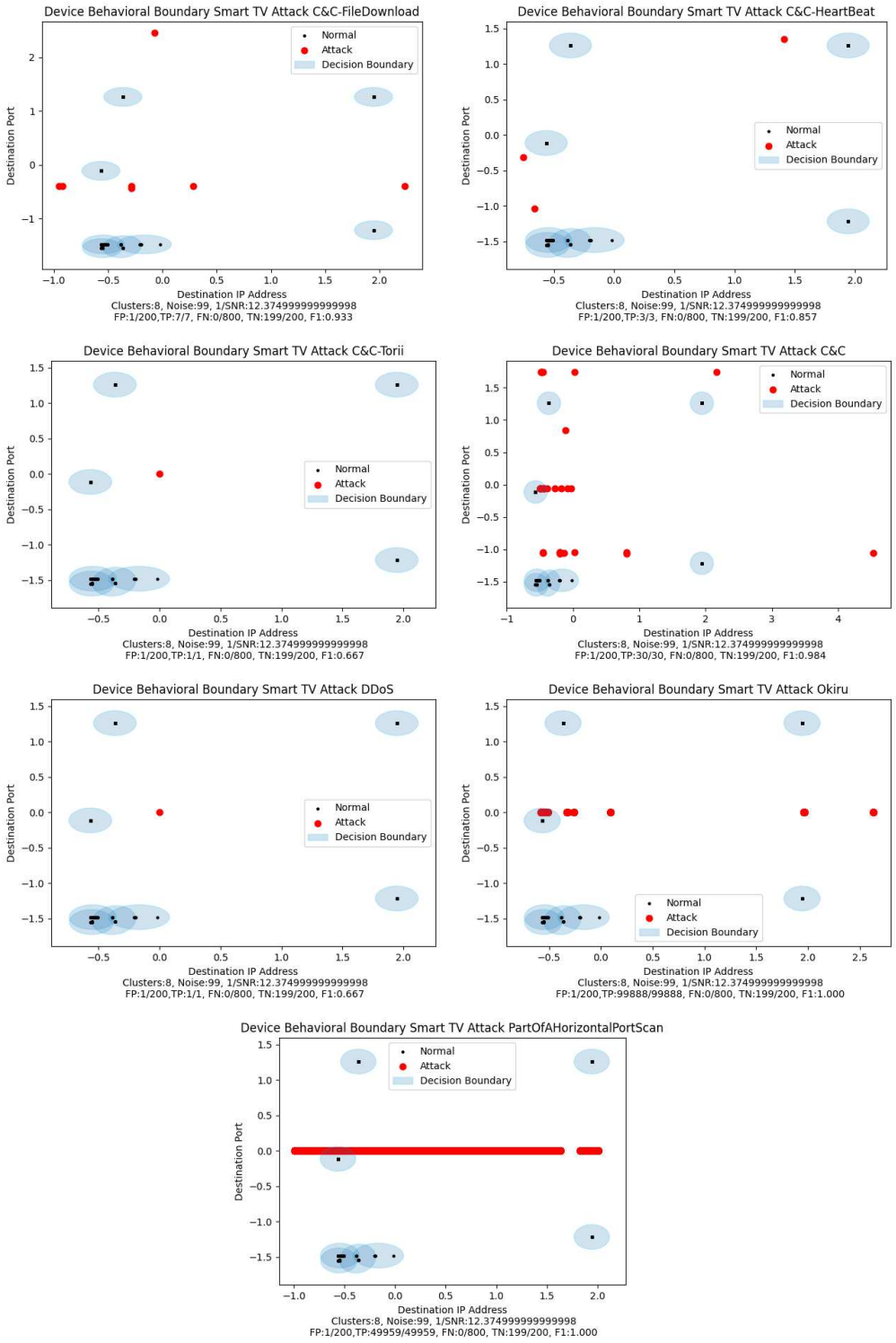


Figure C.18: Smart-TV: LAB

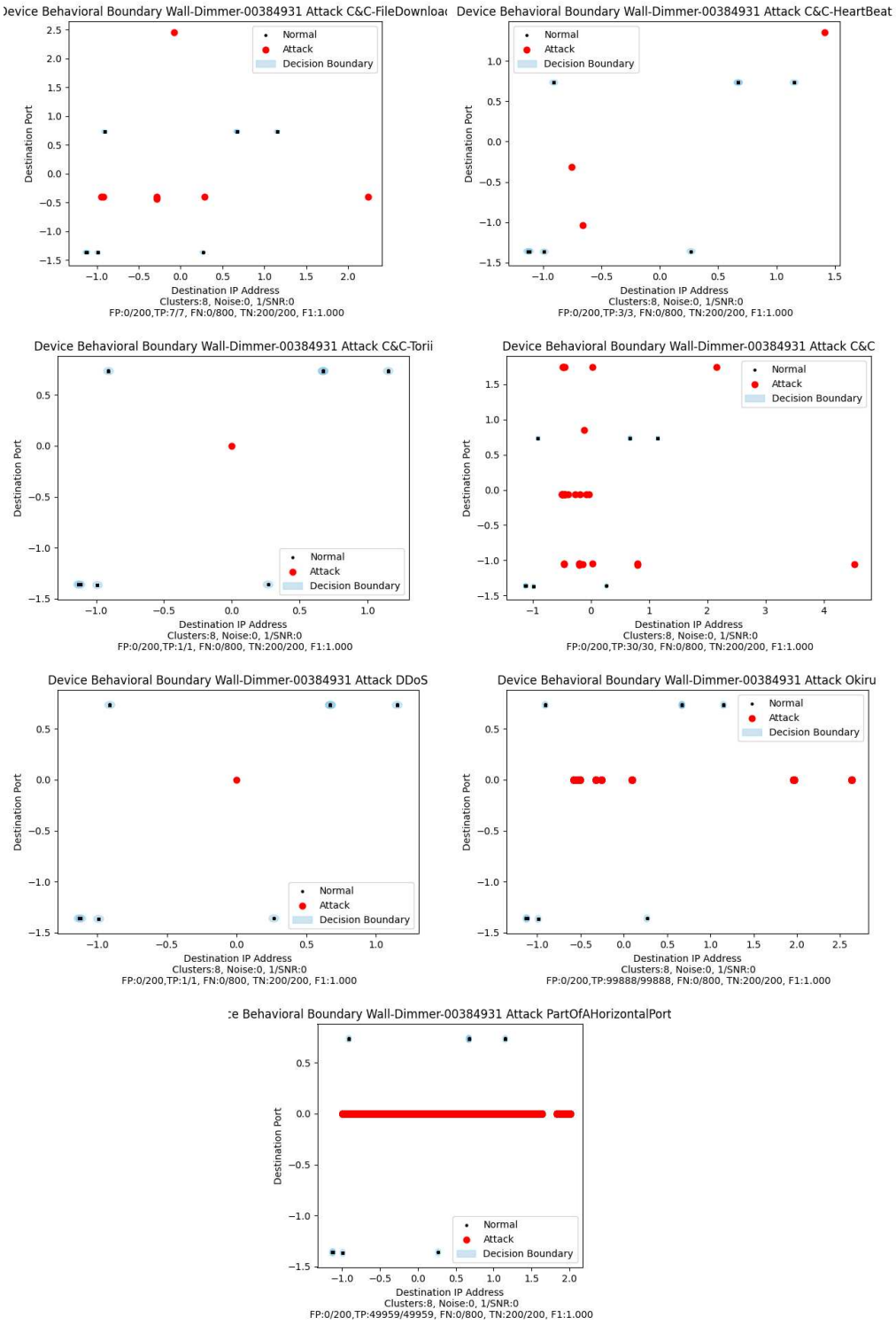


Figure C.19: Wall-Dimmer-00384931: LAB

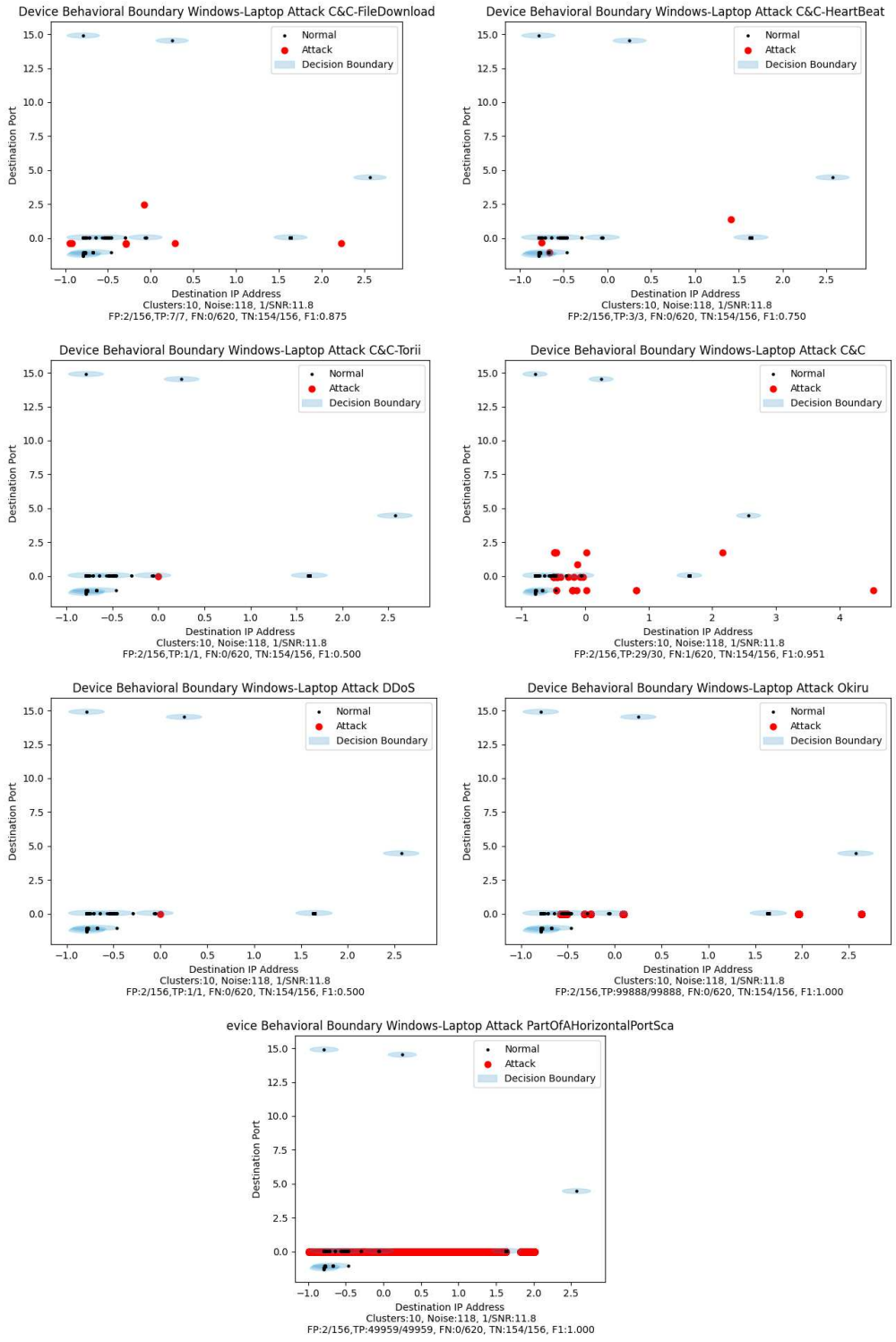


Figure C.20: Windows-Laptop: LAB

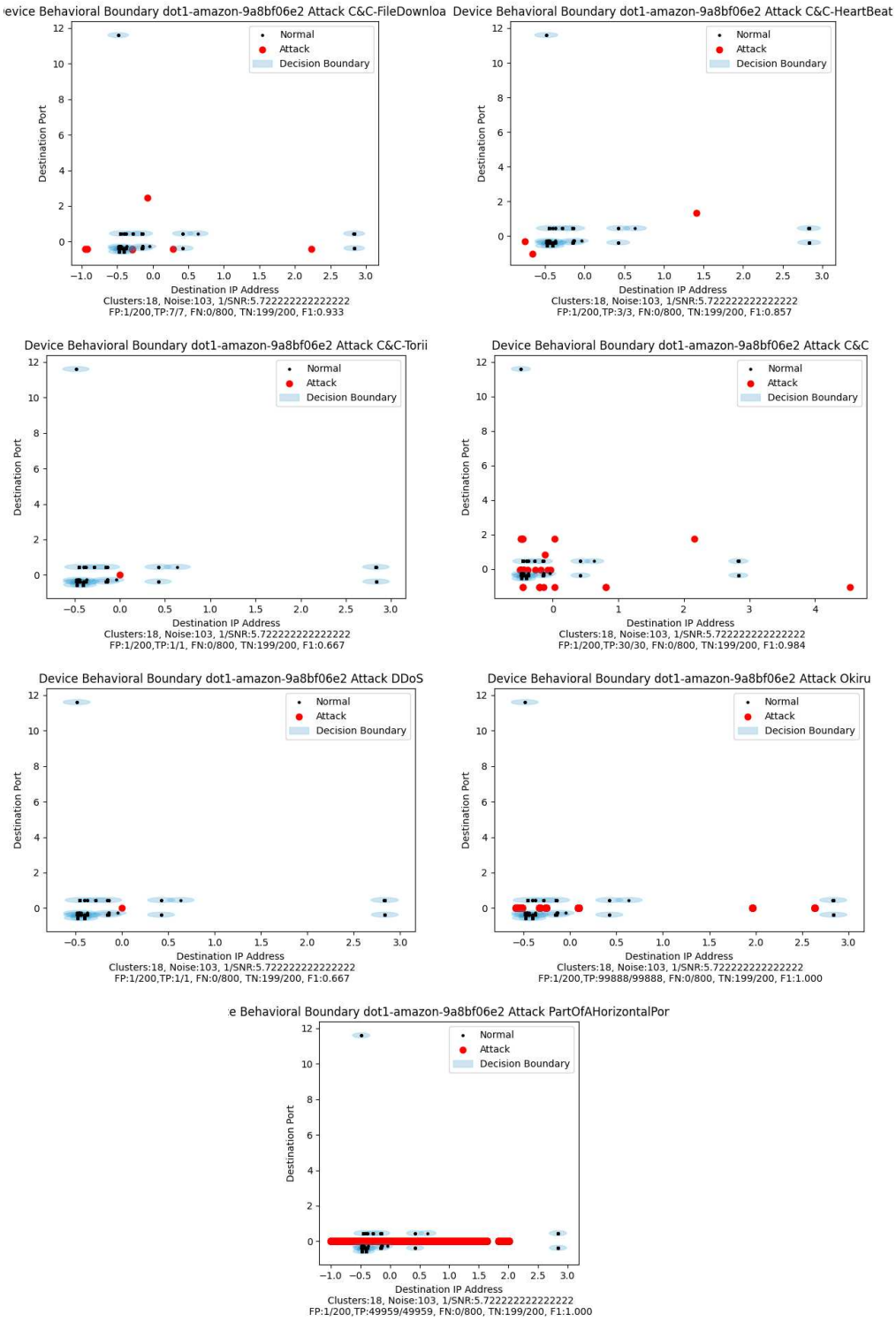


Figure C.21: dot1-amazon-9a8bf06e2: LAB

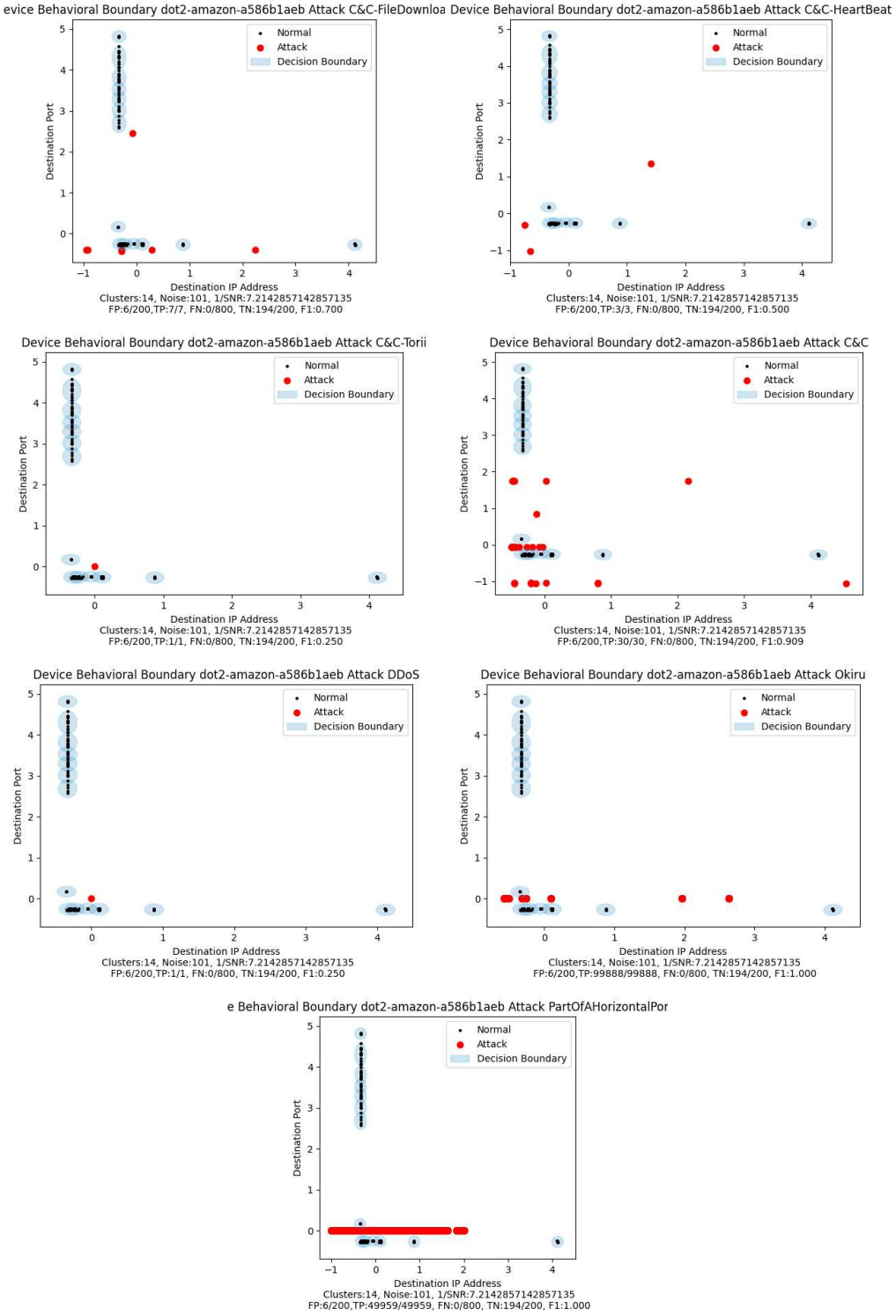


Figure C.22: dot2-amazon-a586b1aeb: LAB

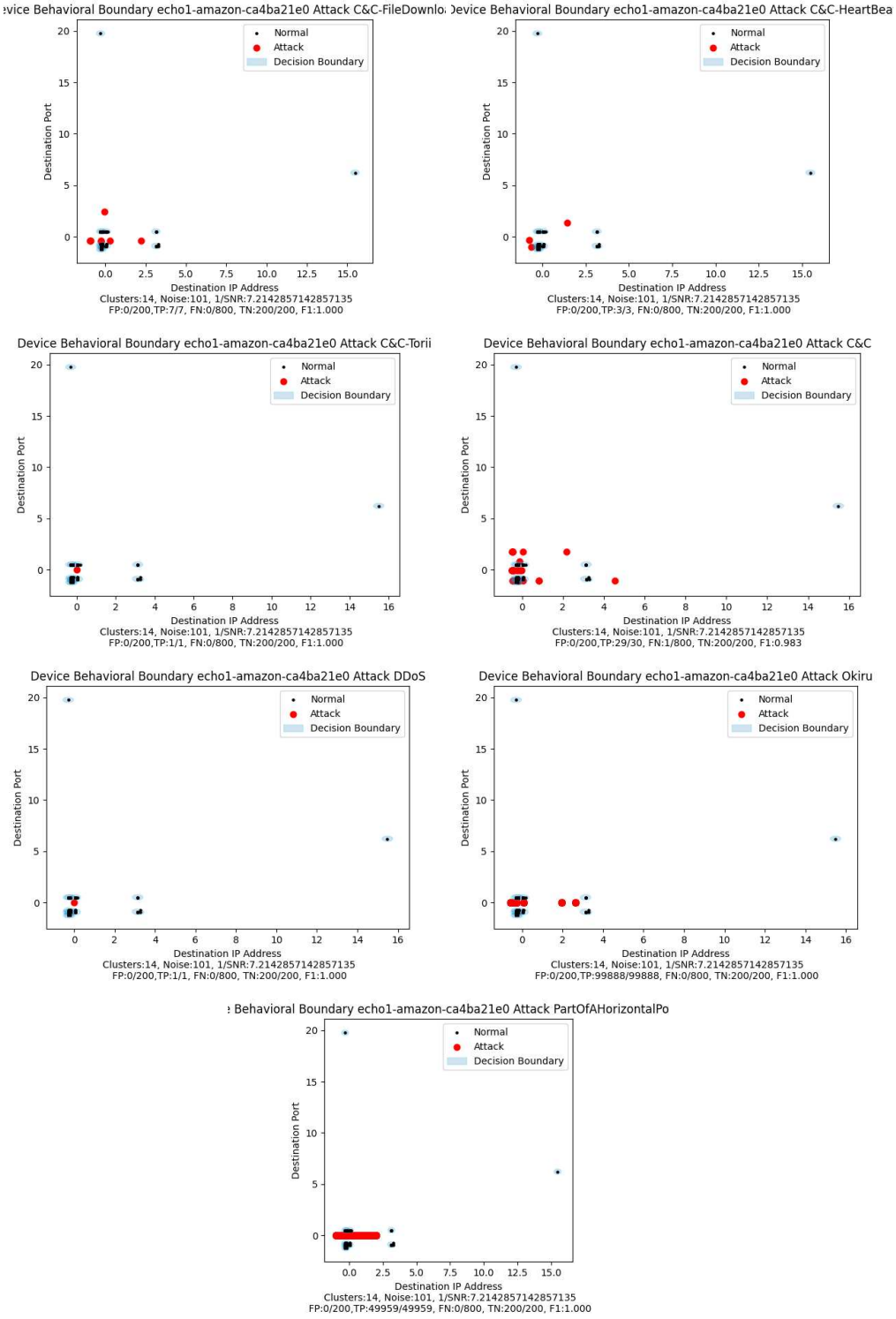


Figure C.23: echo1-amazon-ca4ba21e0: LAB

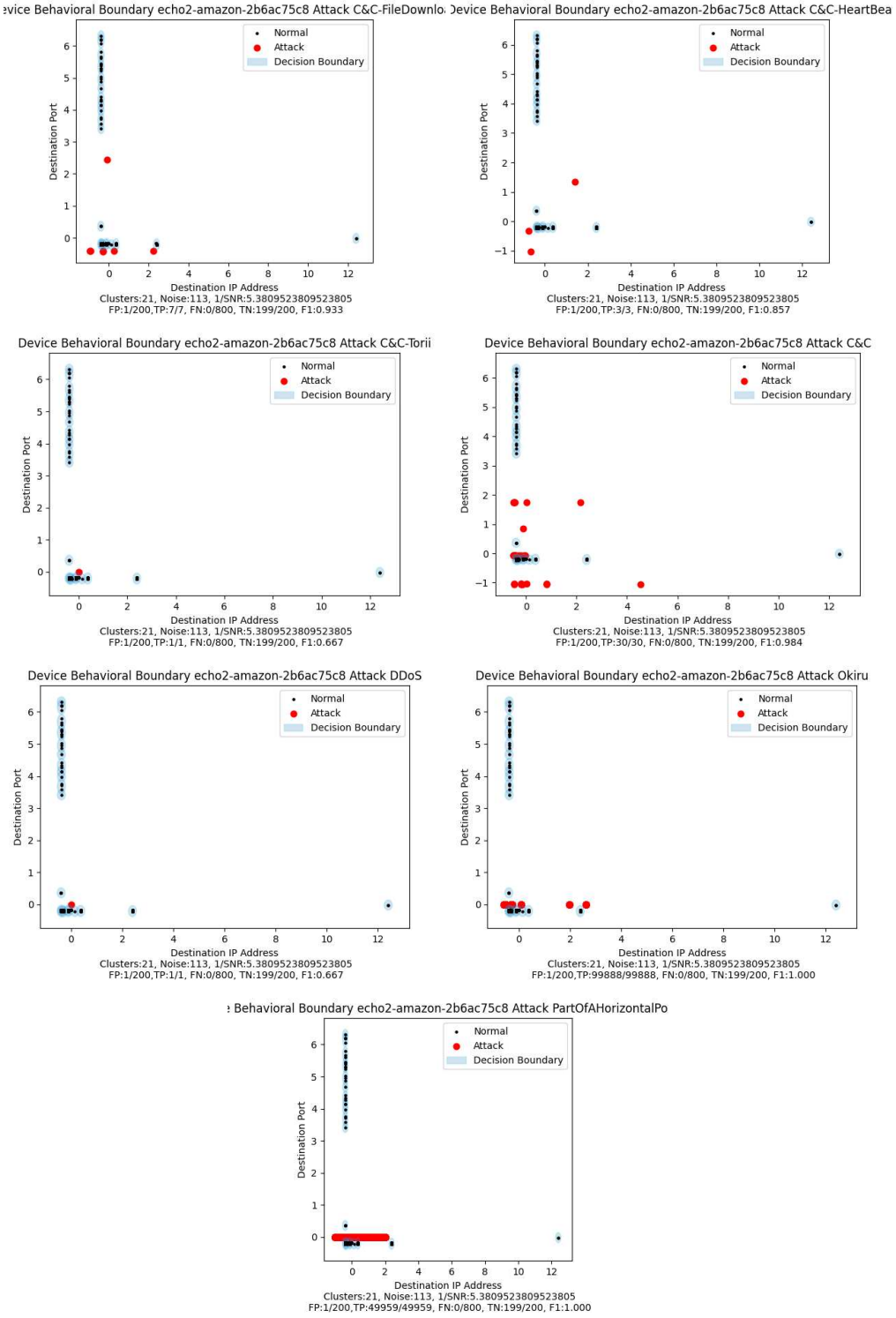


Figure C.24: echo2-amazon-2b6ac75c8: LAB

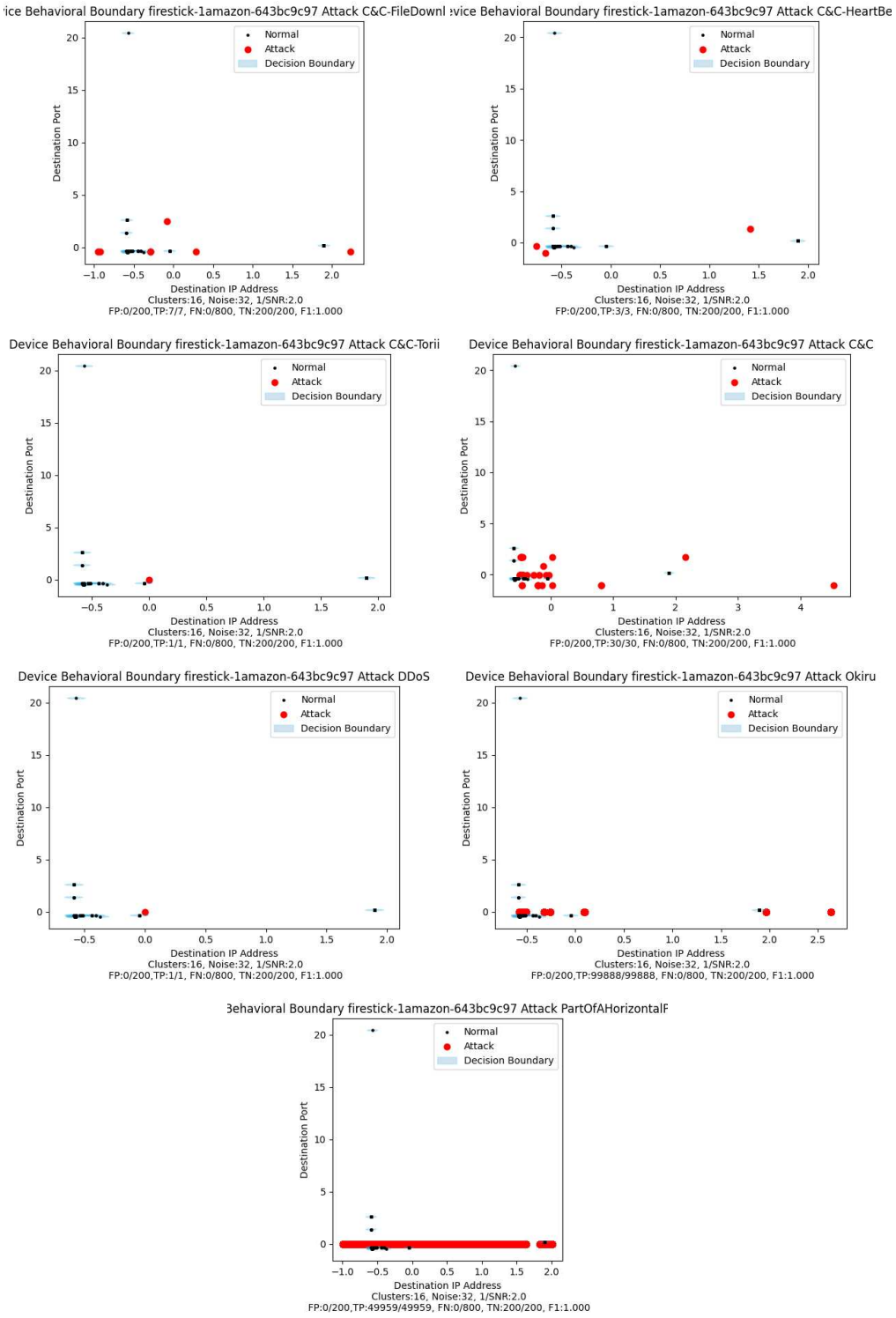


Figure C.25: firestick-1amazon-643bc9c97: LAB

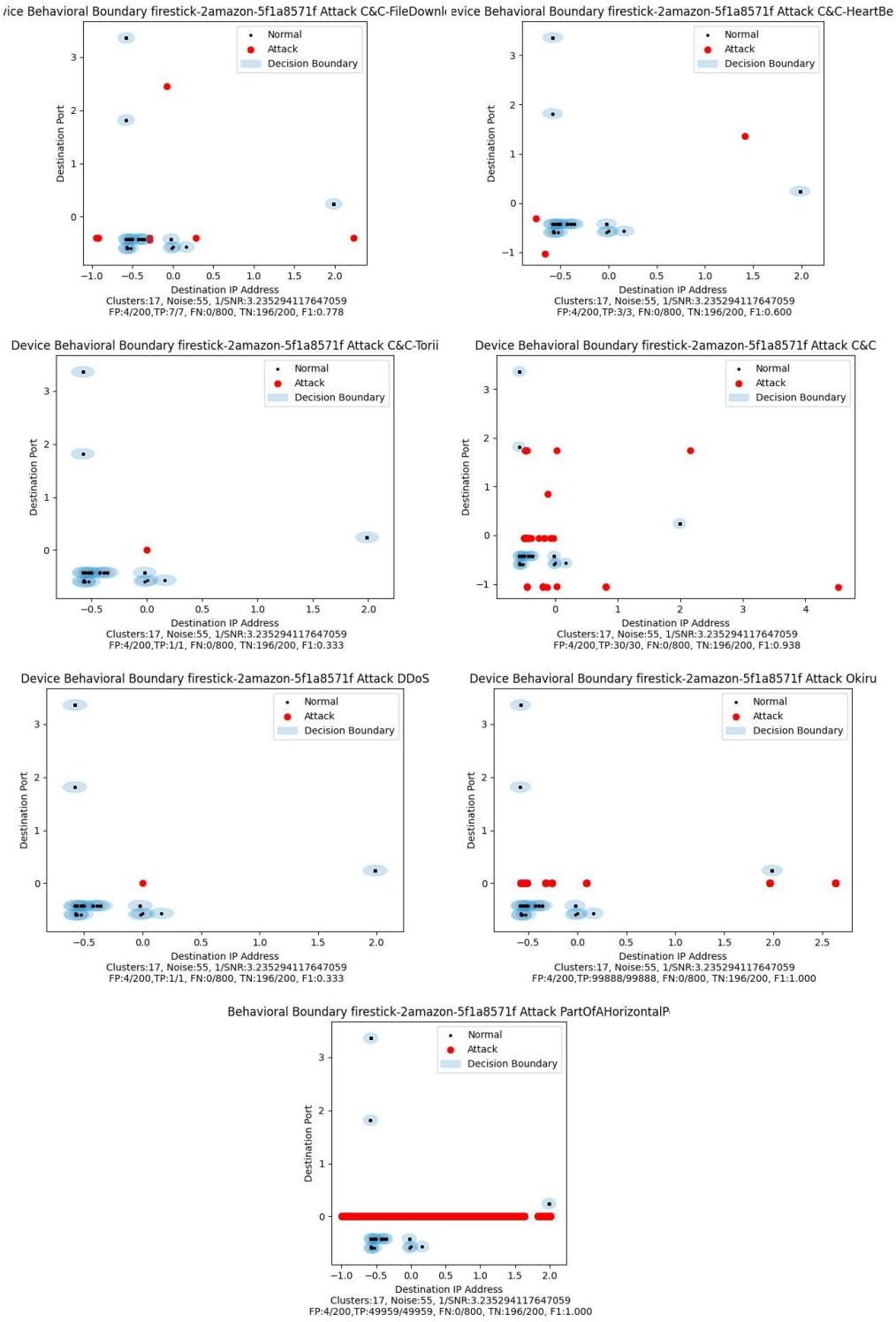


Figure C.26: firestick-2amazon-5f1a8571f: LAB

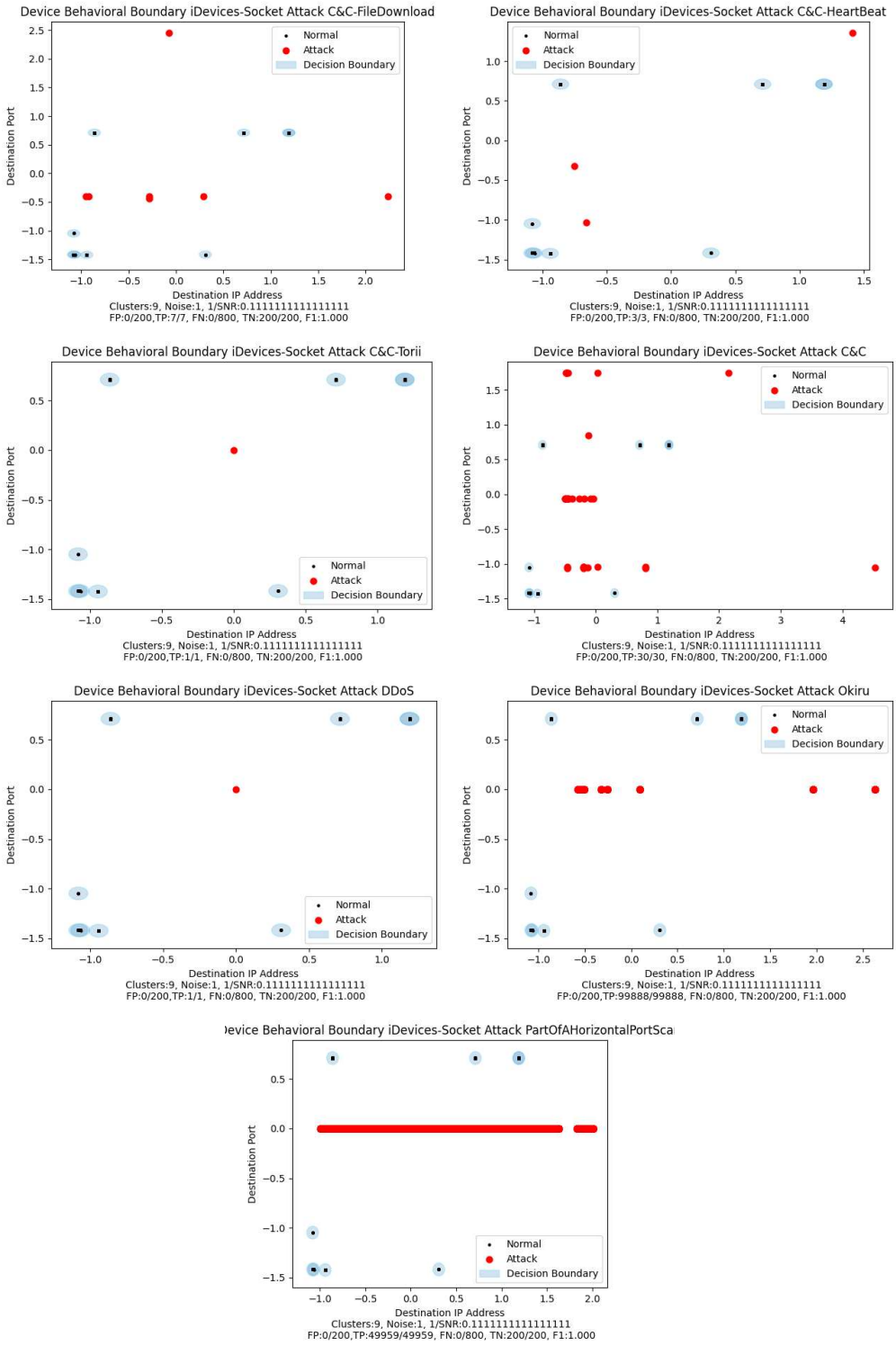


Figure C.27: iDevices-Socket: LAB

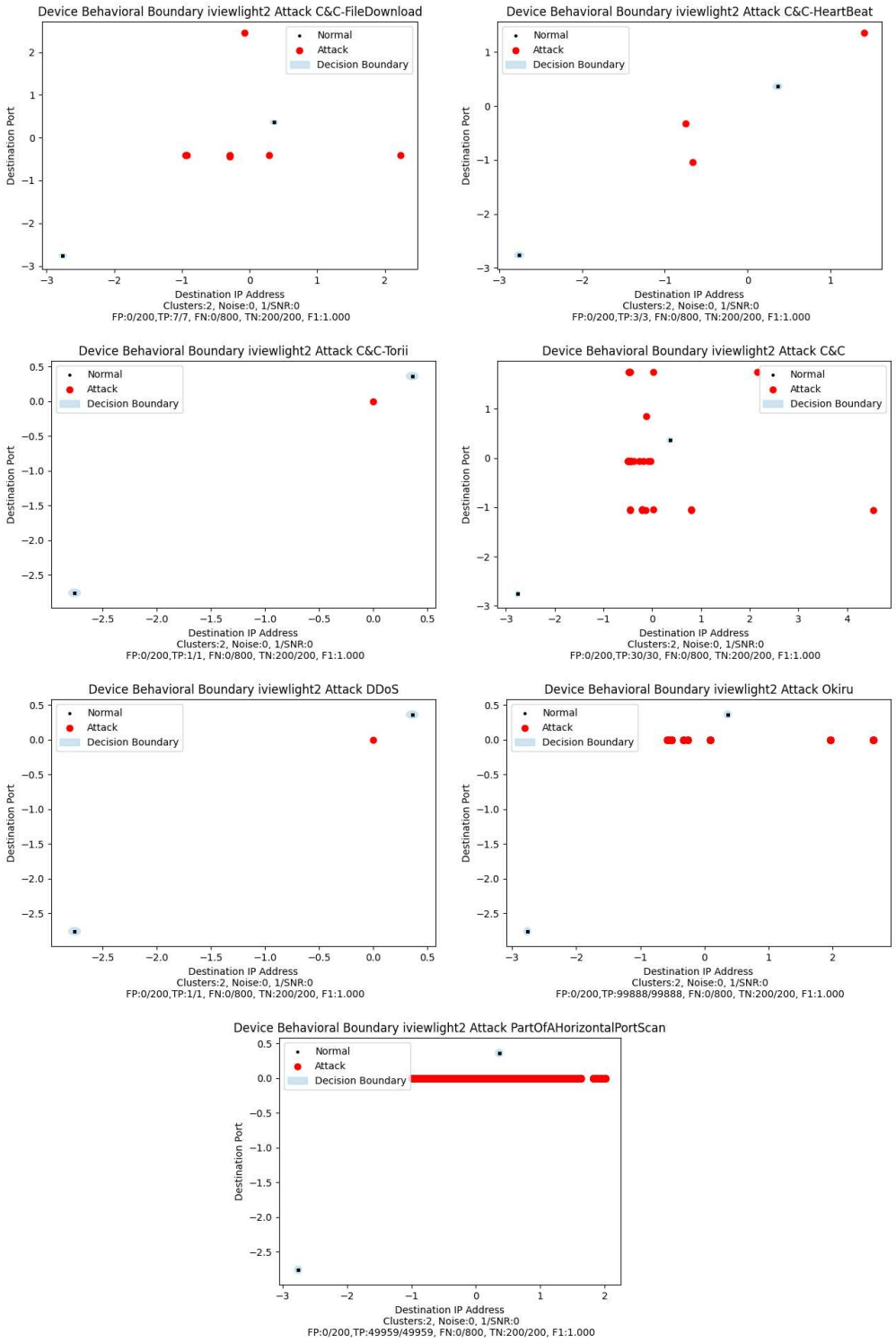


Figure C.28: iviewlight2: LAB

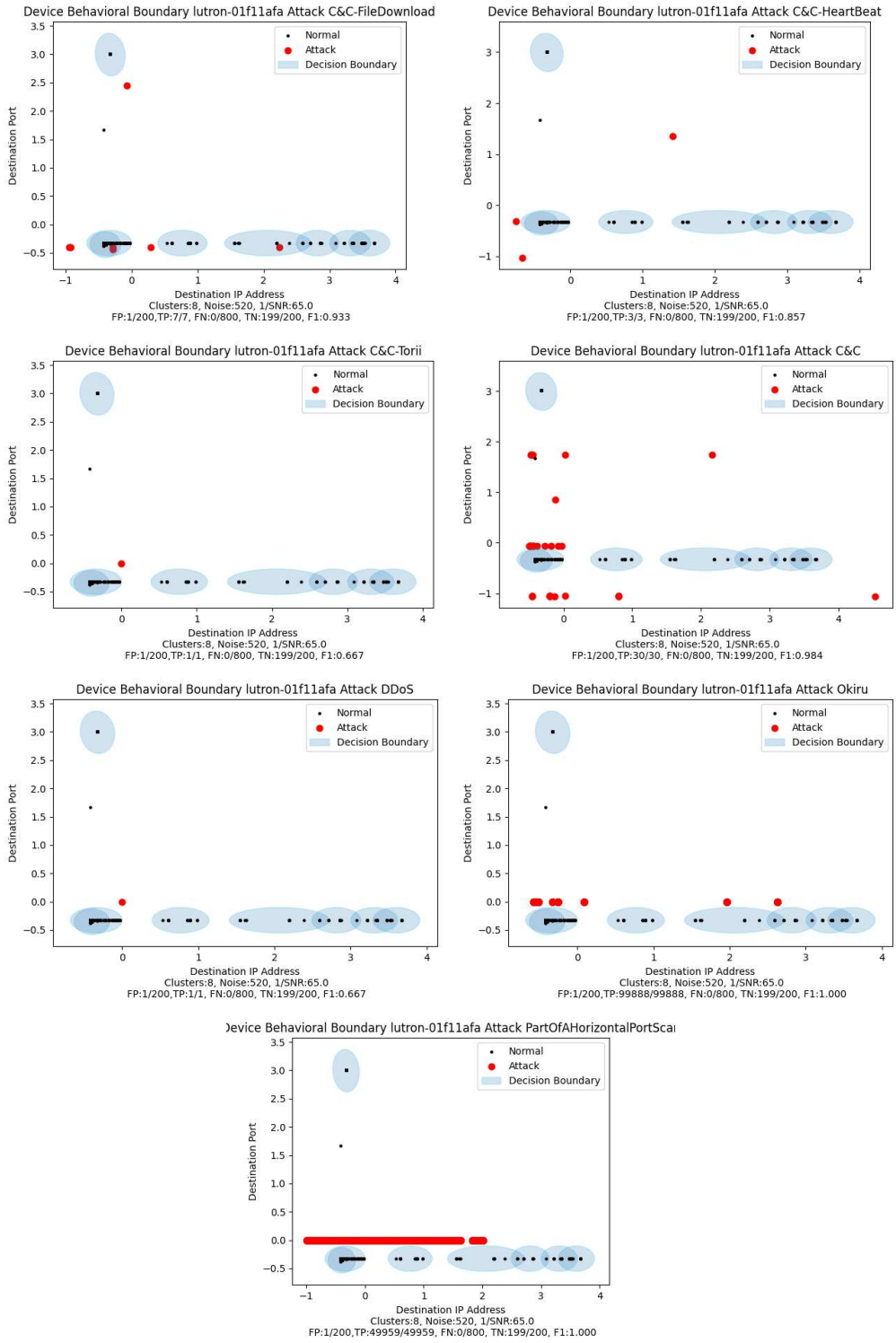


Figure C.29: lutron-01f11afa: LAB

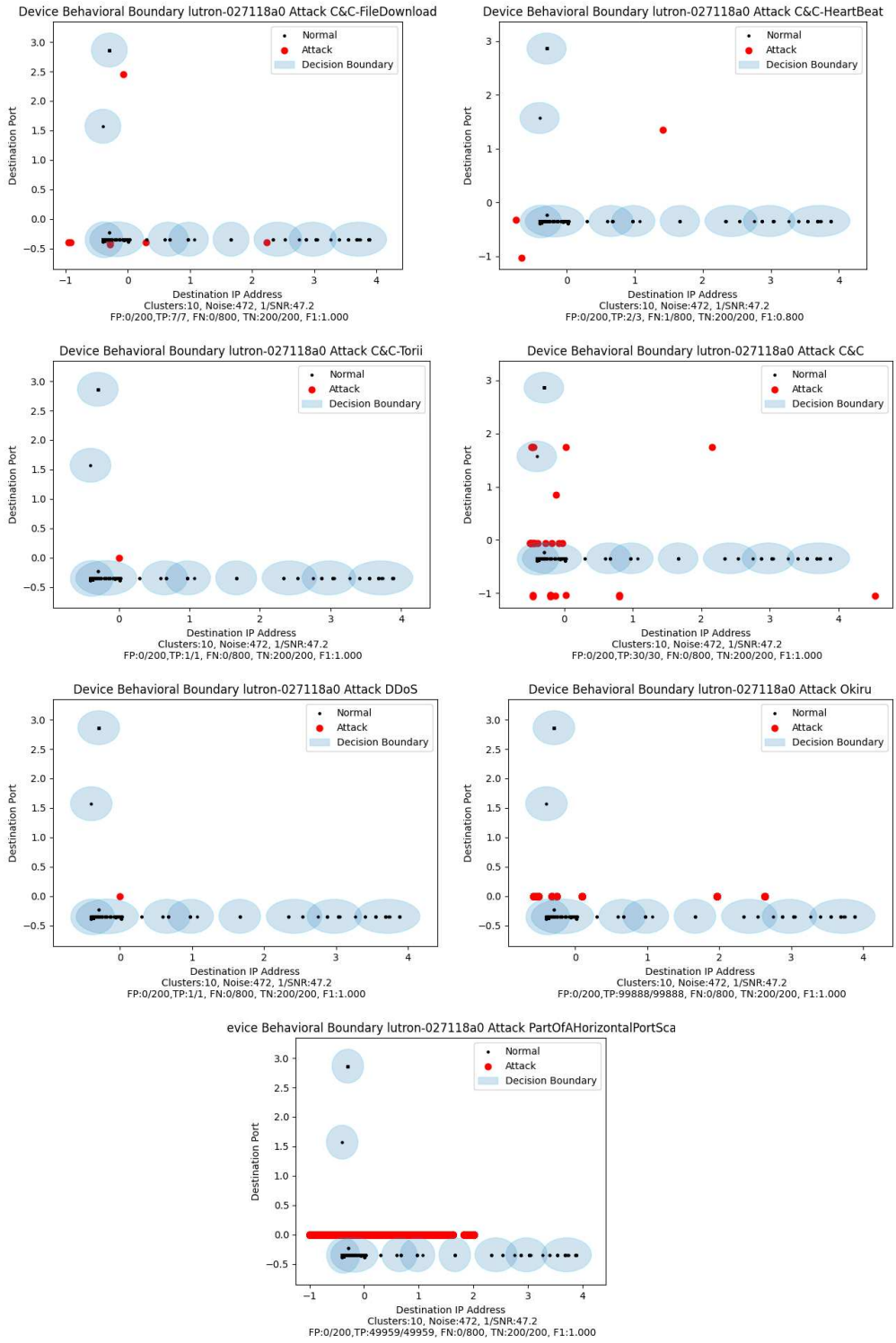


Figure C.30: lutron-027118a0: LAB

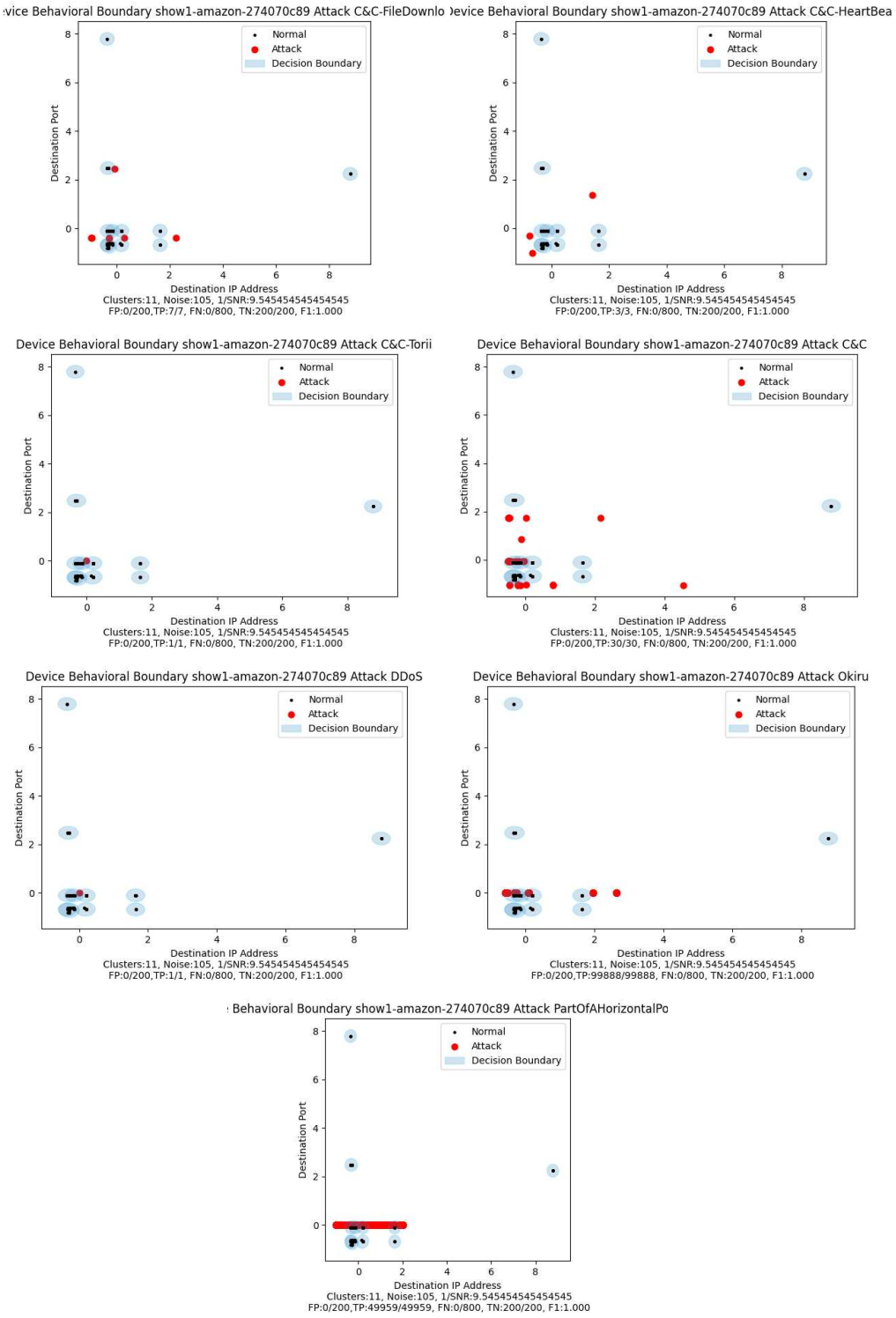


Figure C.31: show1-amazon-274070c89: LAB

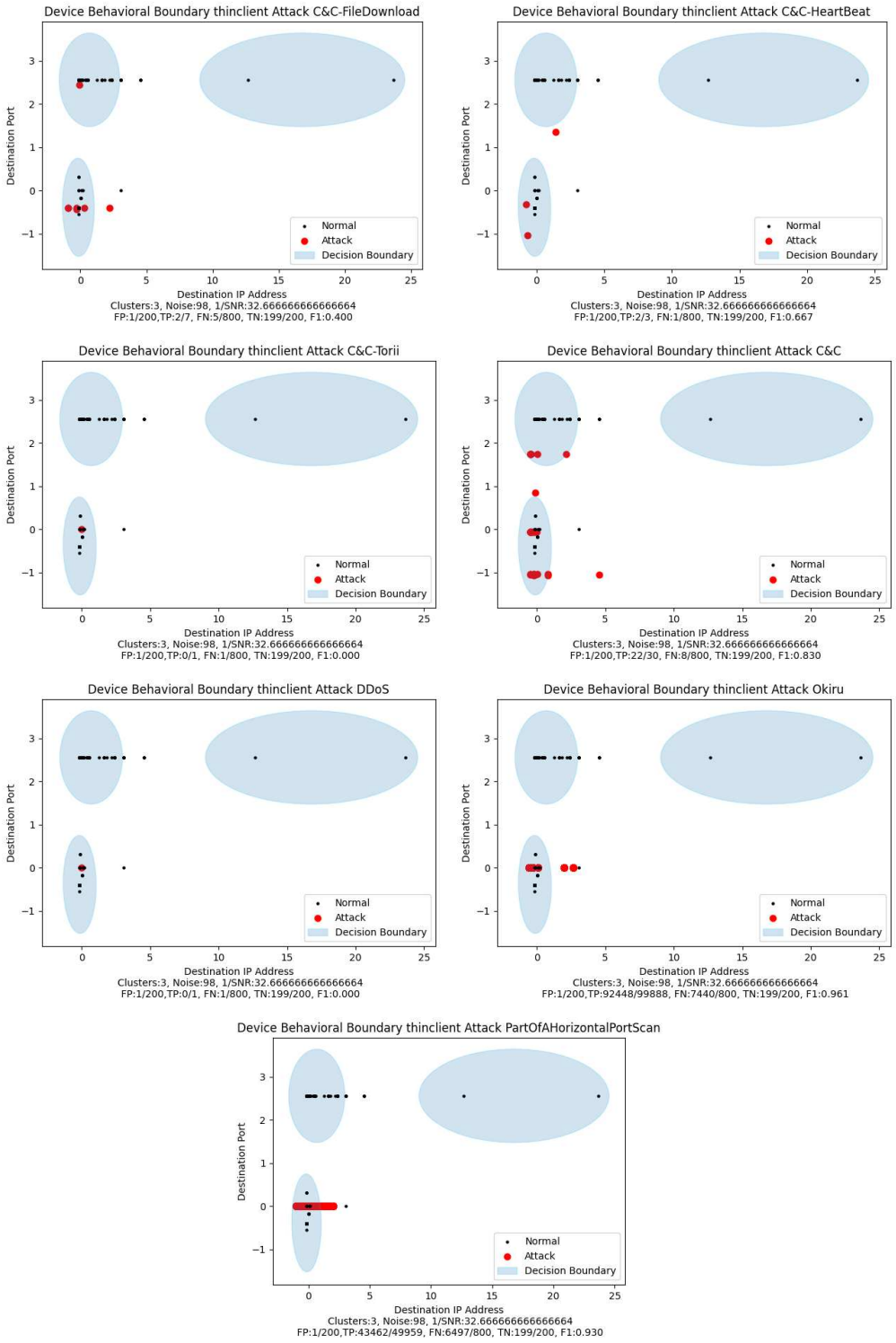


Figure C.32: thinclient: LAB

Appendix D

UNSW

This Appendix shows the Gaussian Boundary for each device in the UNSW dataset against all seven attacks listed in 3.6.

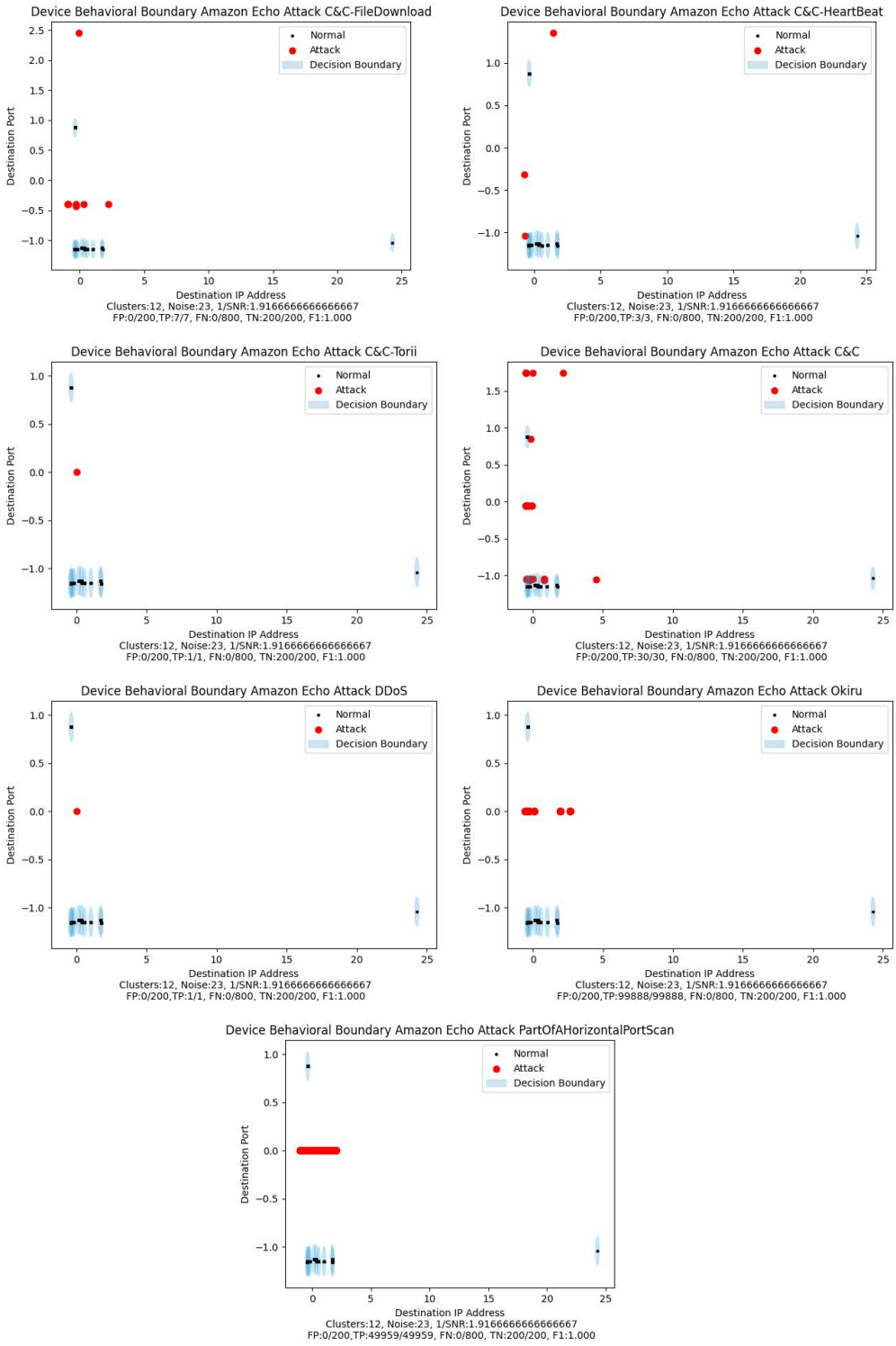


Figure D.1: Amazon-Echo: UNSW

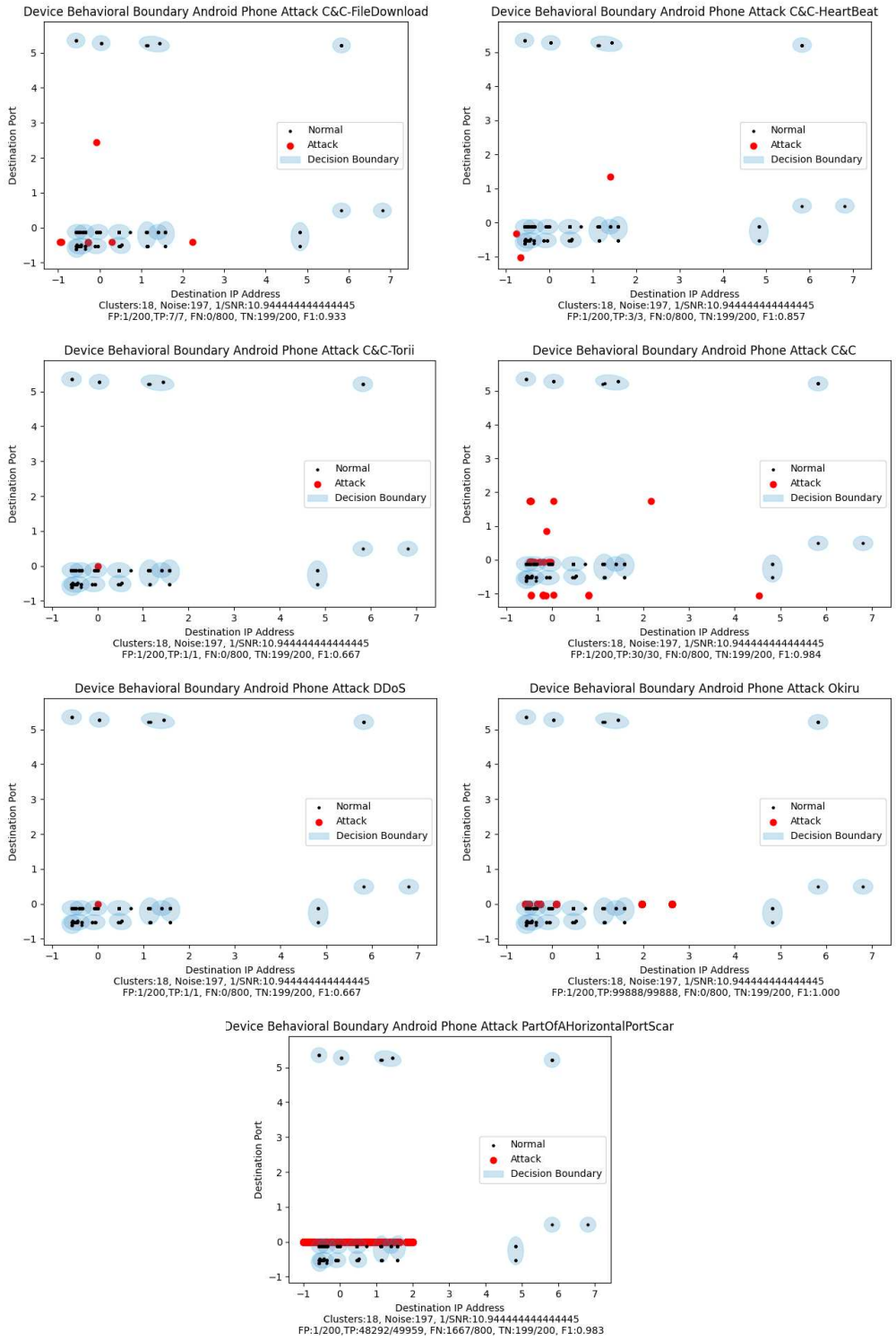


Figure D.2: Android-Phone: UNSW

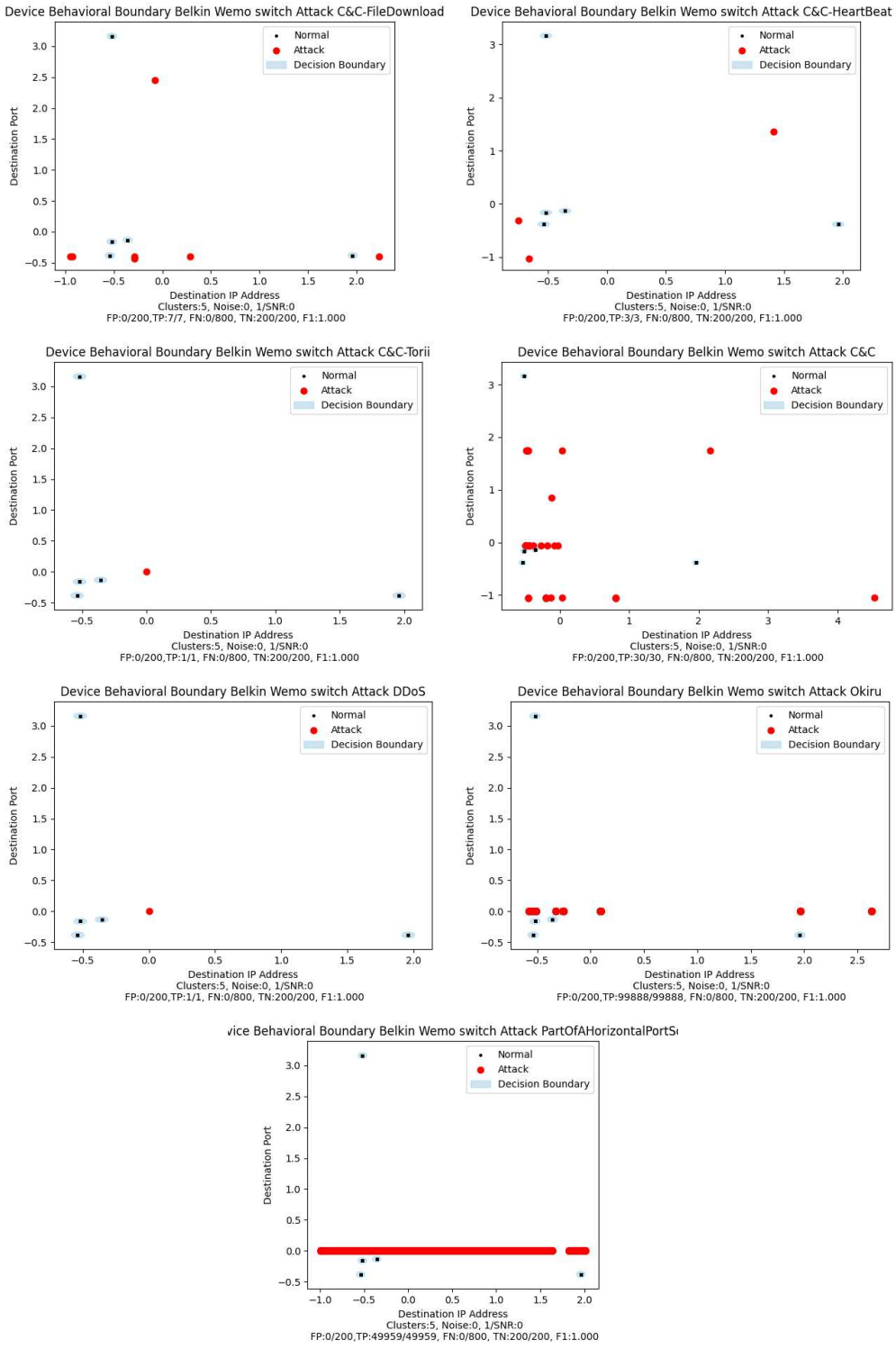


Figure D.3: Belkin-Wemo-switch: UNSW

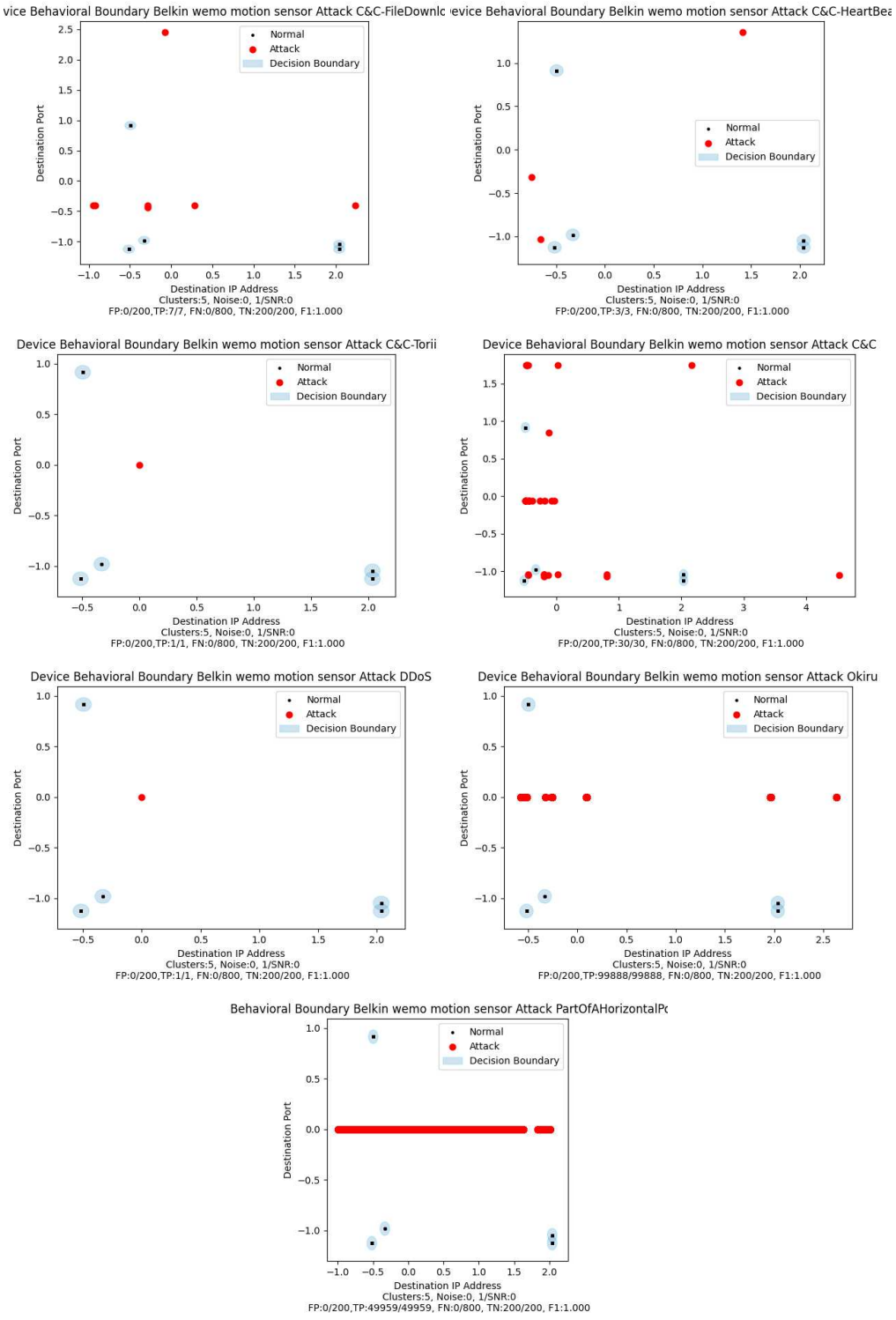


Figure D.4: Belkin-wemo-motion-sensor: UNSW

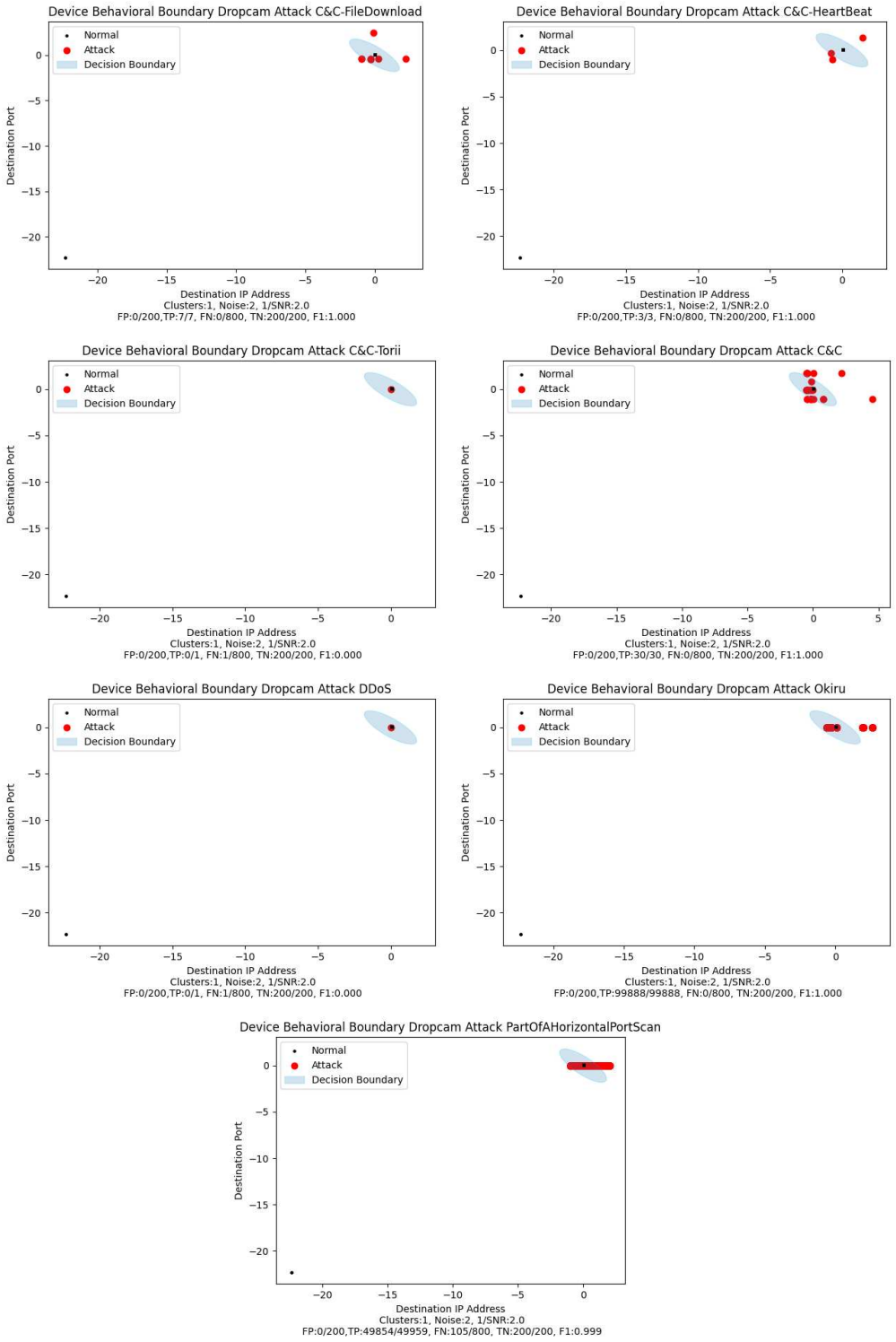


Figure D.5: Dropcam: UNSW

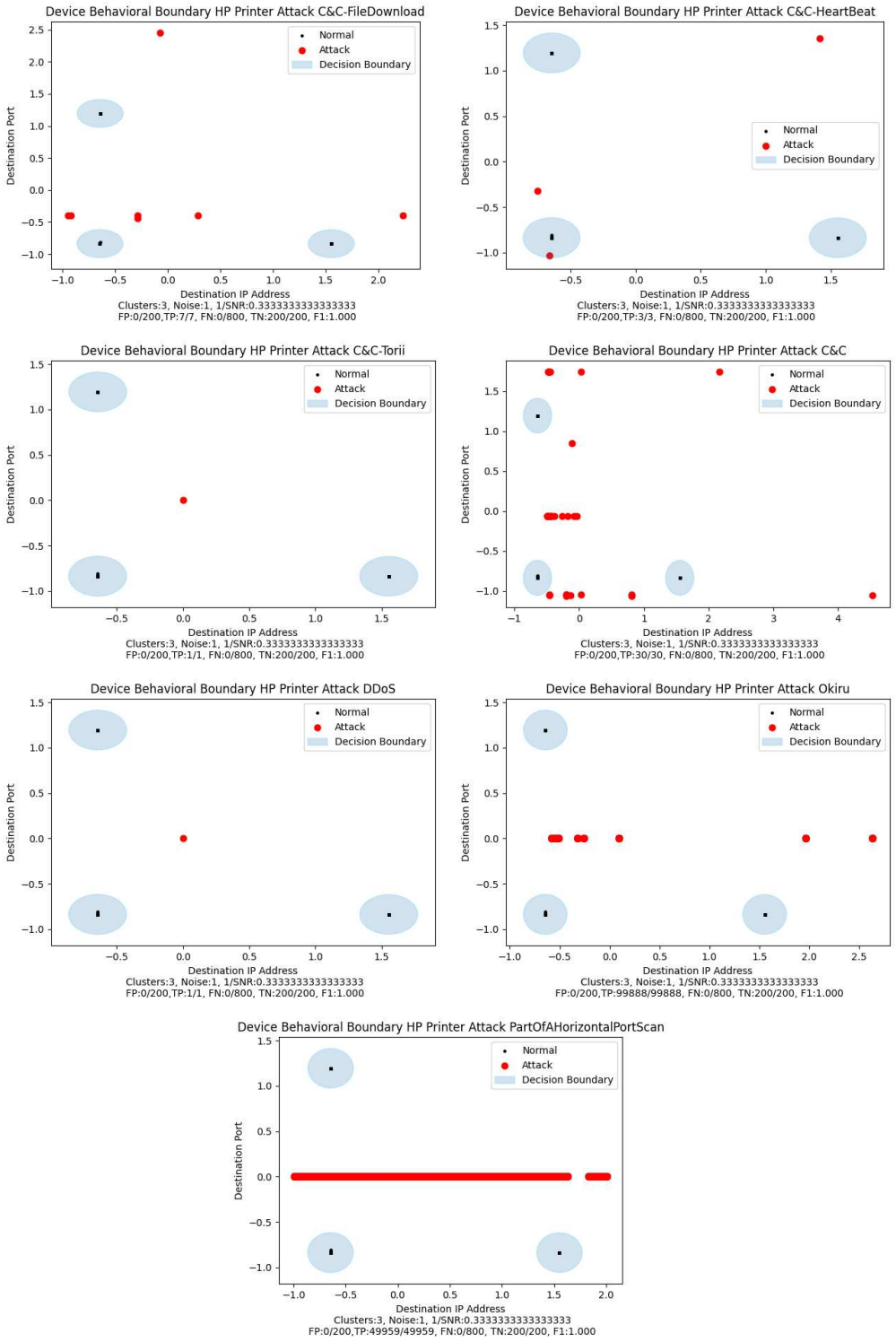


Figure D.6: HP-Printer: UNSW

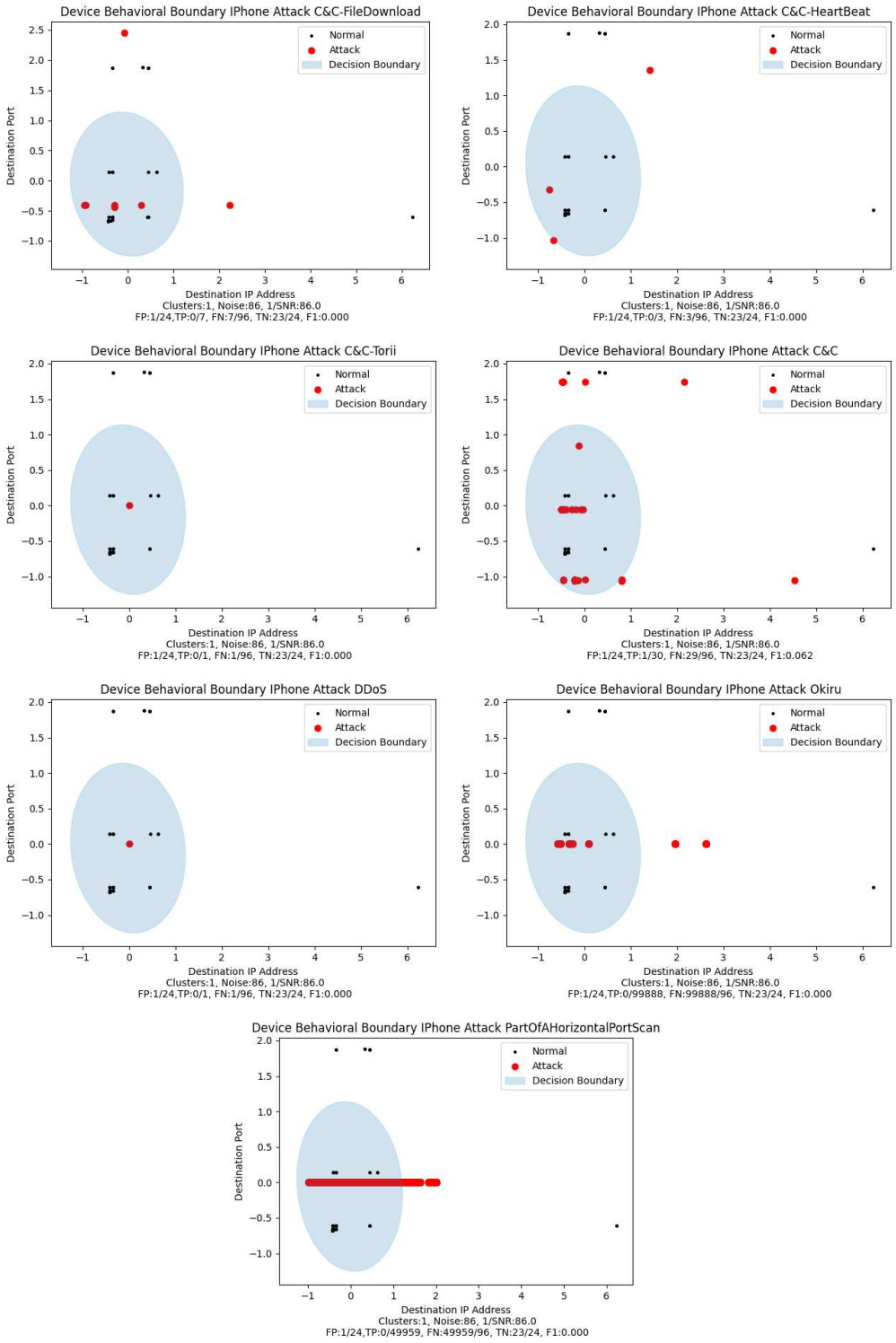


Figure D.7: IPhone: UNSW

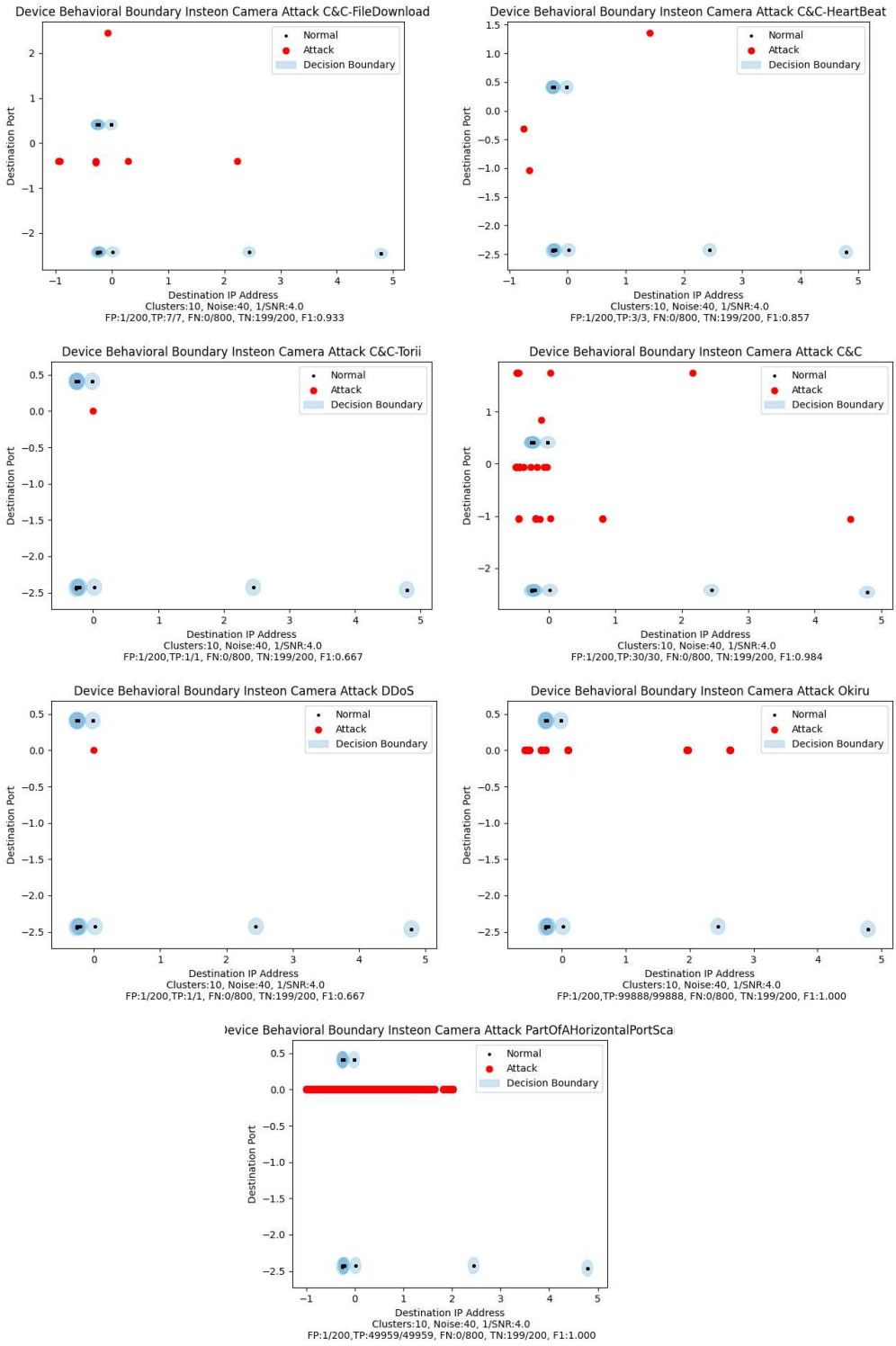


Figure D.8: Insteon-Camera: UNSW

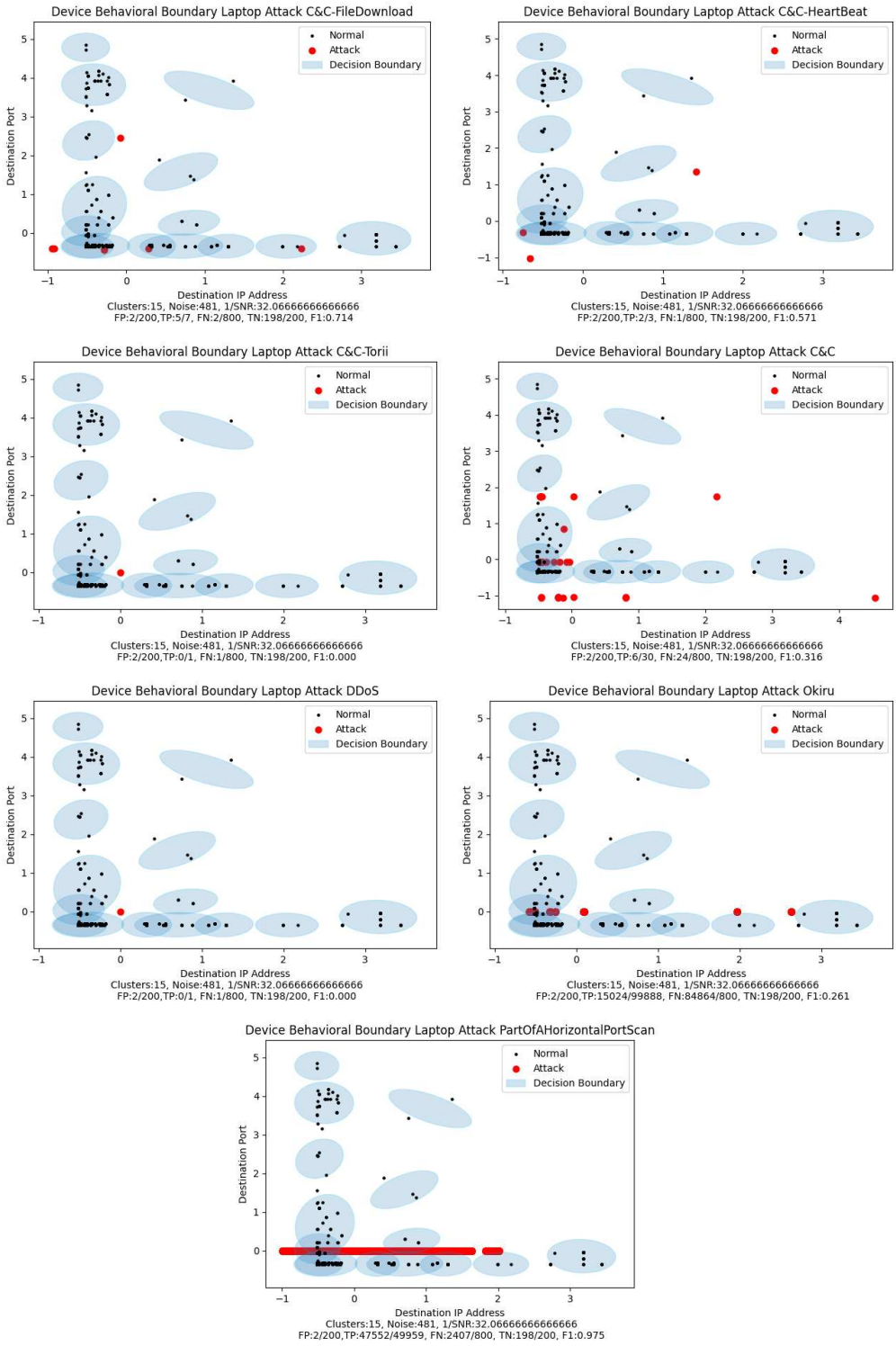


Figure D.9: Laptop: UNSW

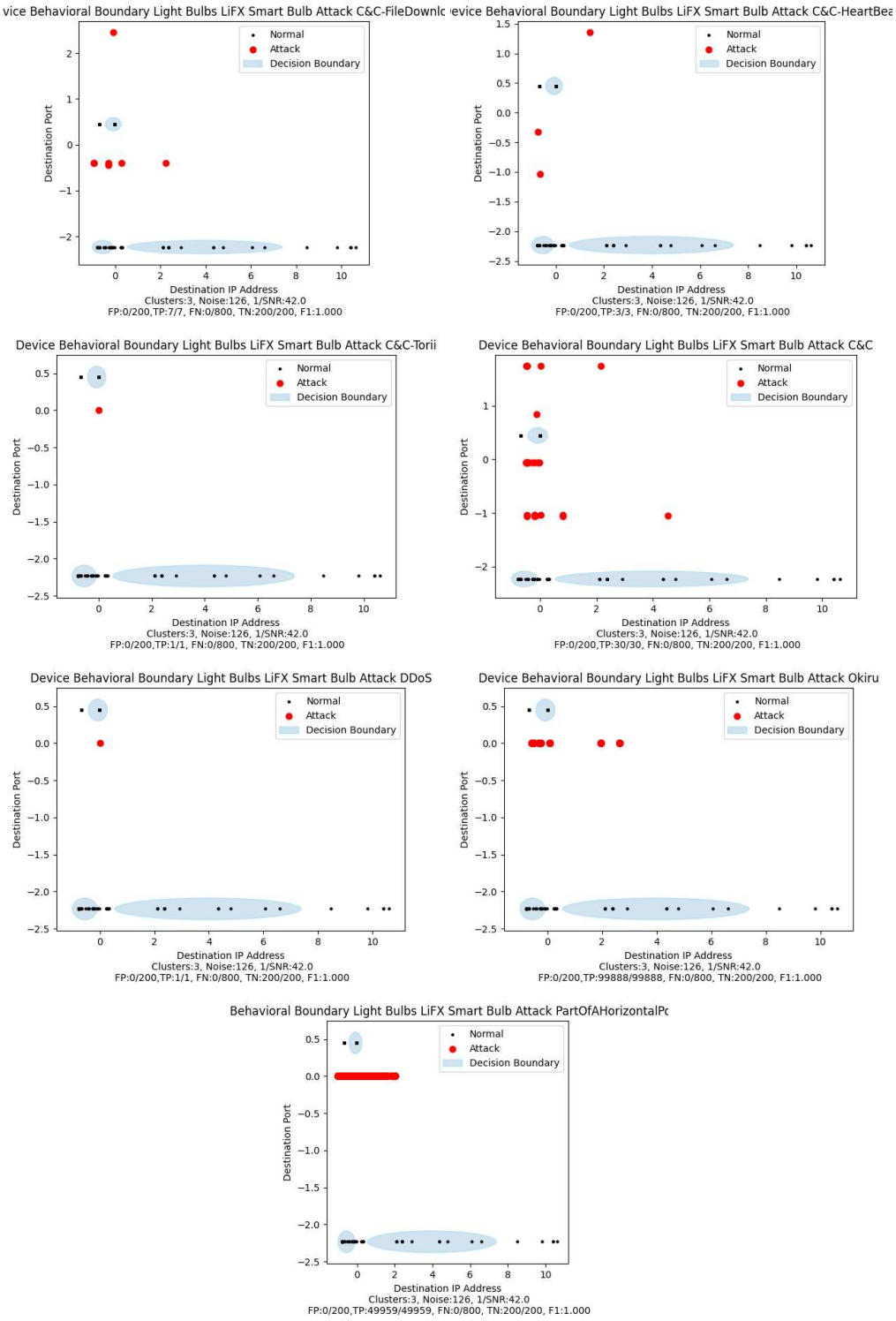


Figure D.10: Light-Bulbs-LiFX-Smart-Bulb: UNSW

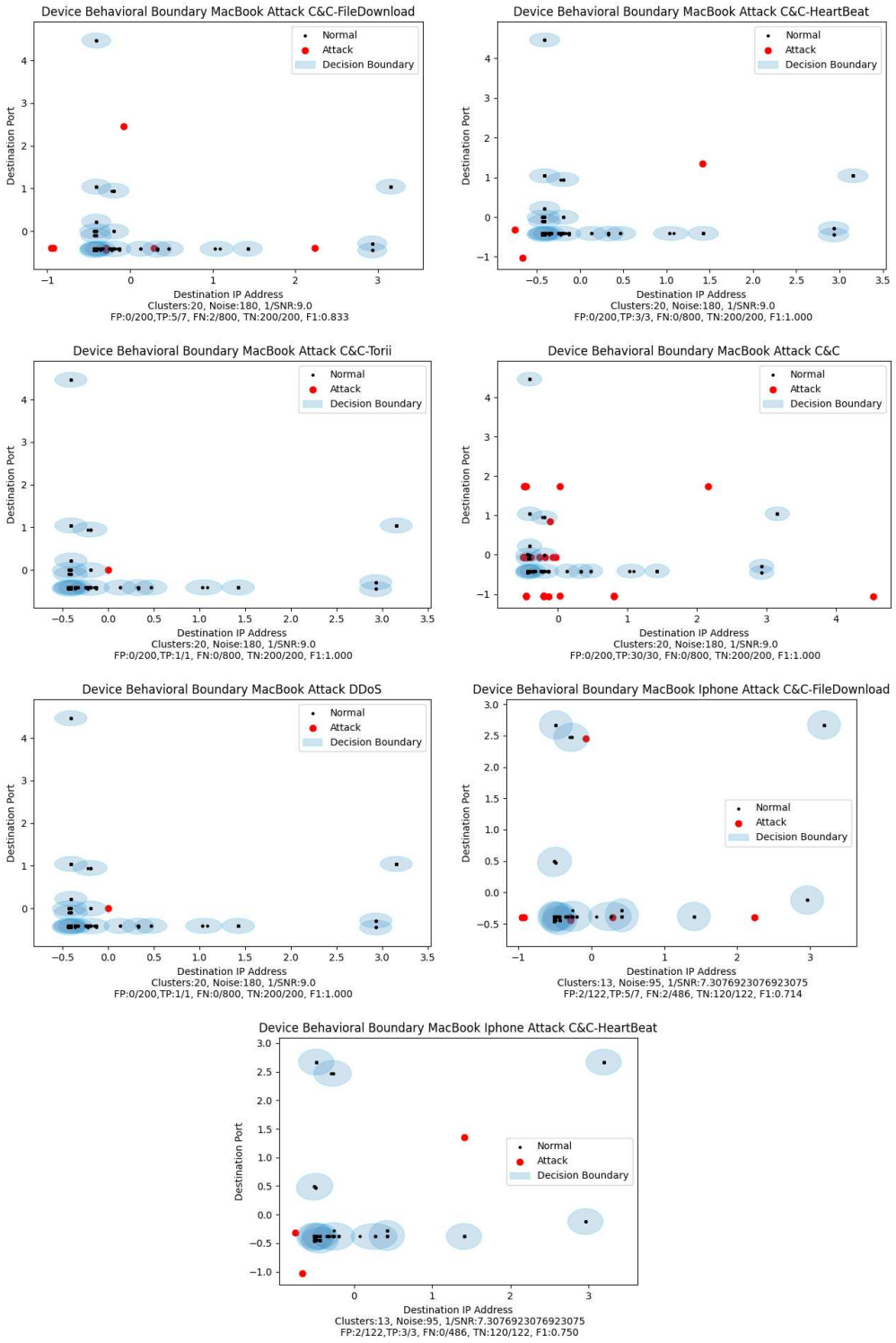


Figure D.11: MacBook-Iphone: UNSW

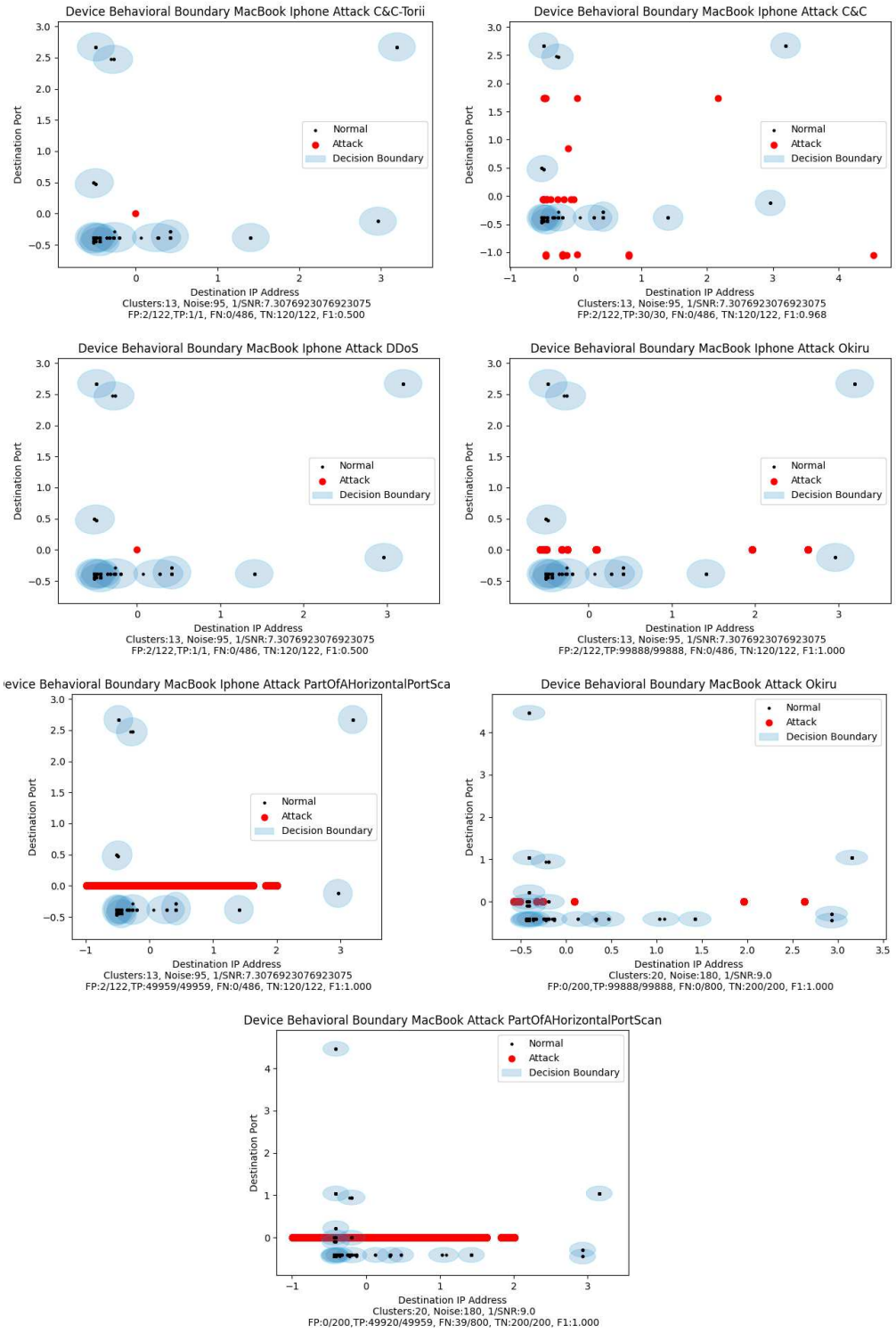


Figure D.12: MacBook: UNSW

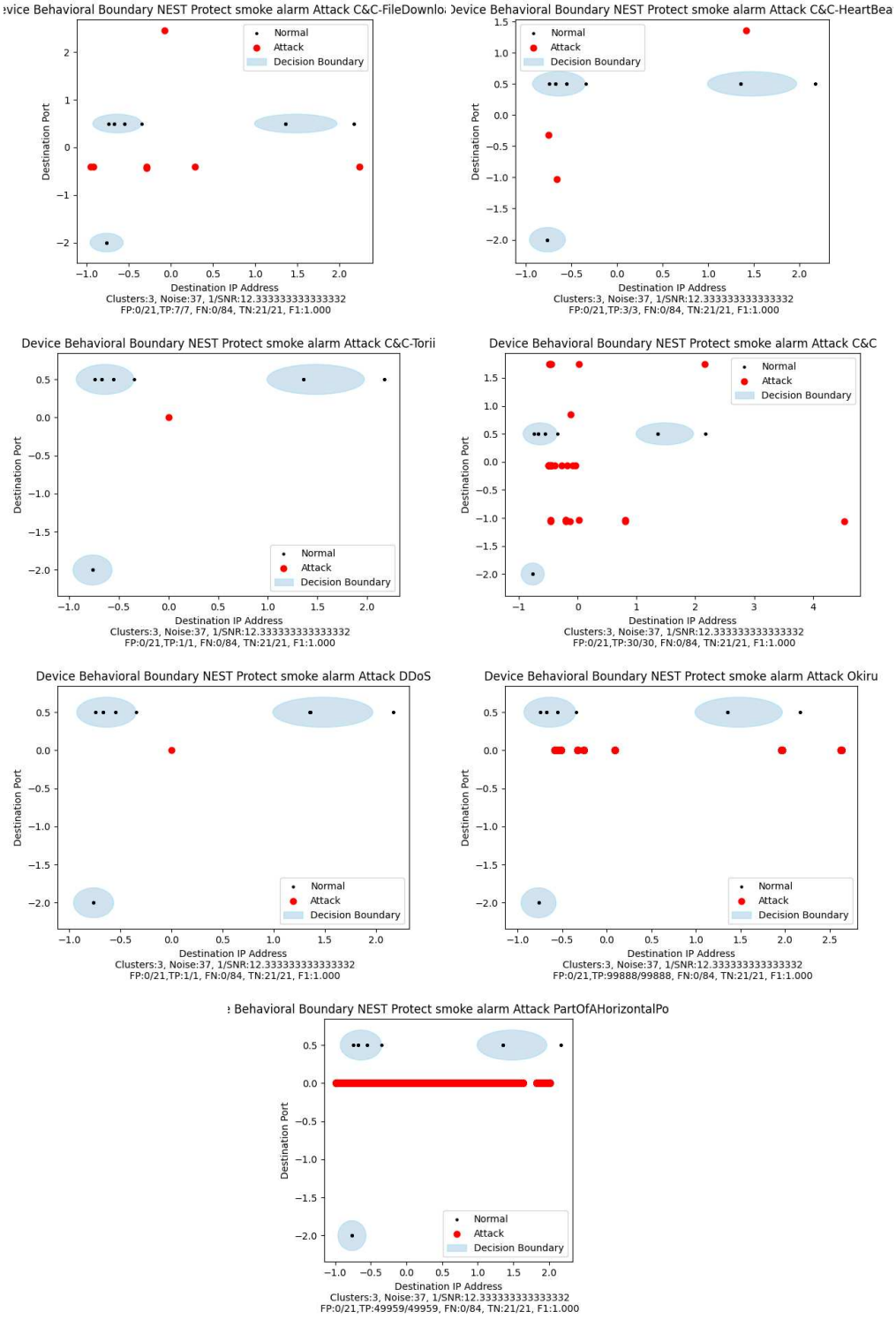


Figure D.13: NEST-Protect-smoke-alarm: UNSW

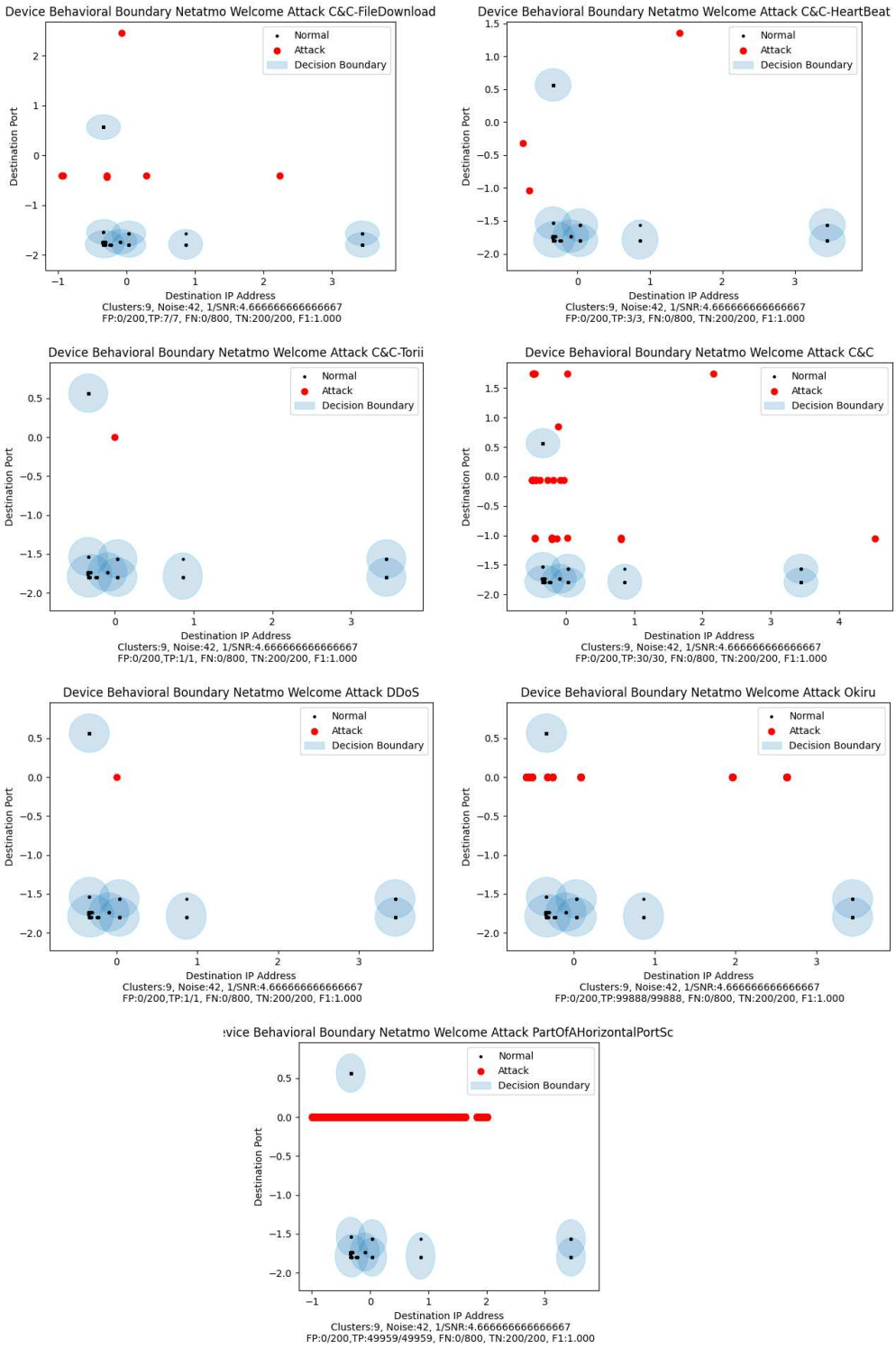


Figure D.14: Netatmo-Welcome: UNSW

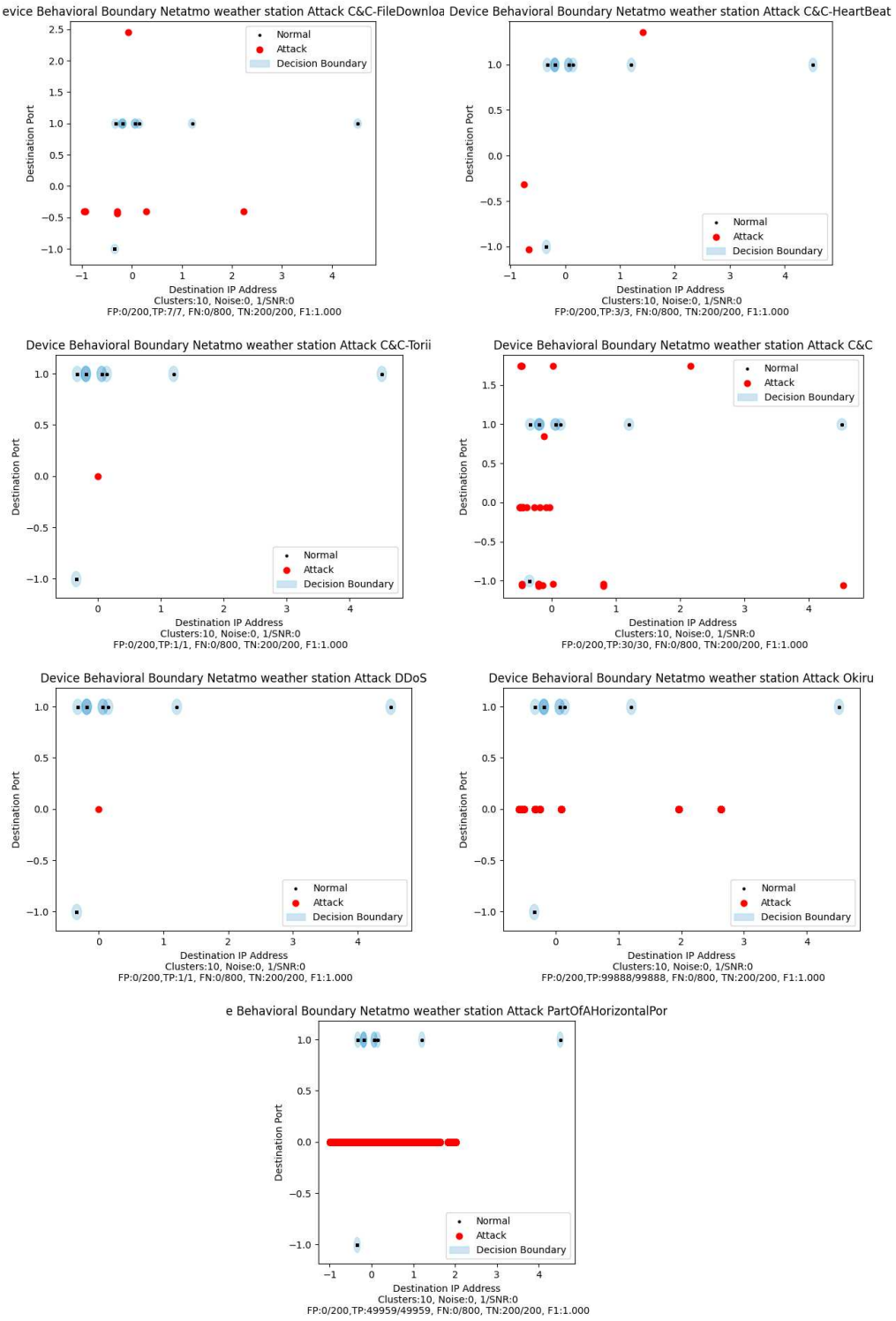


Figure D.15: Netatmo-weather-station: UNSW

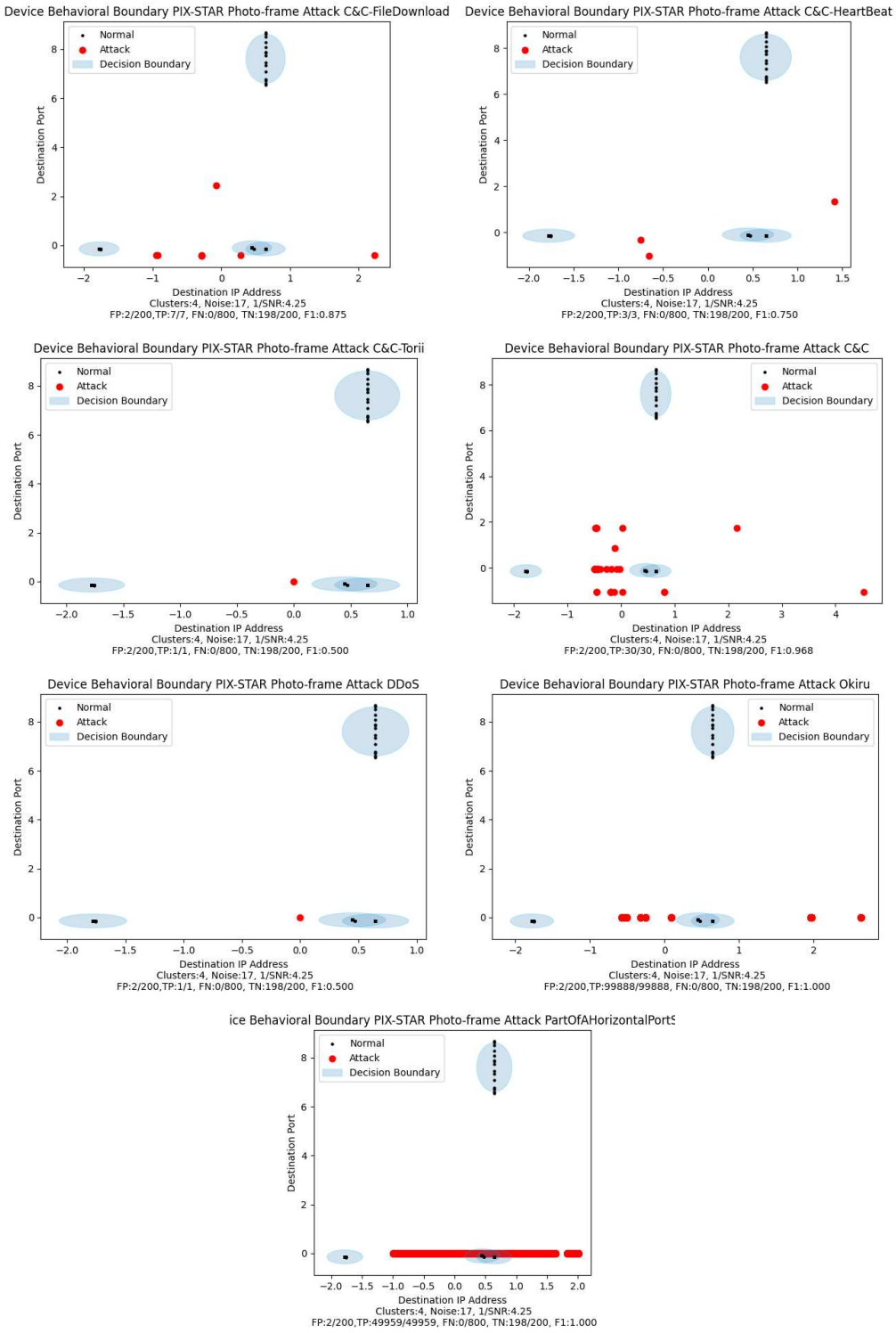


Figure D.16: PIX-STAR-Photo-frame: UNSW

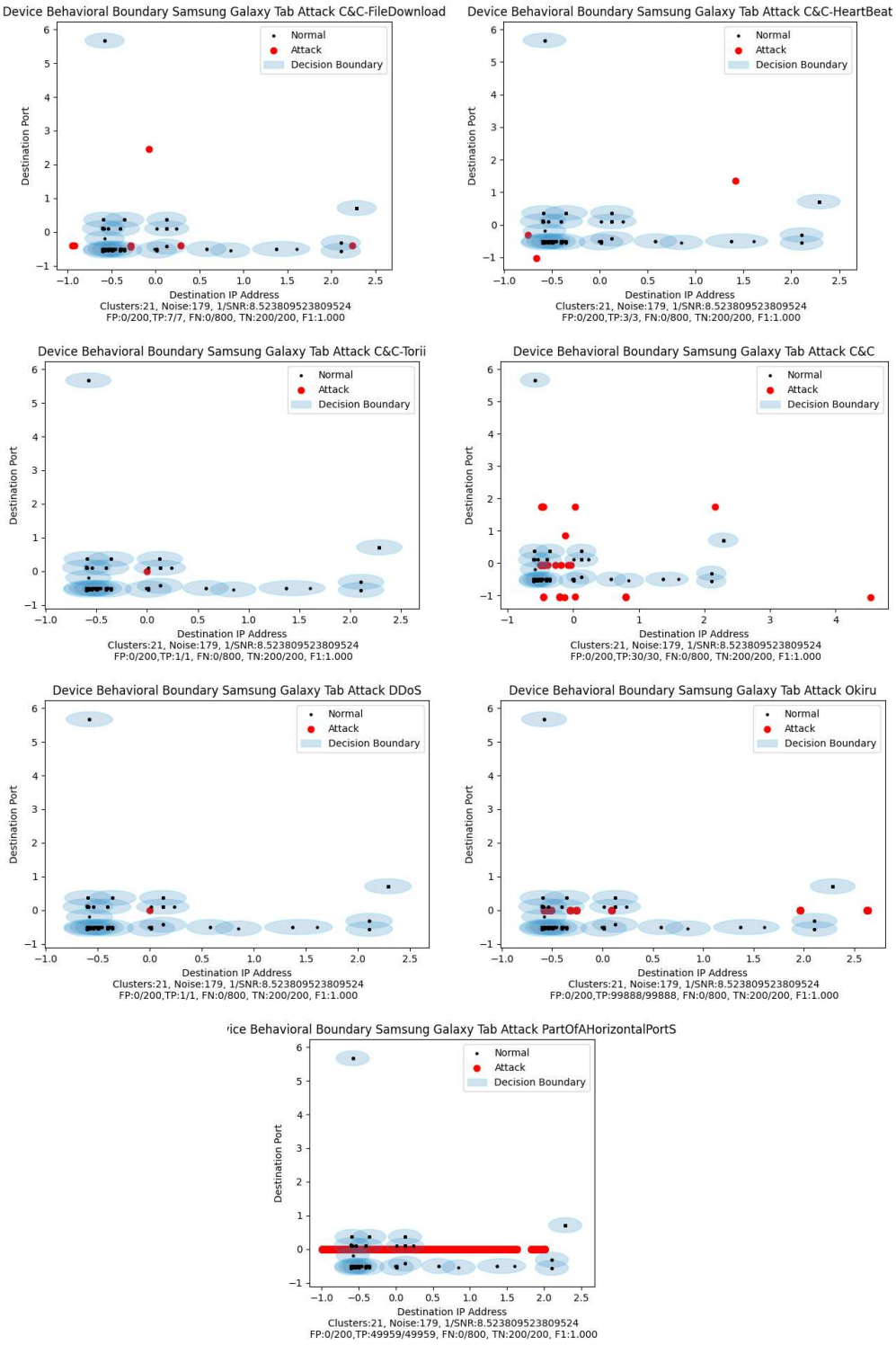


Figure D.17: Samsung-Galaxy-Tab: UNSW

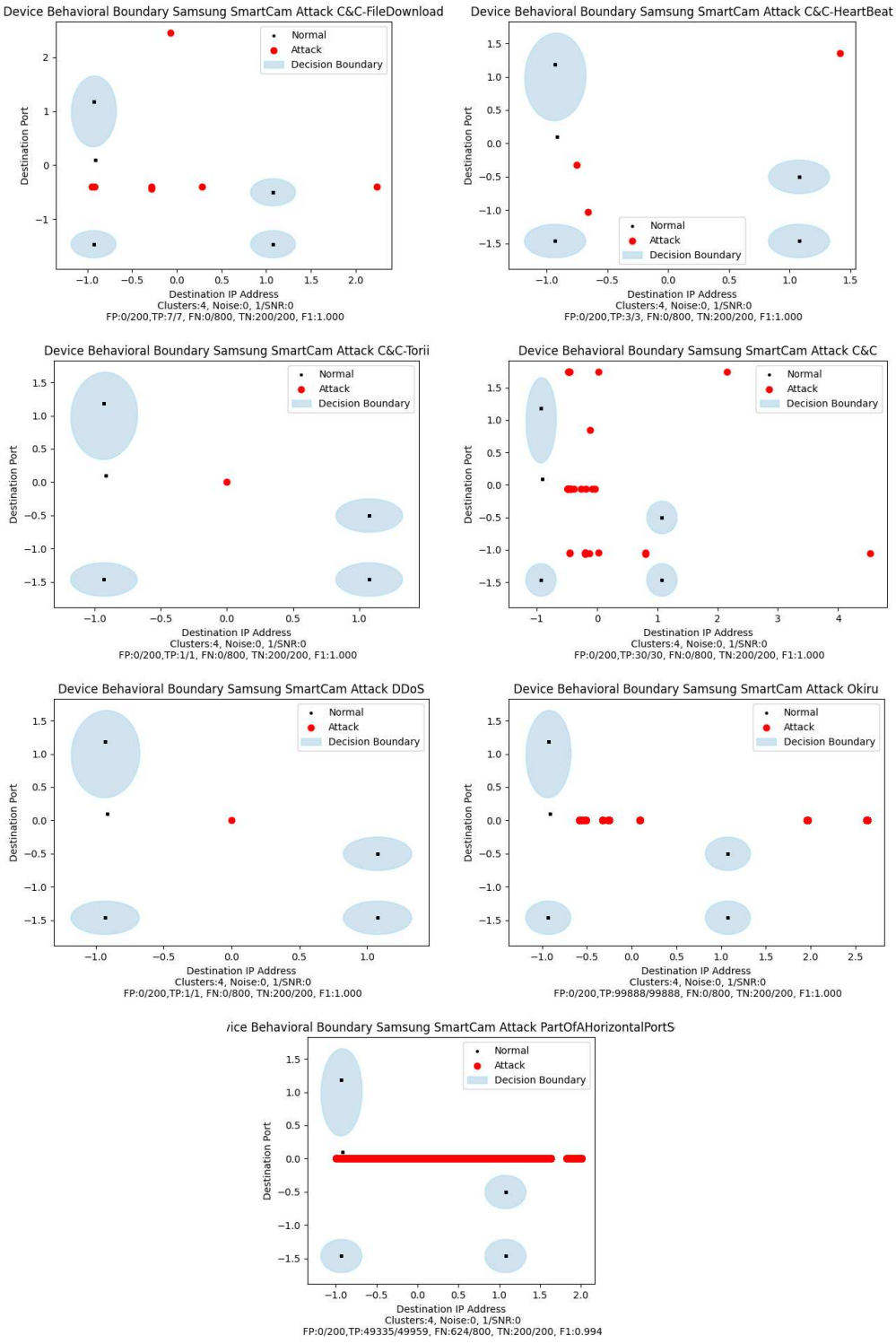


Figure D.18: Samsung-SmartCam: UNSW

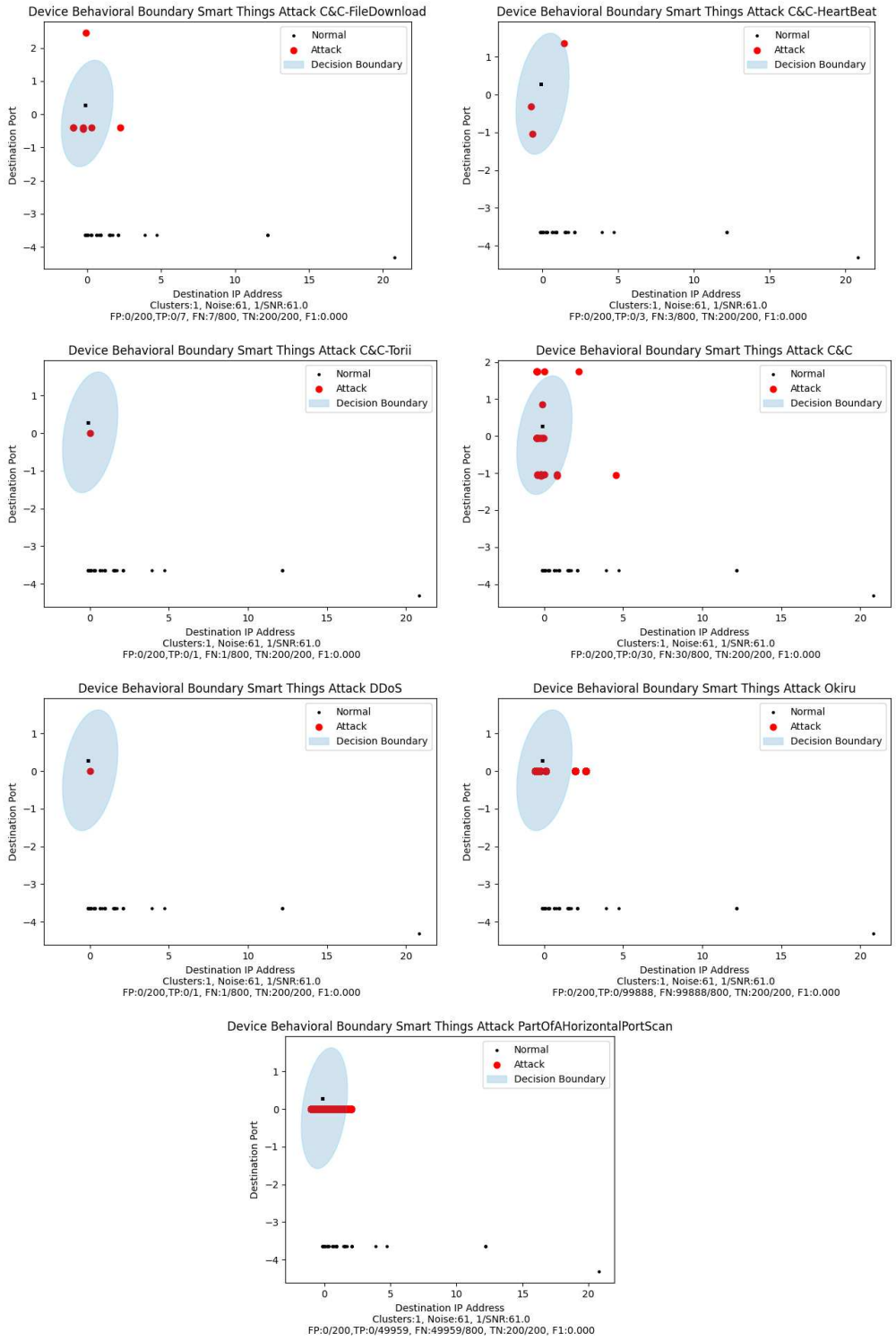


Figure D.19: Smart-Things: UNSW

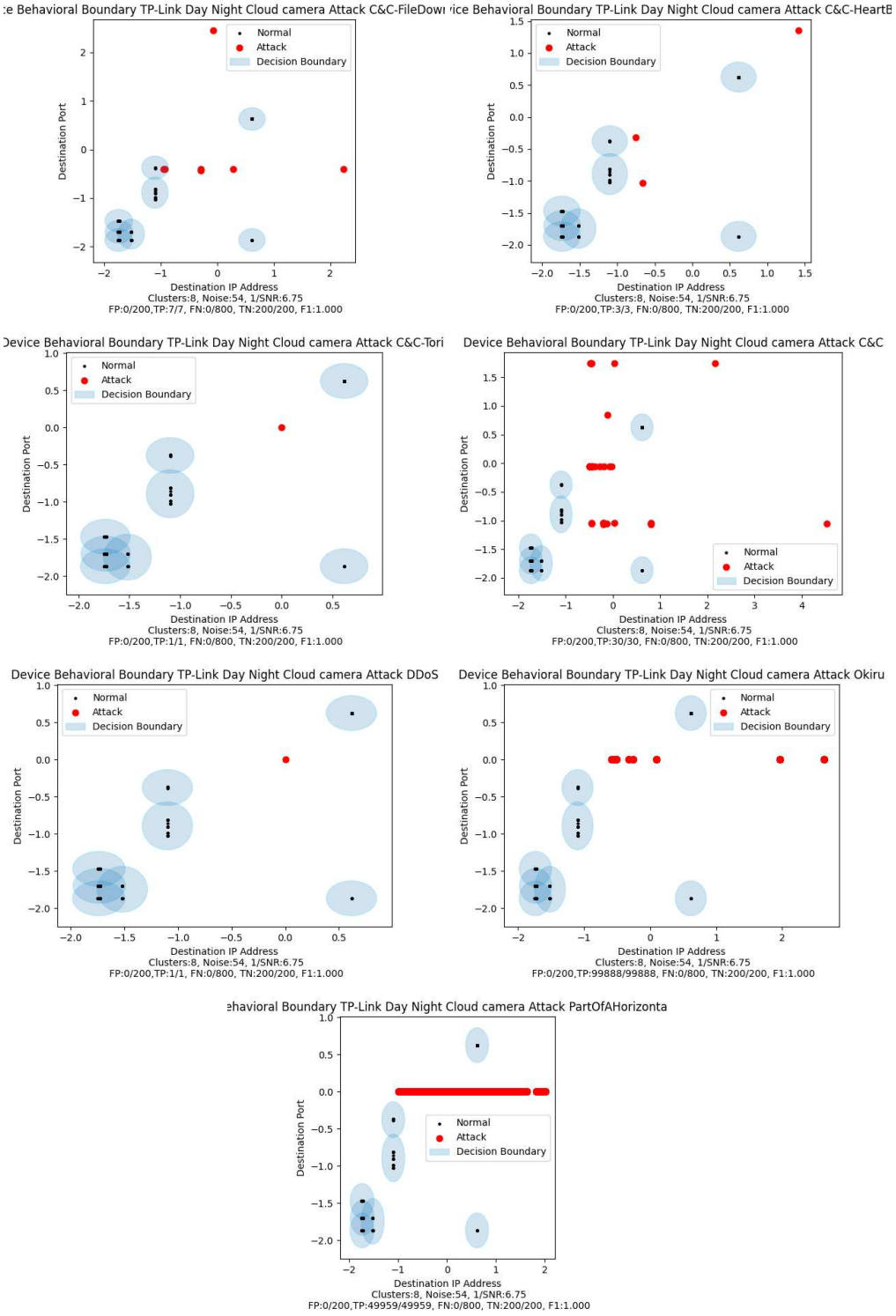


Figure D.20: TP-Link-Day-Night-Cloud-camera: UNSW

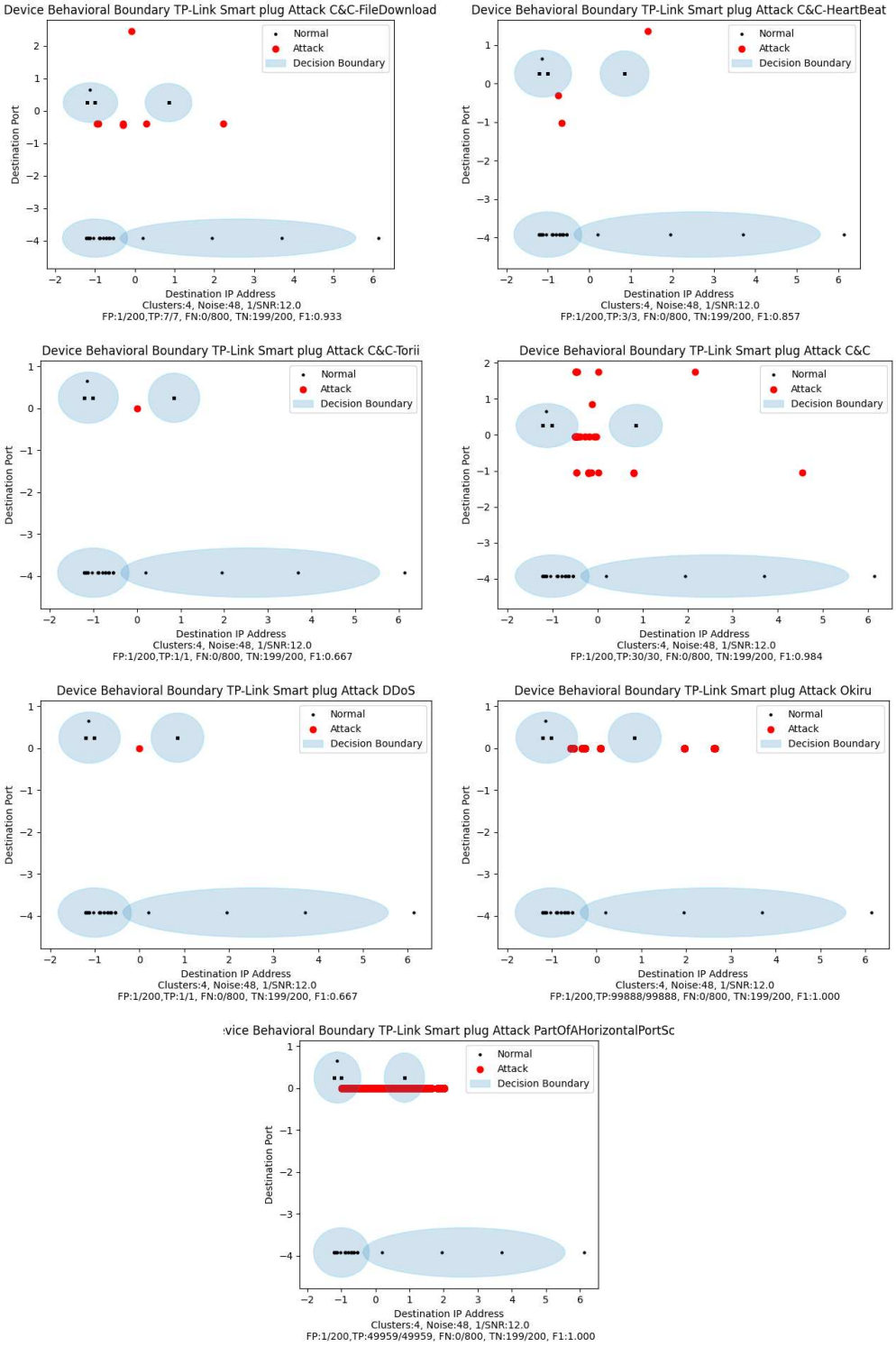


Figure D.21: TP-Link-Smart-plug: UNSW

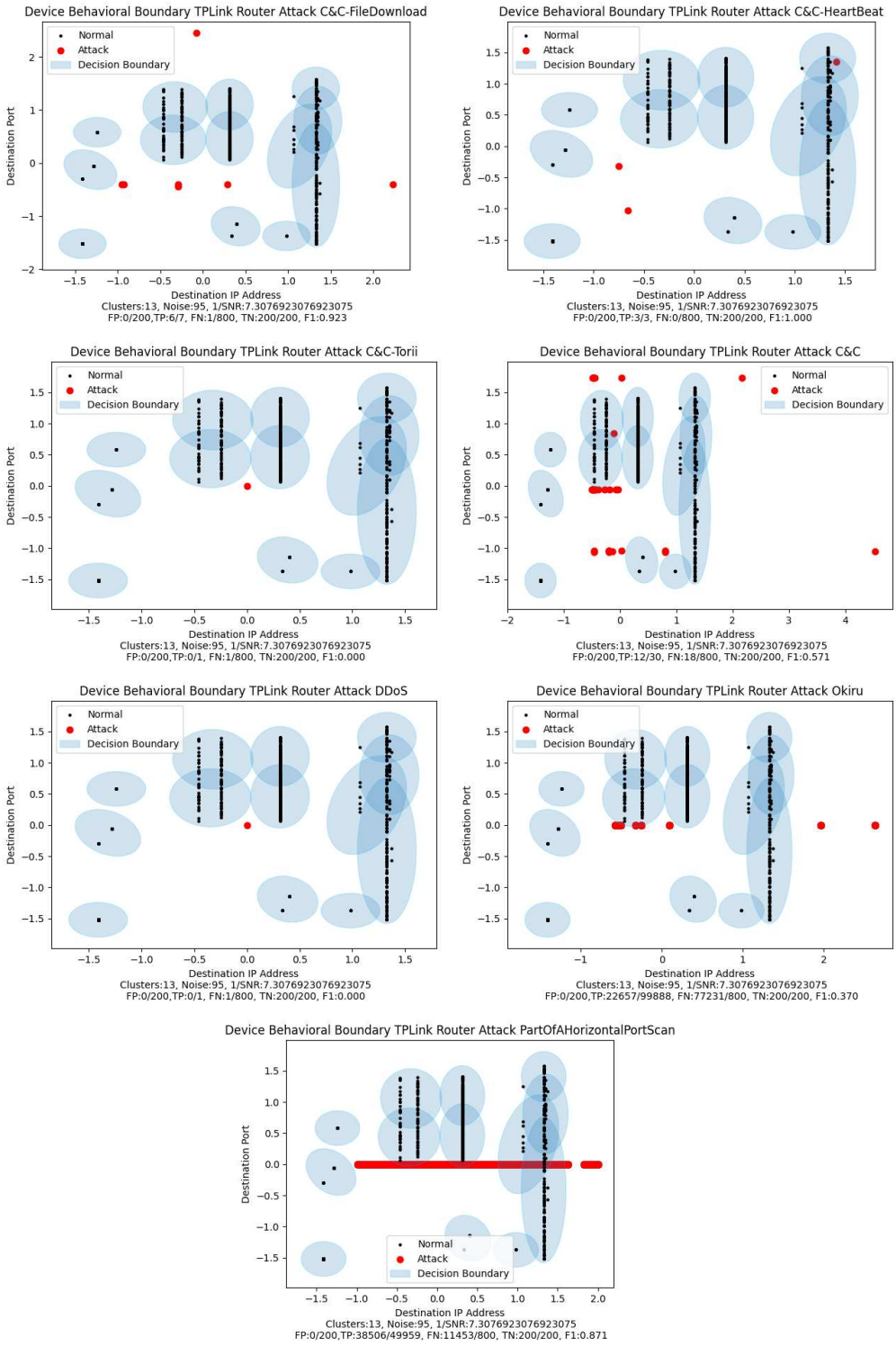


Figure D.22: TPLink-Router: UNSW

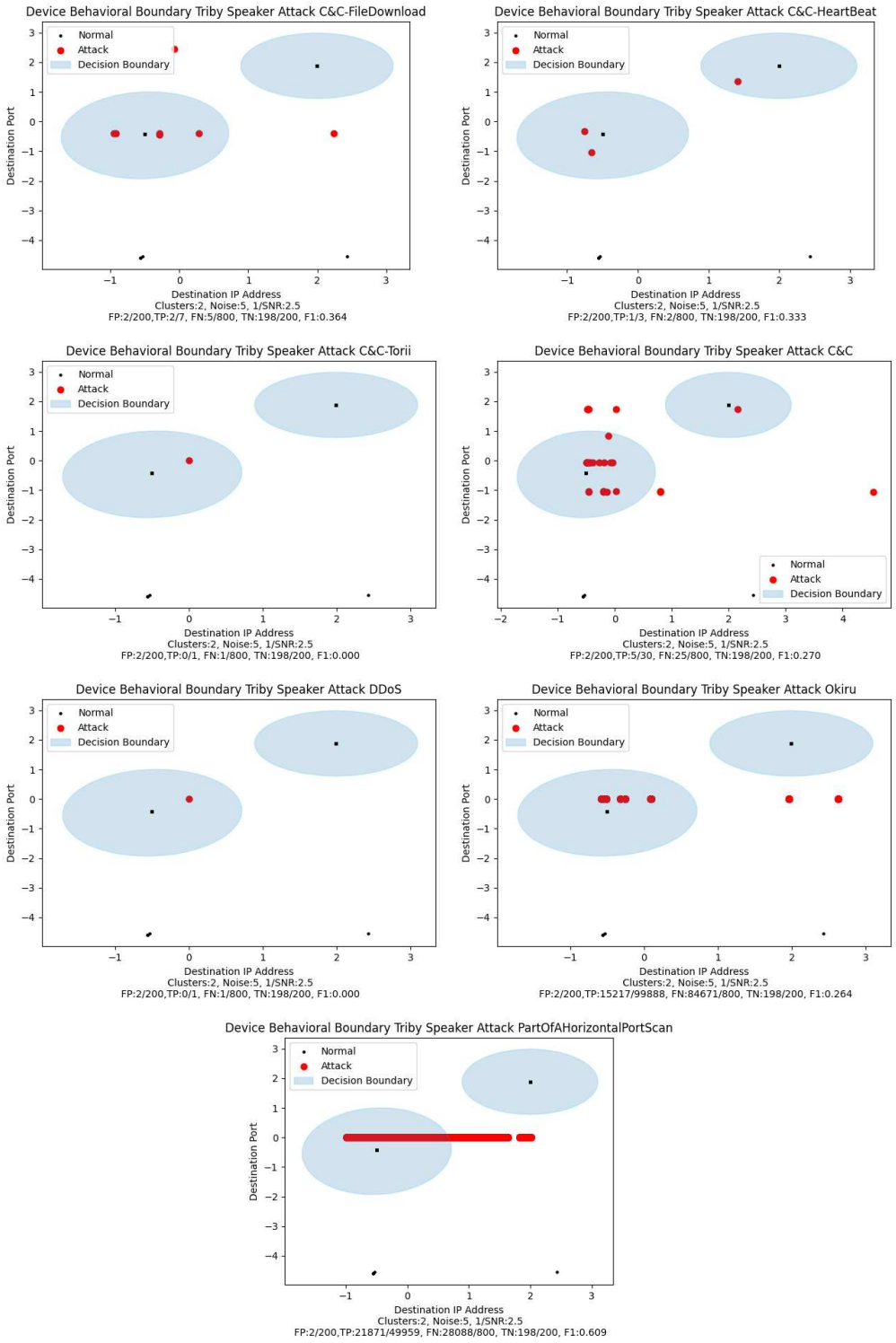


Figure D.23: Triby-Speaker: UNSW

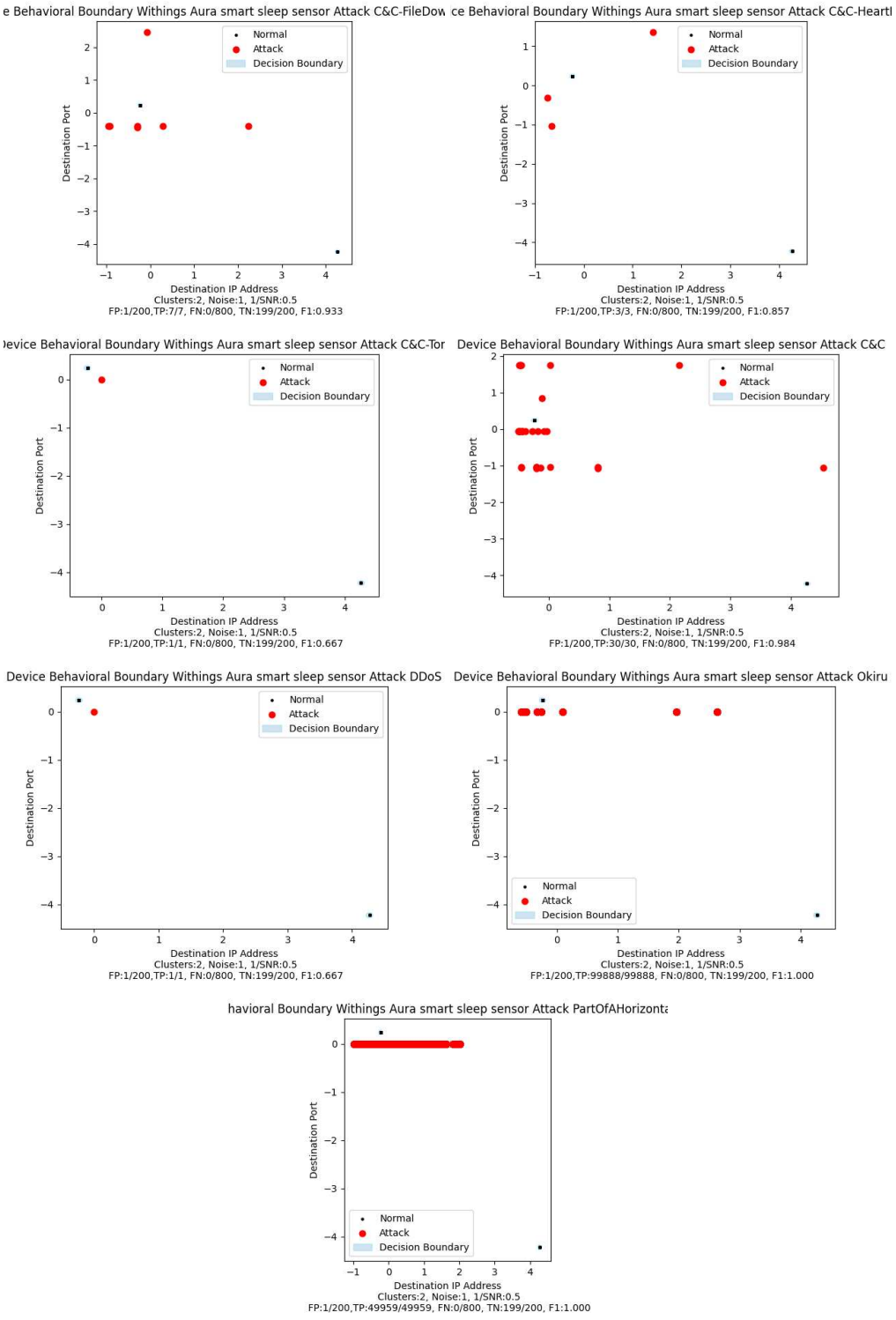


Figure D.24: Withings-Aura-smart-sleep-sensor: UNSW

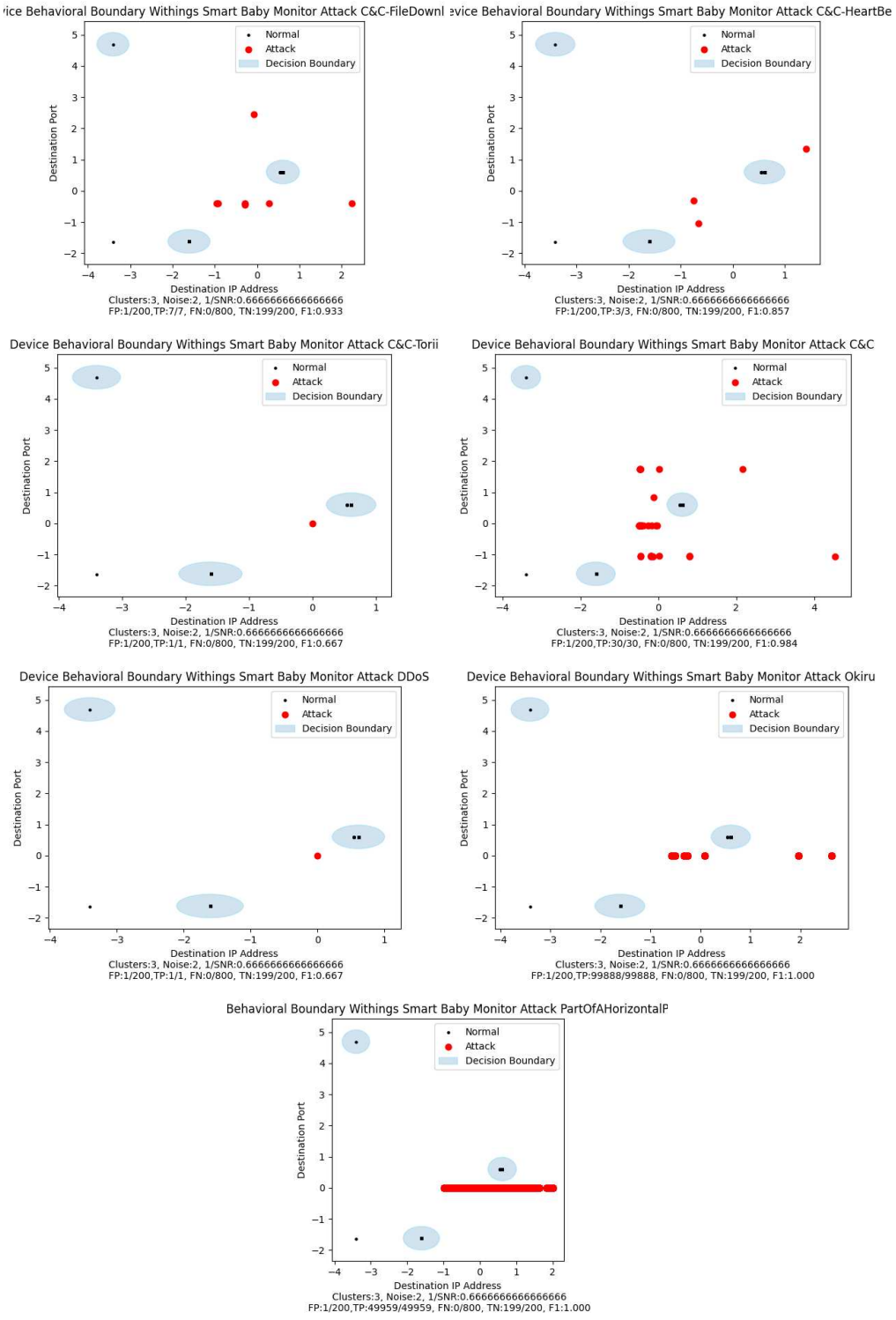


Figure D.25: Withings-Smart-Baby-Monitor: UNSW

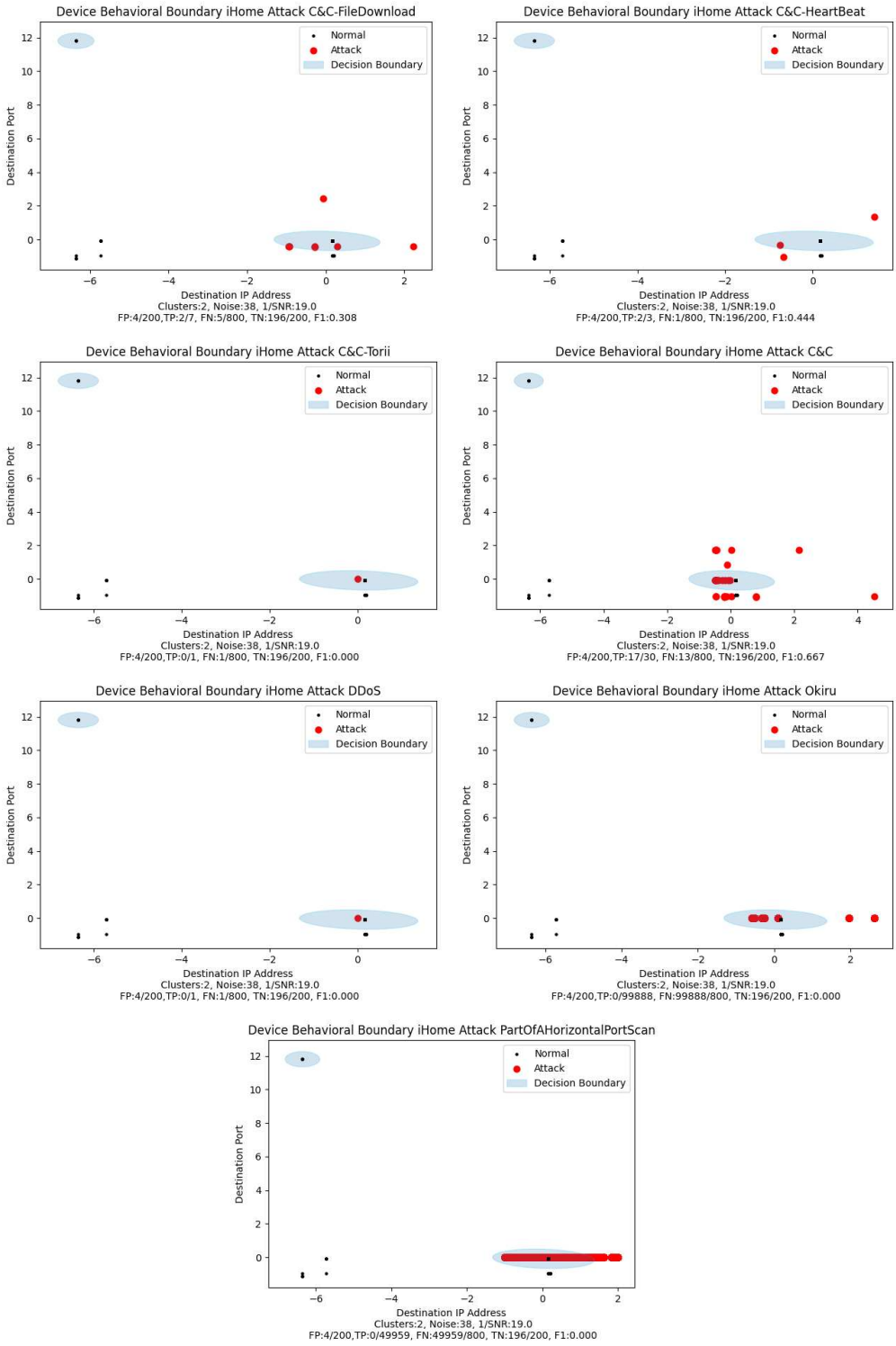


Figure D.26: iHome: UNSW

Appendix E

SCADA

This Appendix shows the Gaussian Boundary for each device in the SCADA dataset against all seven attacks listed in 3.6.

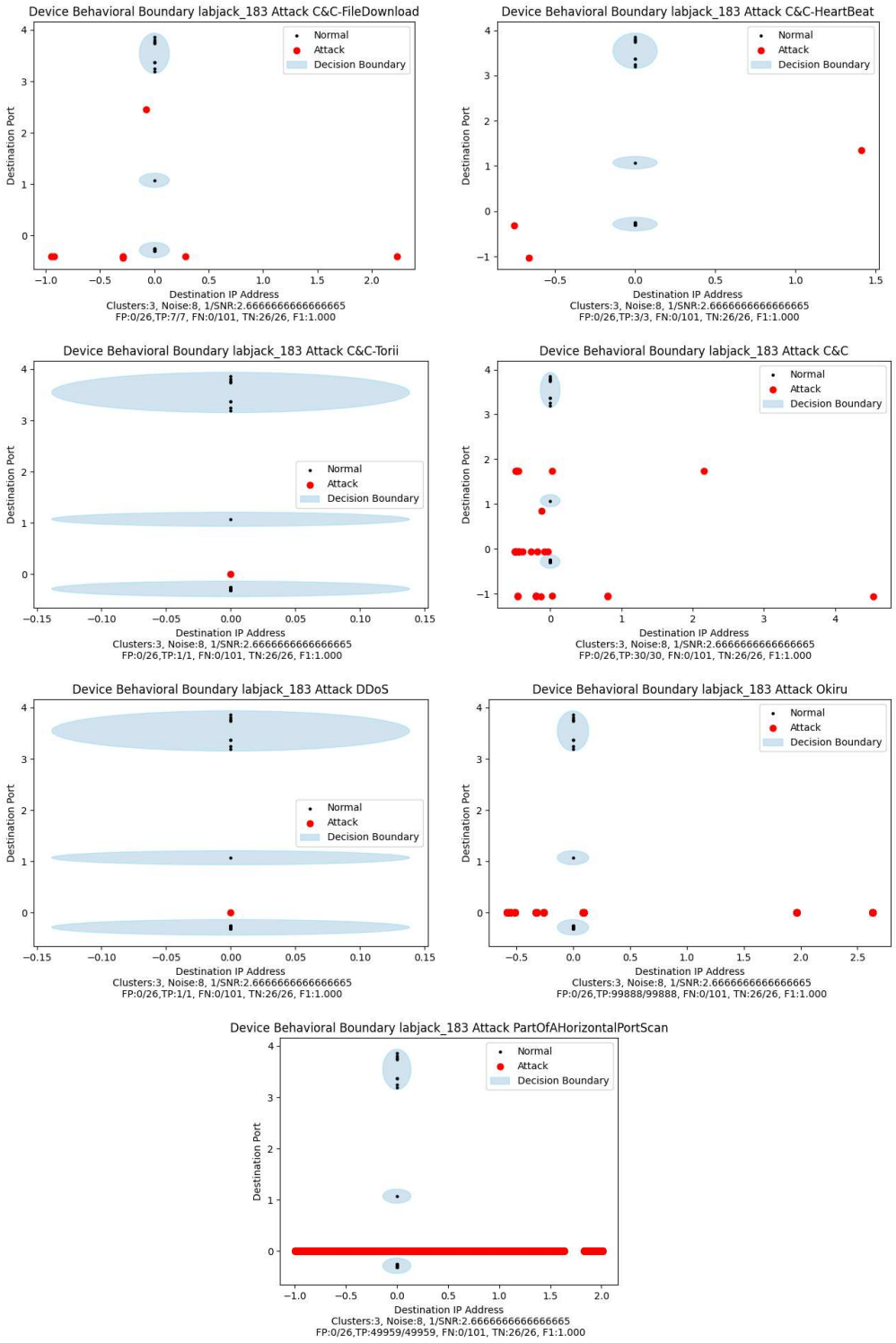


Figure E.1: labjack-183: SCADA

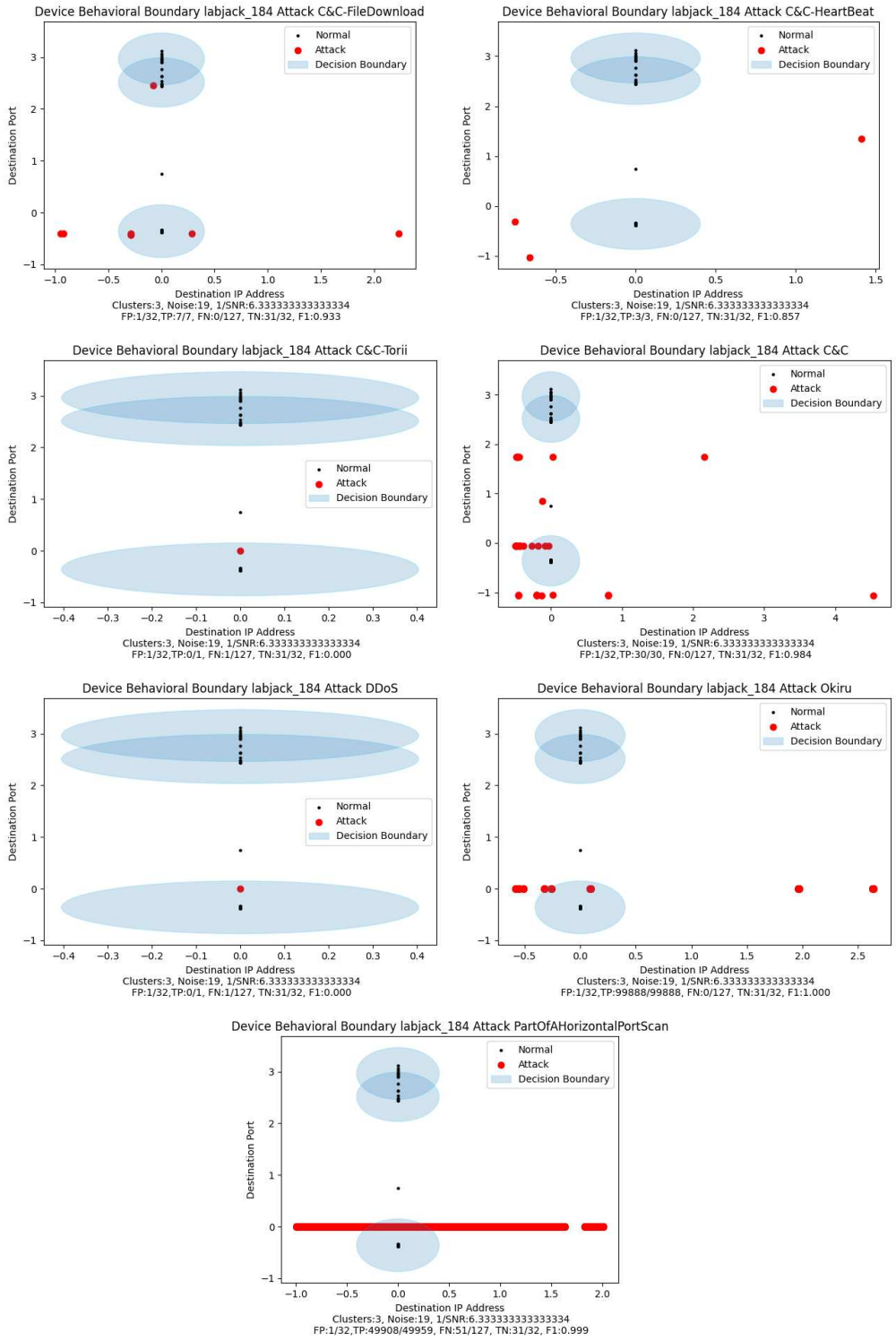


Figure E.2: labjack-184: SCADA

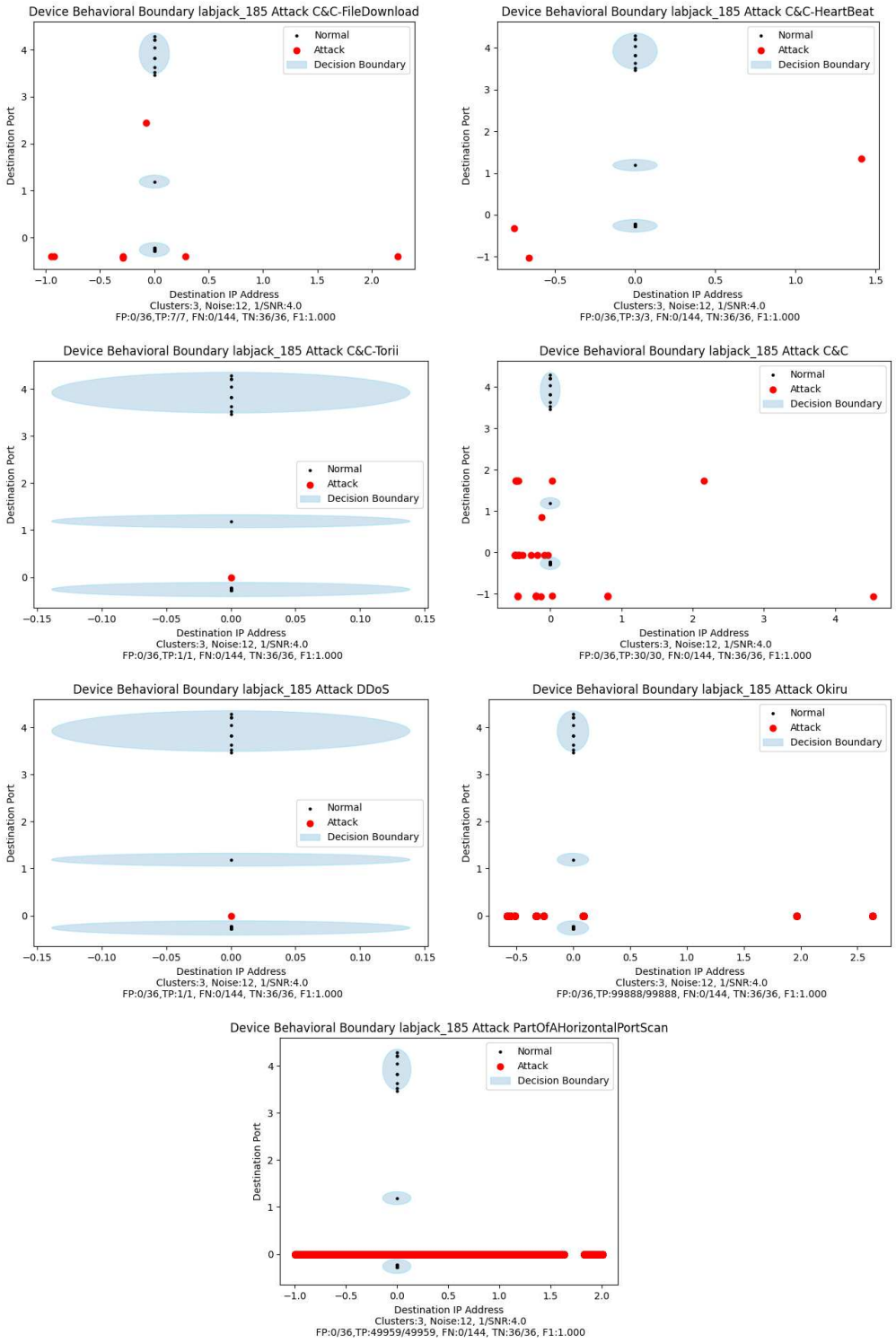


Figure E.3: labjack-185: SCADA

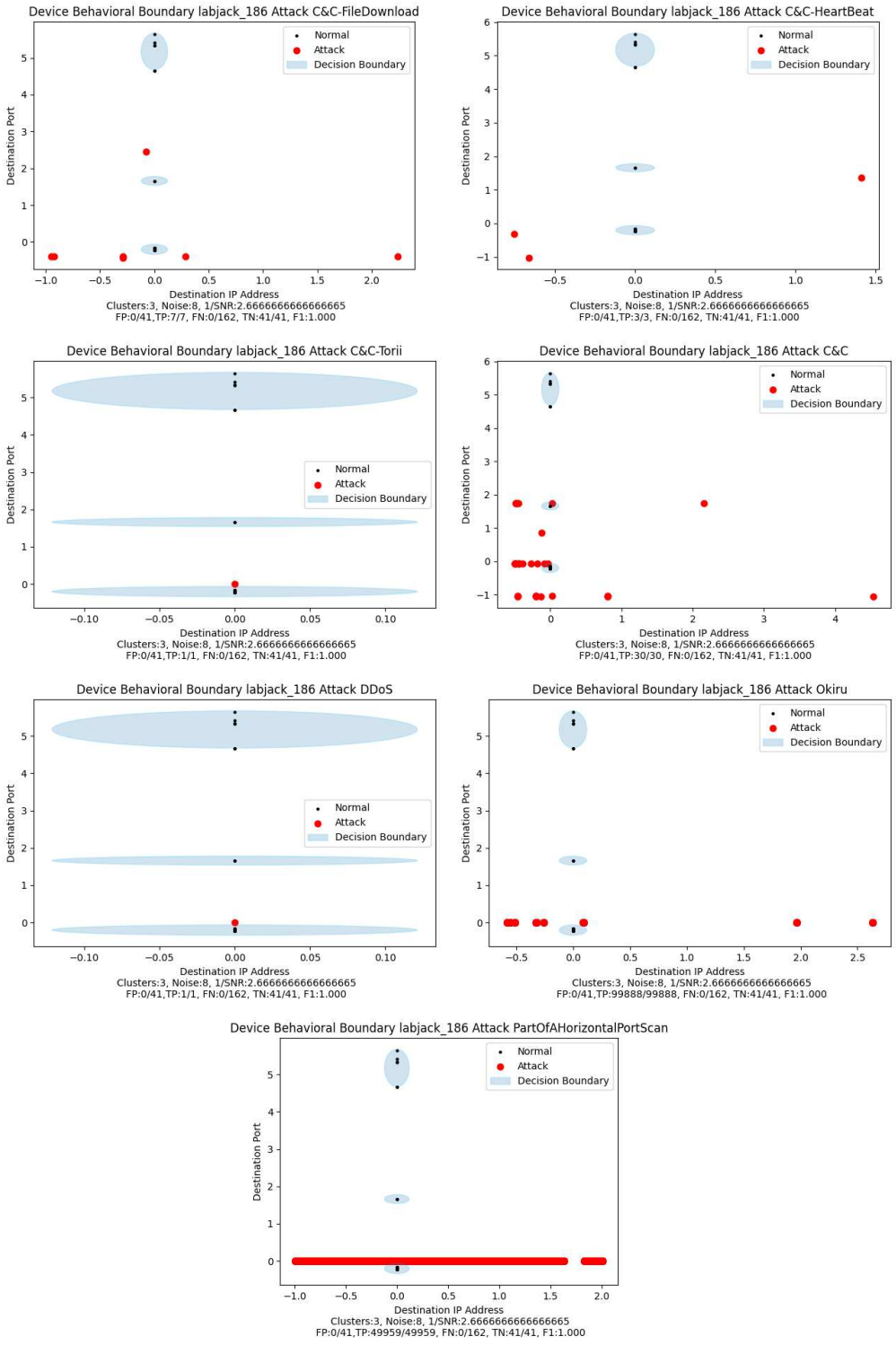


Figure E.4: labjack-186: SCADA

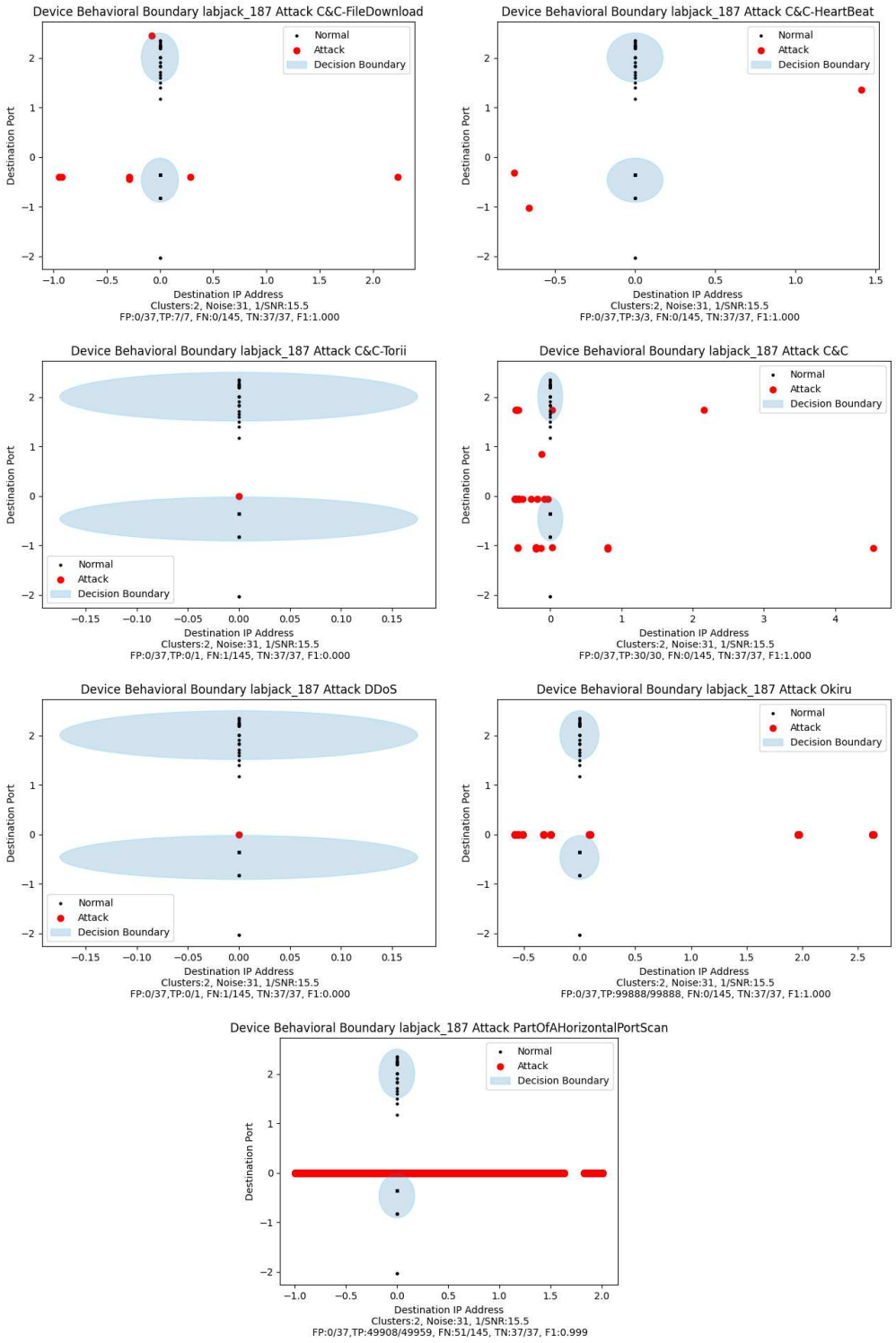


Figure E.5: labjack-187: SCADA

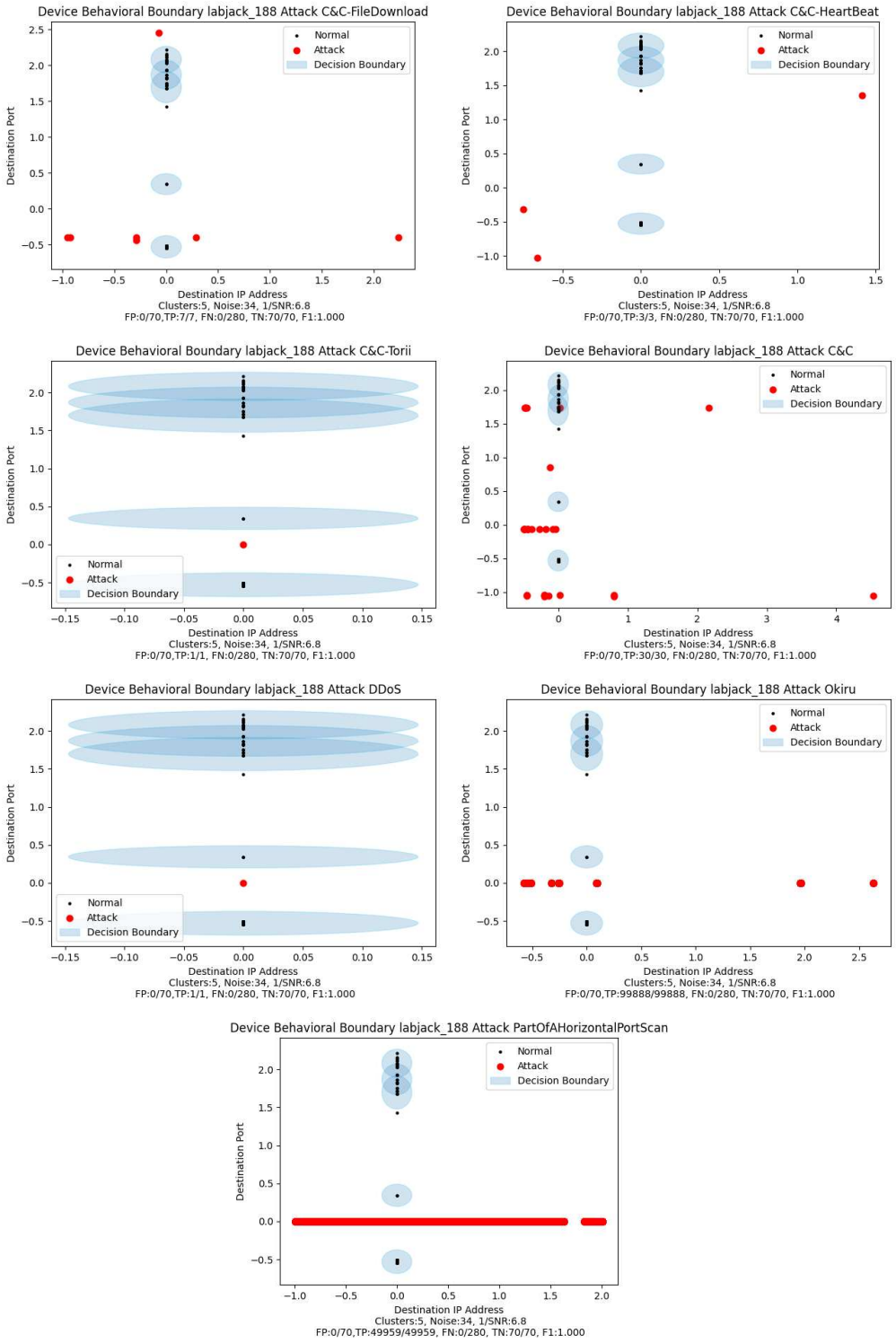


Figure E.6: labjack-188: SCADA

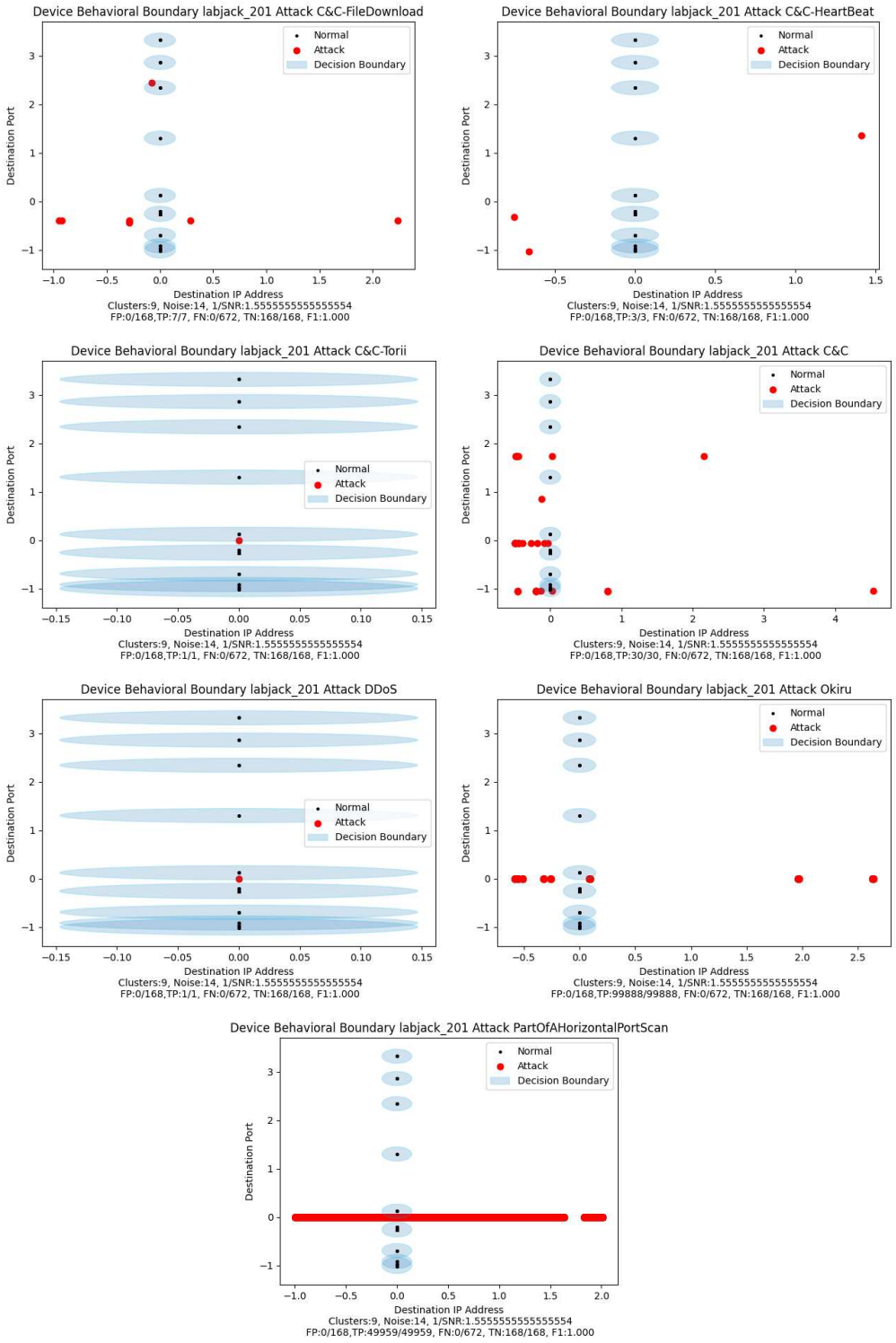


Figure E.7: labjack-201: SCADA

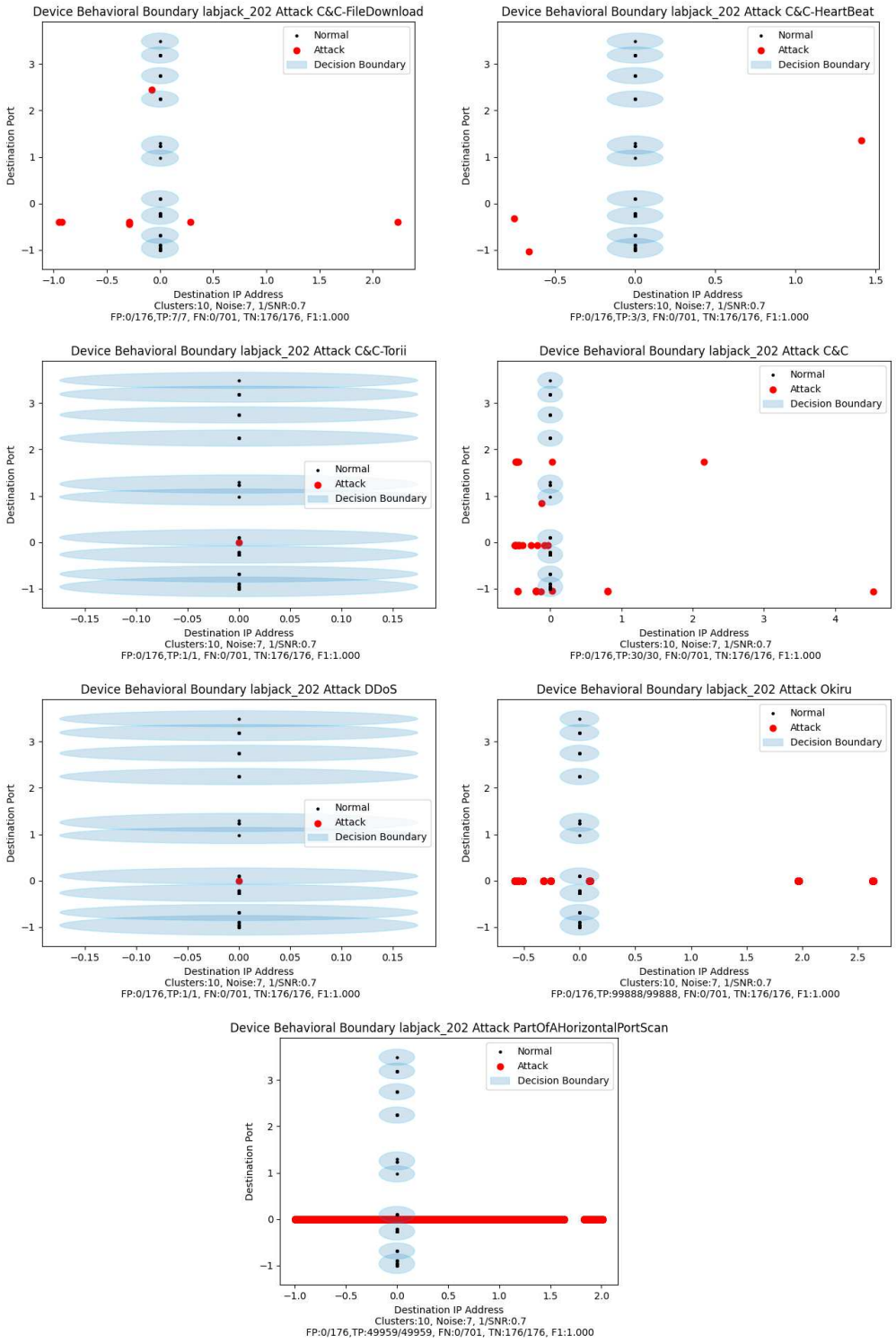


Figure E.8: labjack-202: SCADA

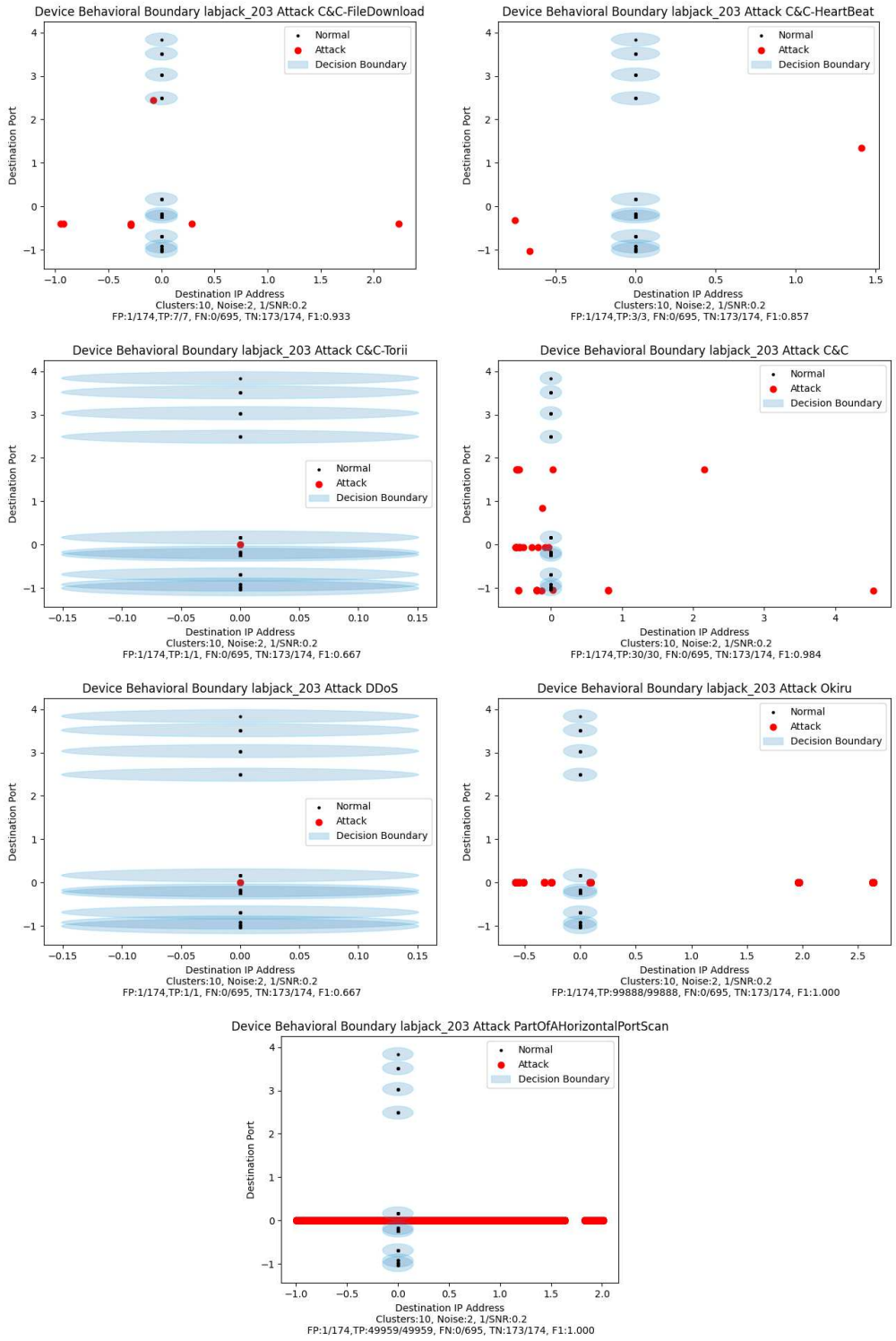


Figure E.9: labjack-203: SCADA

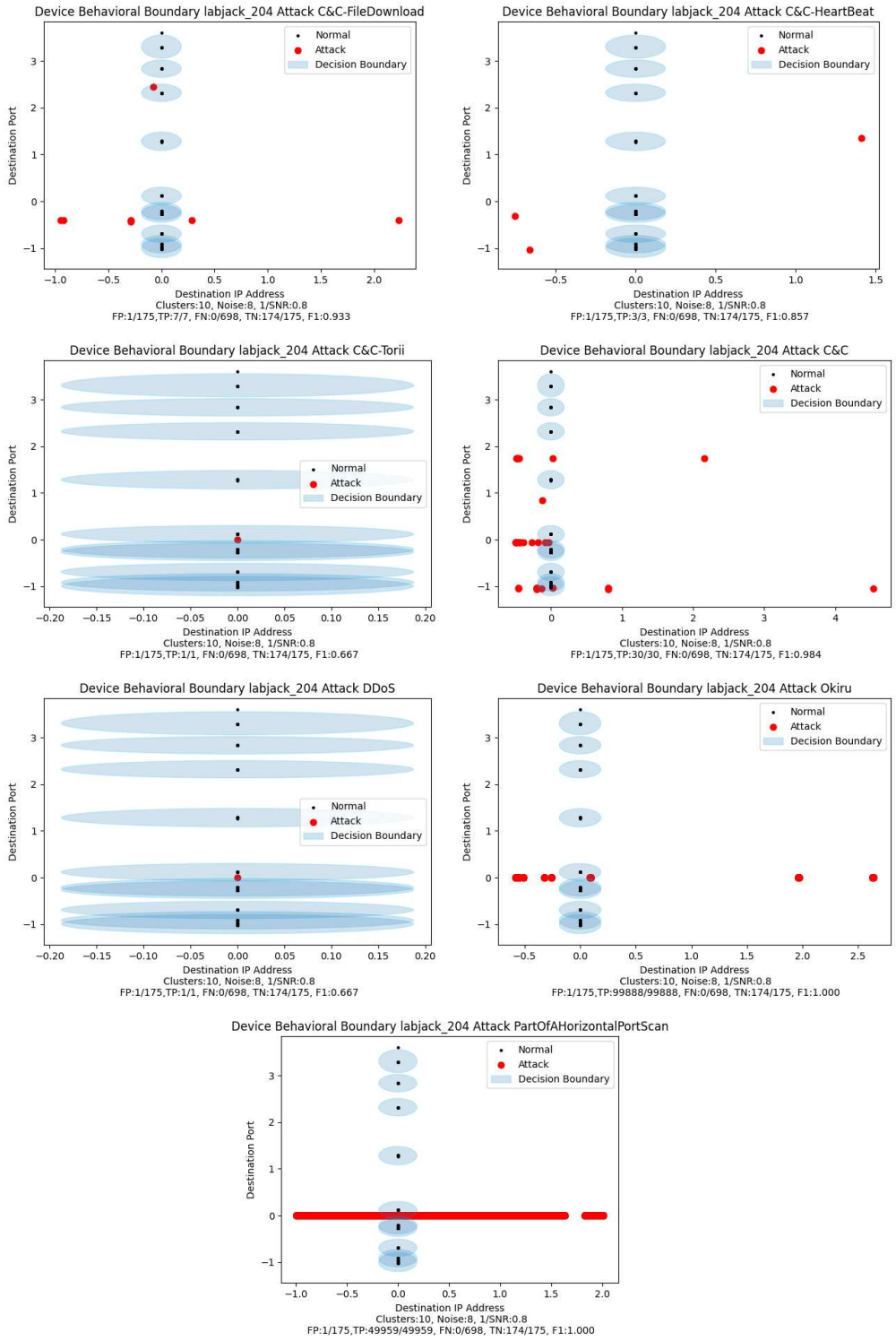


Figure E.10: labjack-204: SCADA

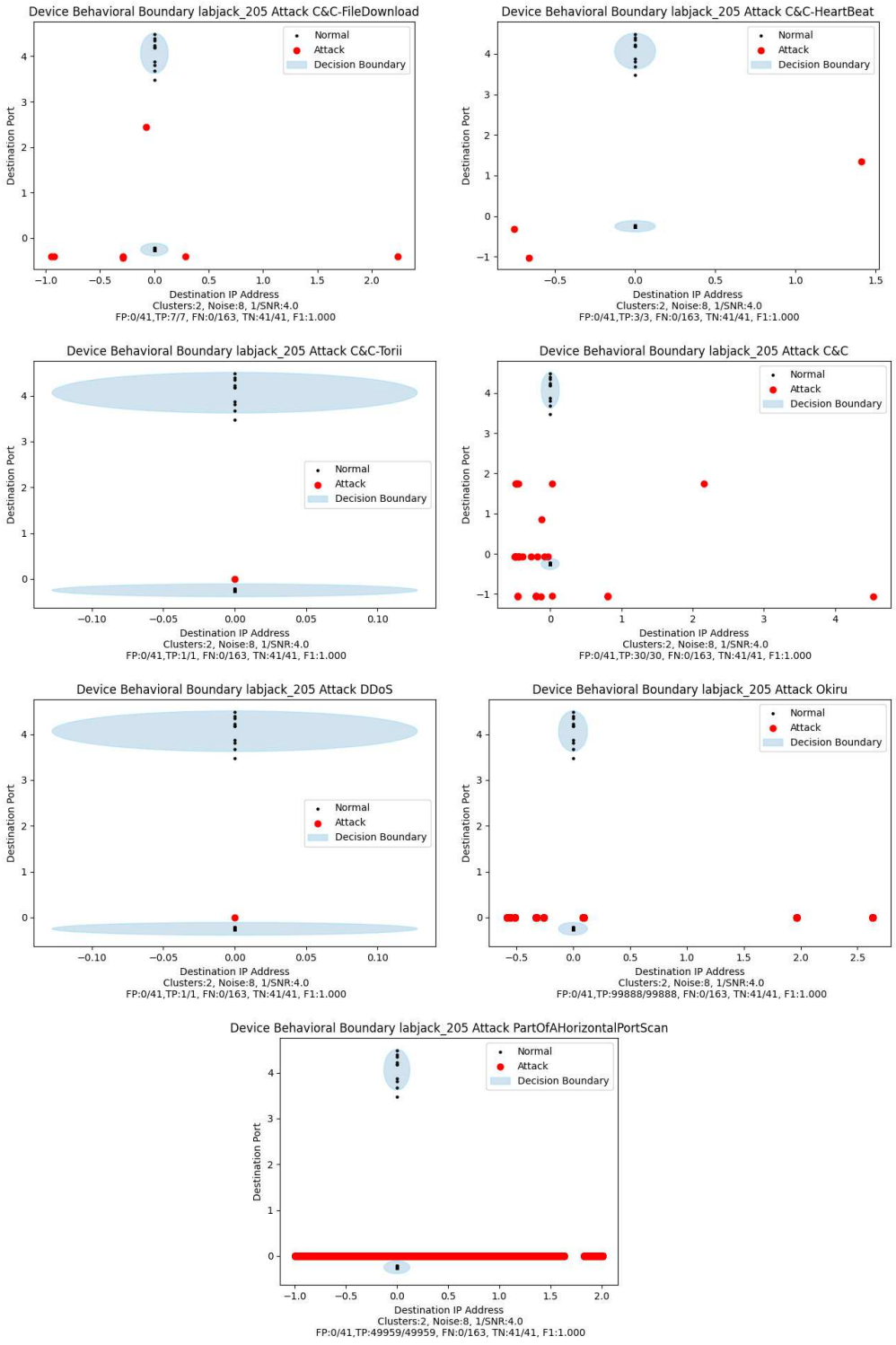


Figure E.11: labjack-205: SCADA

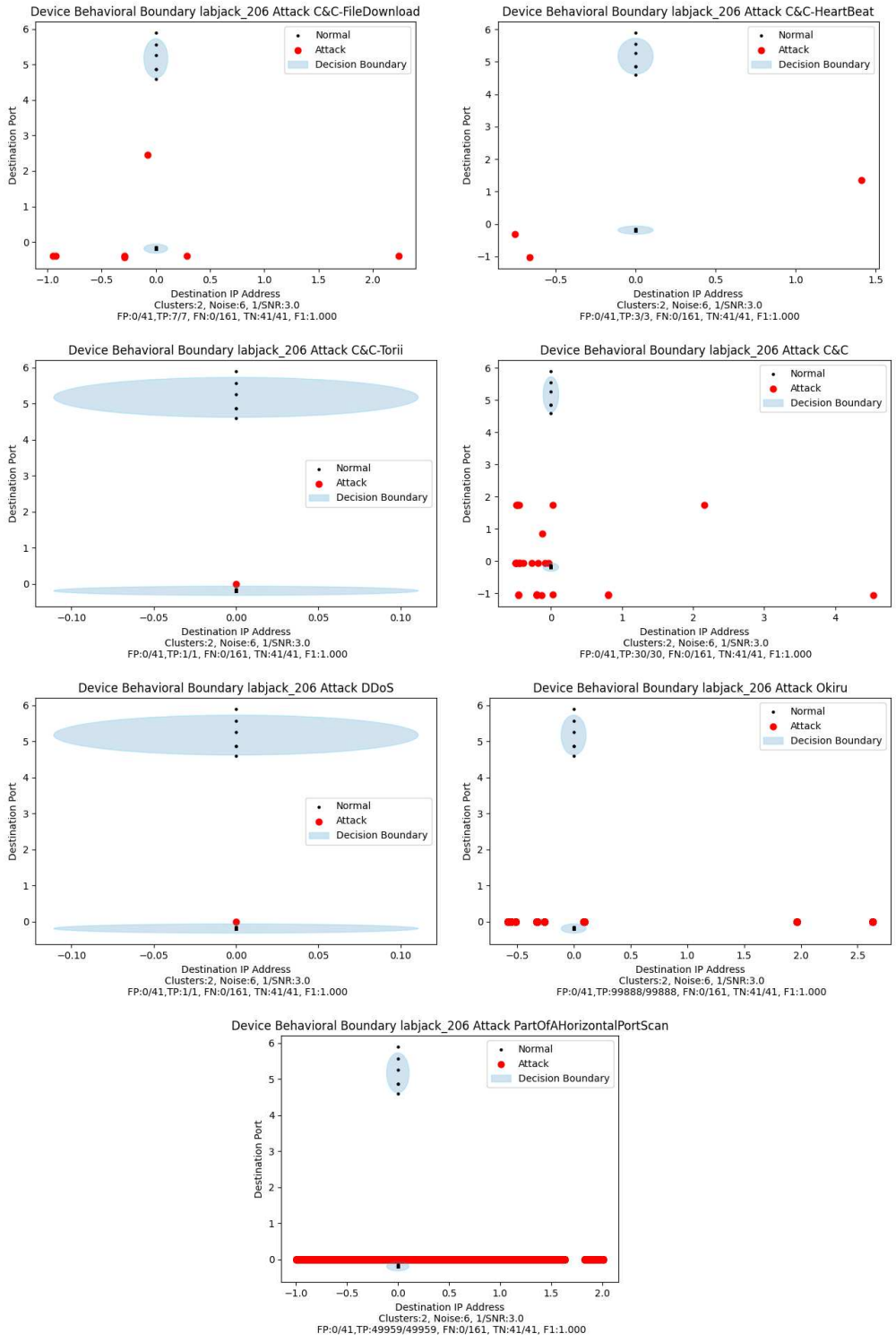


Figure E.12: labjack-206: SCADA

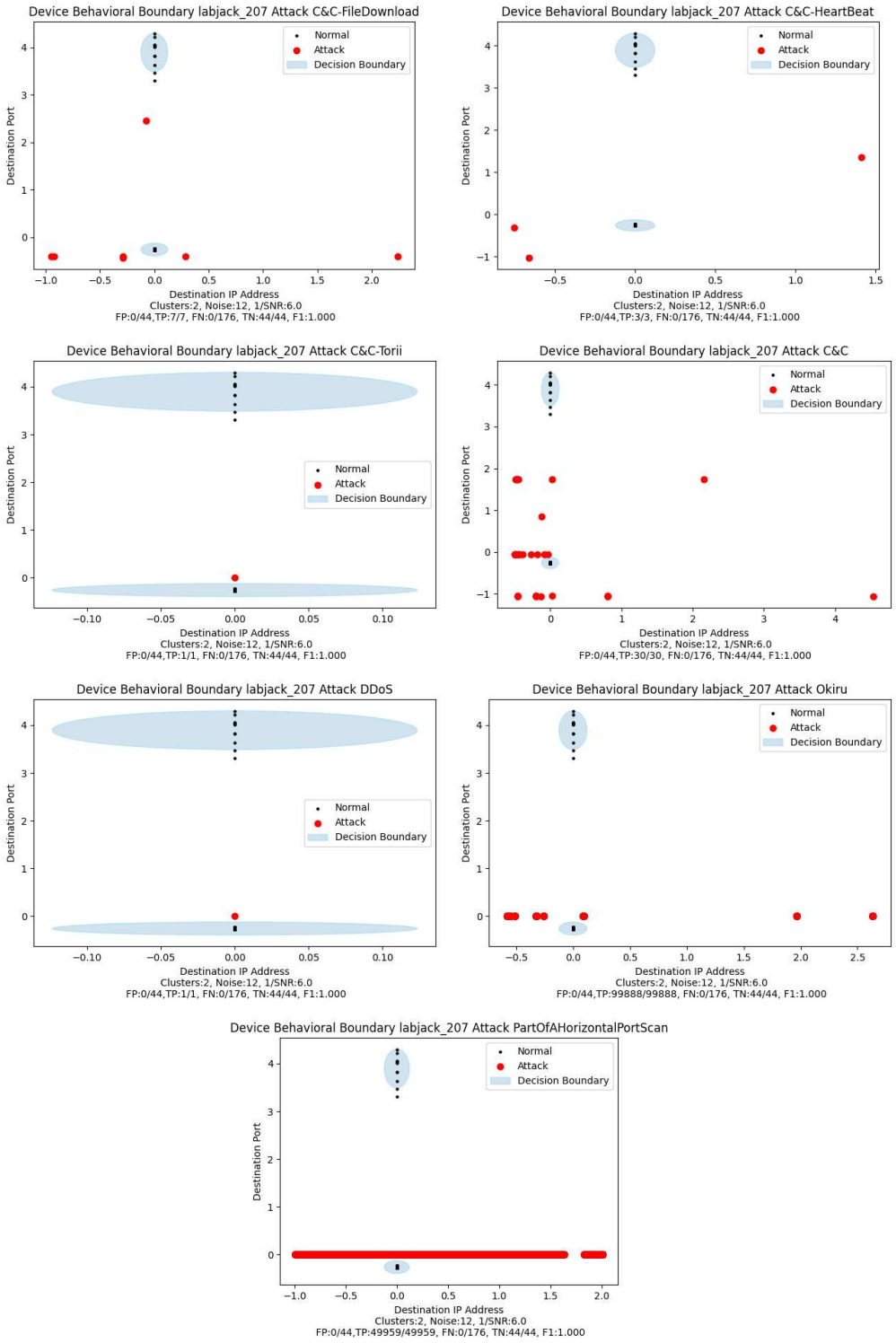


Figure E.13: labjack-207: SCADA

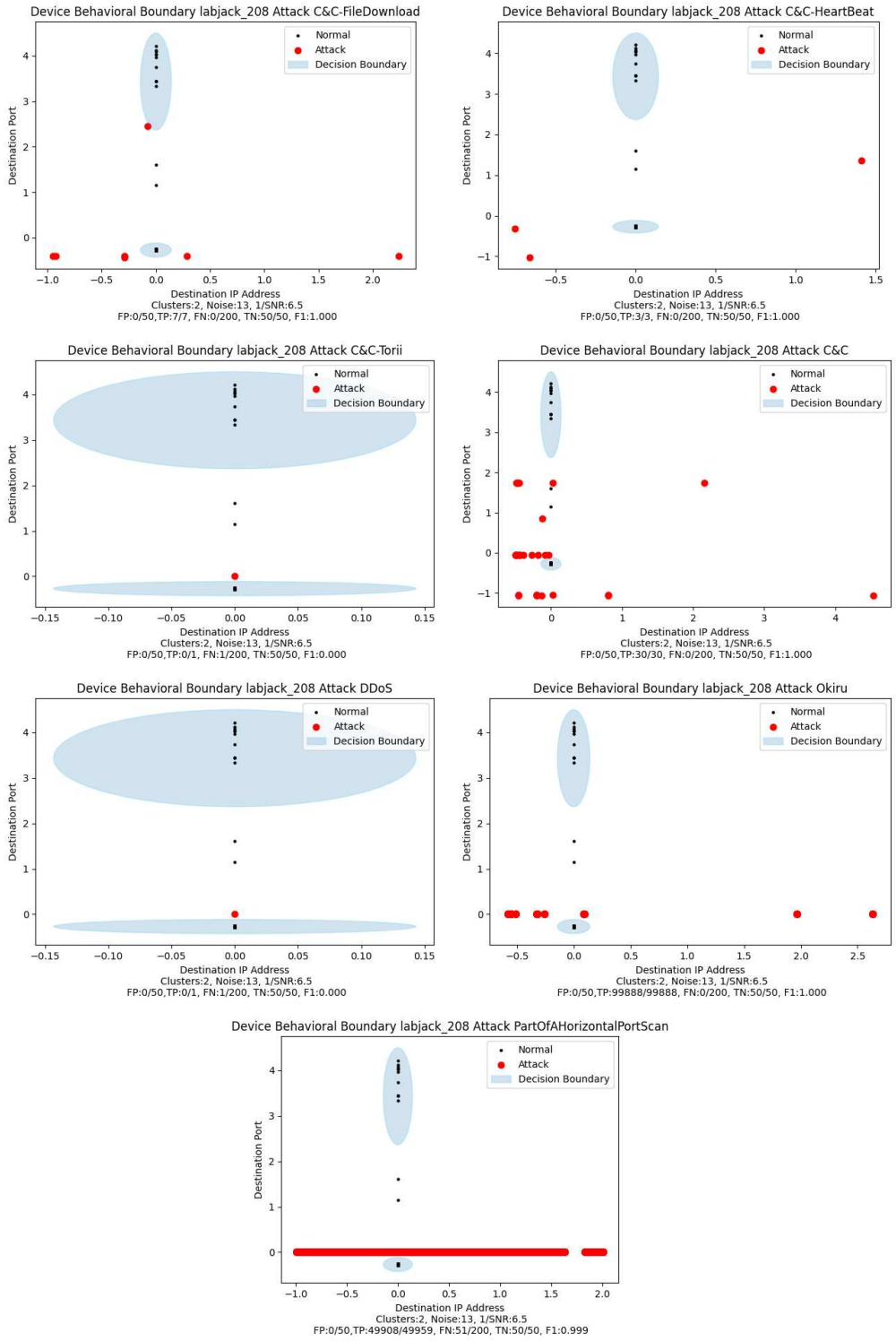


Figure E.14: labjack-208: SCADA

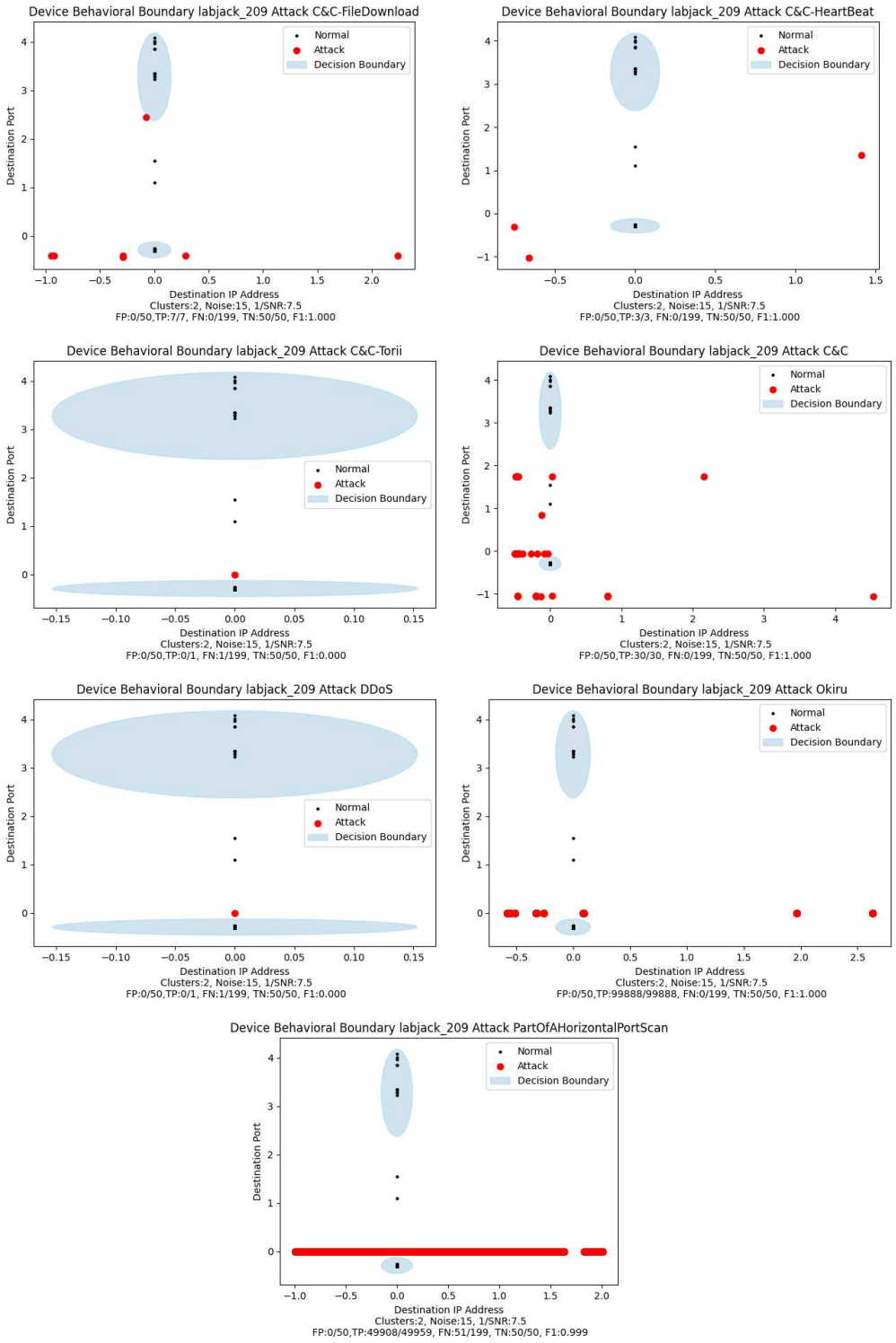


Figure E.15: labjack-209: SCADA

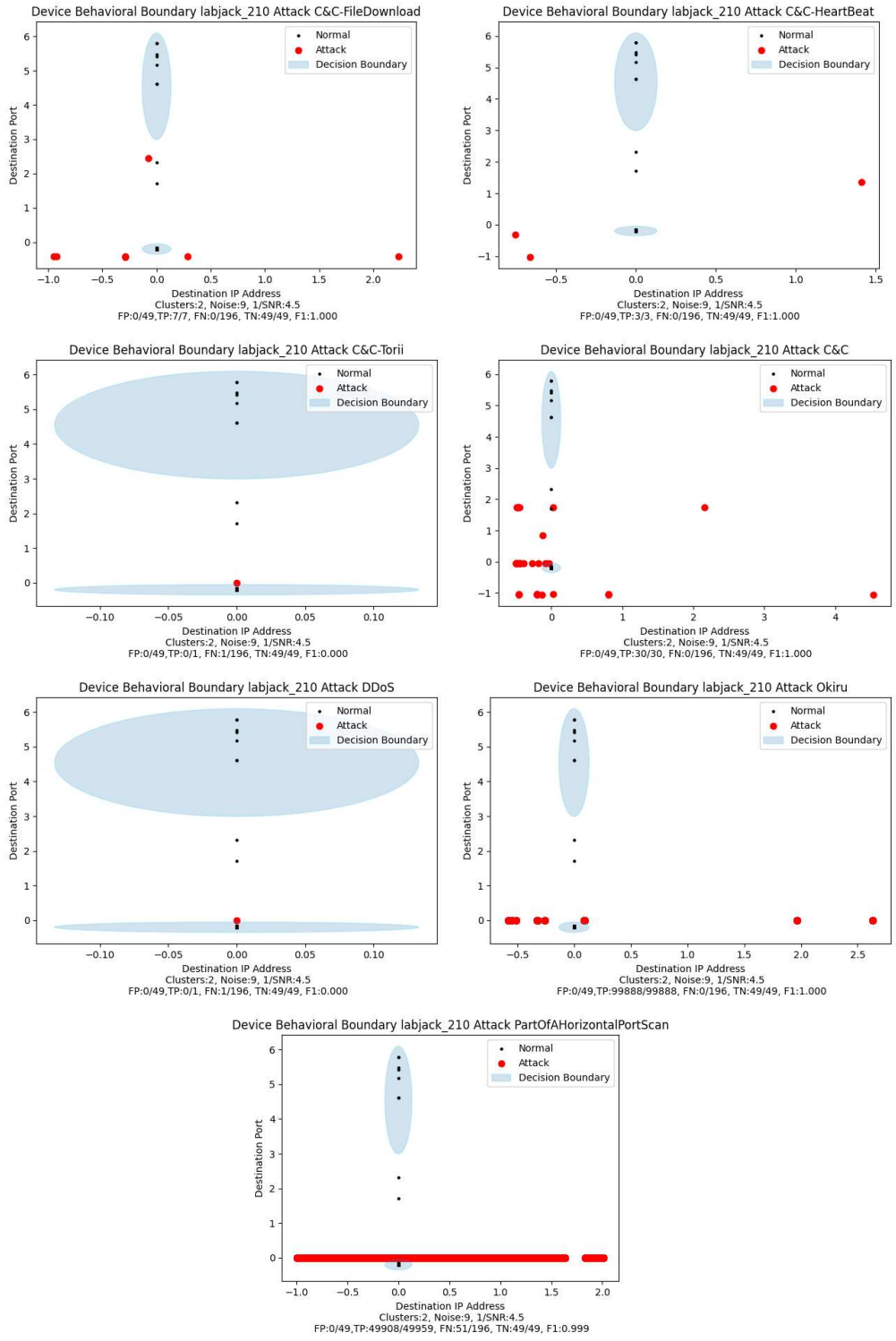


Figure E.16: labjack-210: SCADA

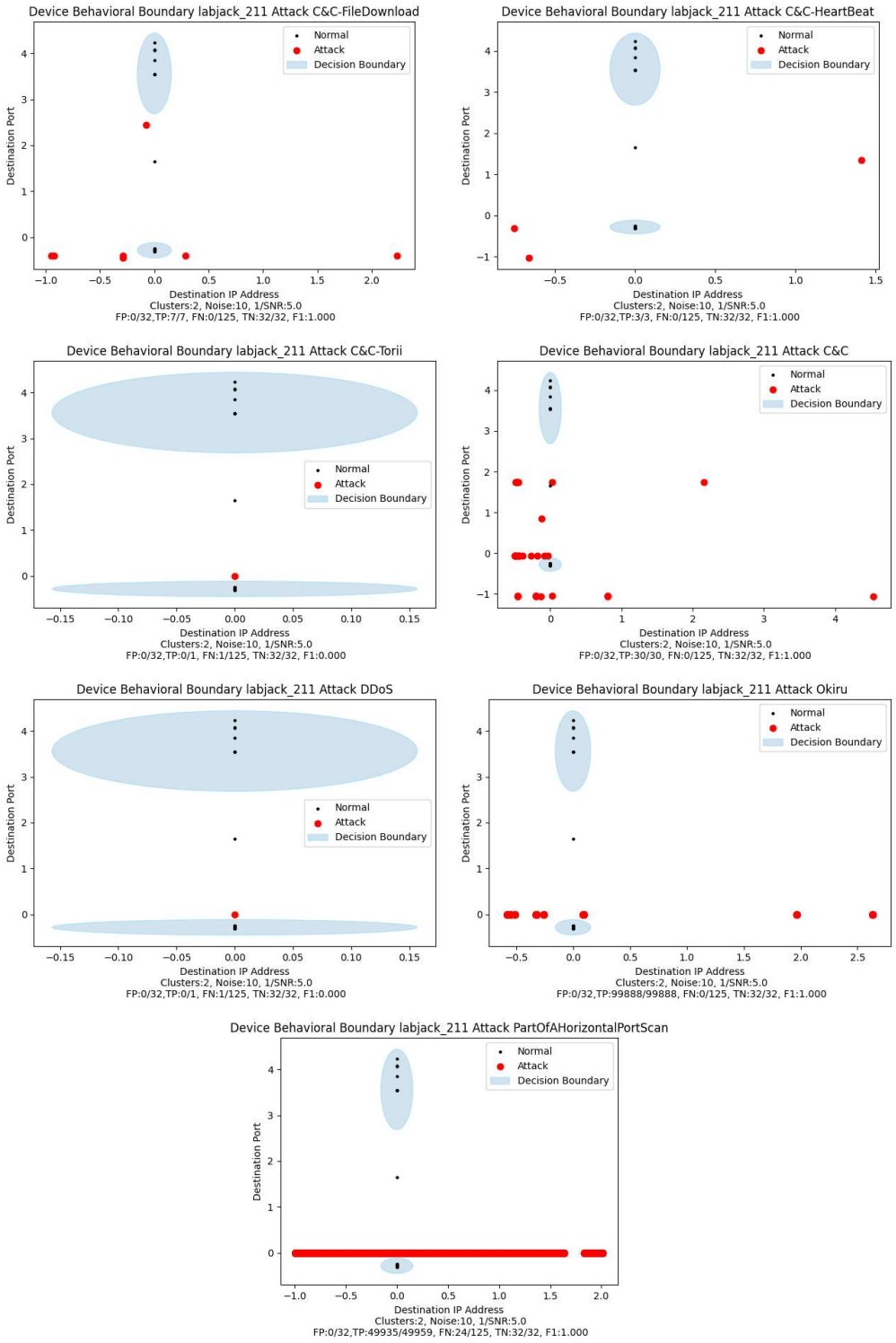


Figure E.17: labjack-211: SCADA

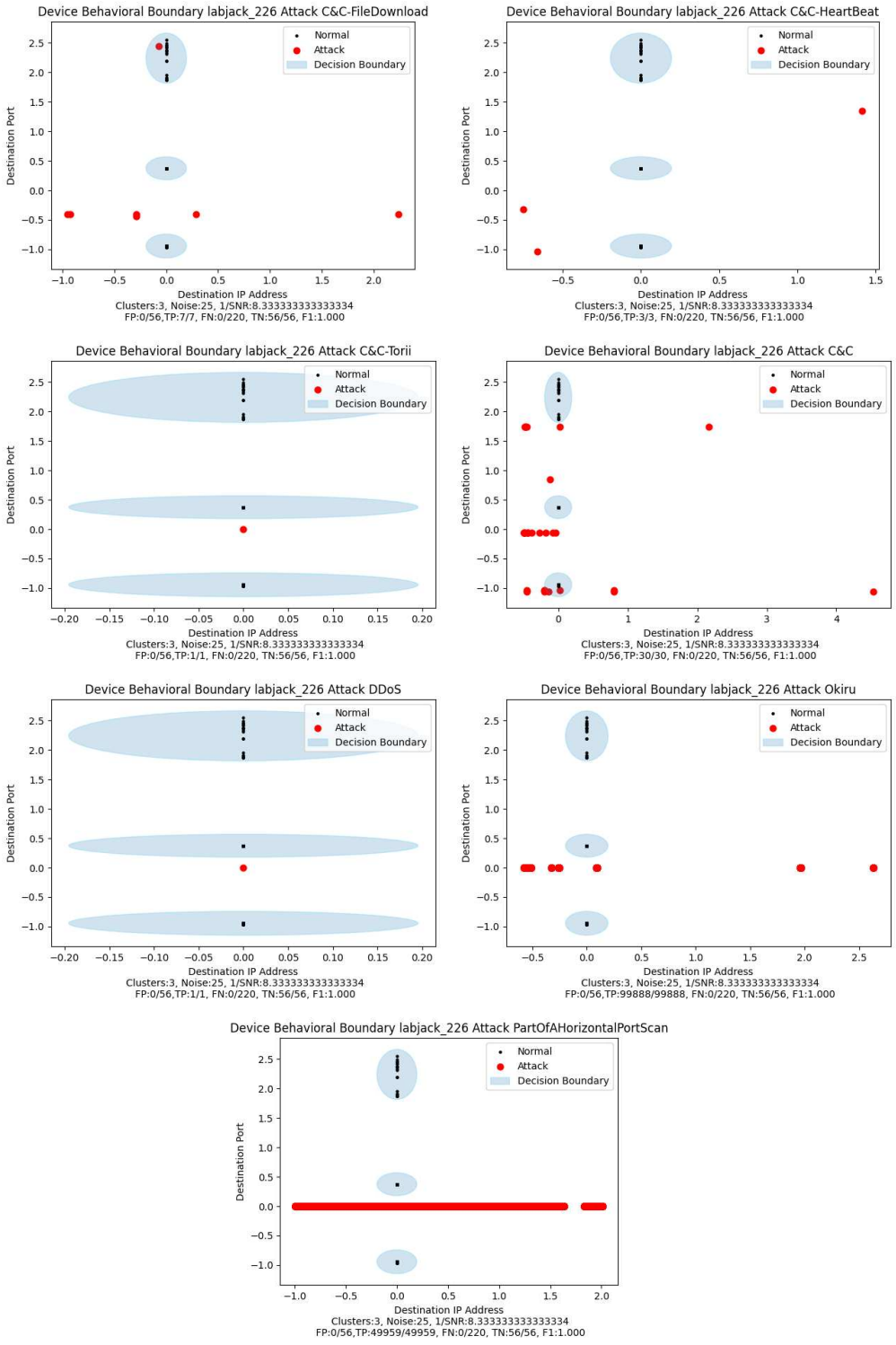


Figure E.18: labjack-226: SCADA

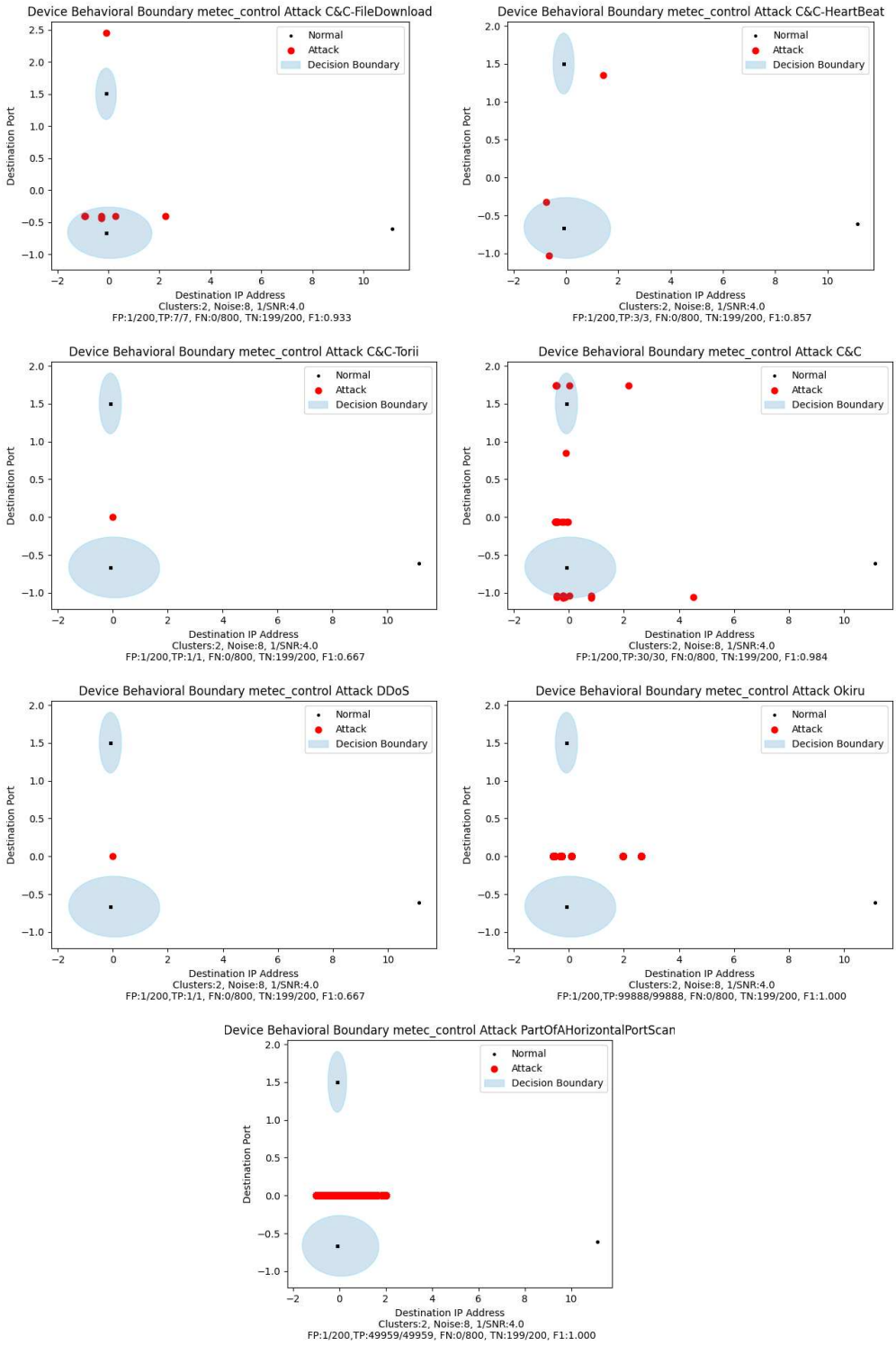


Figure E.19: metec-control: SCADA

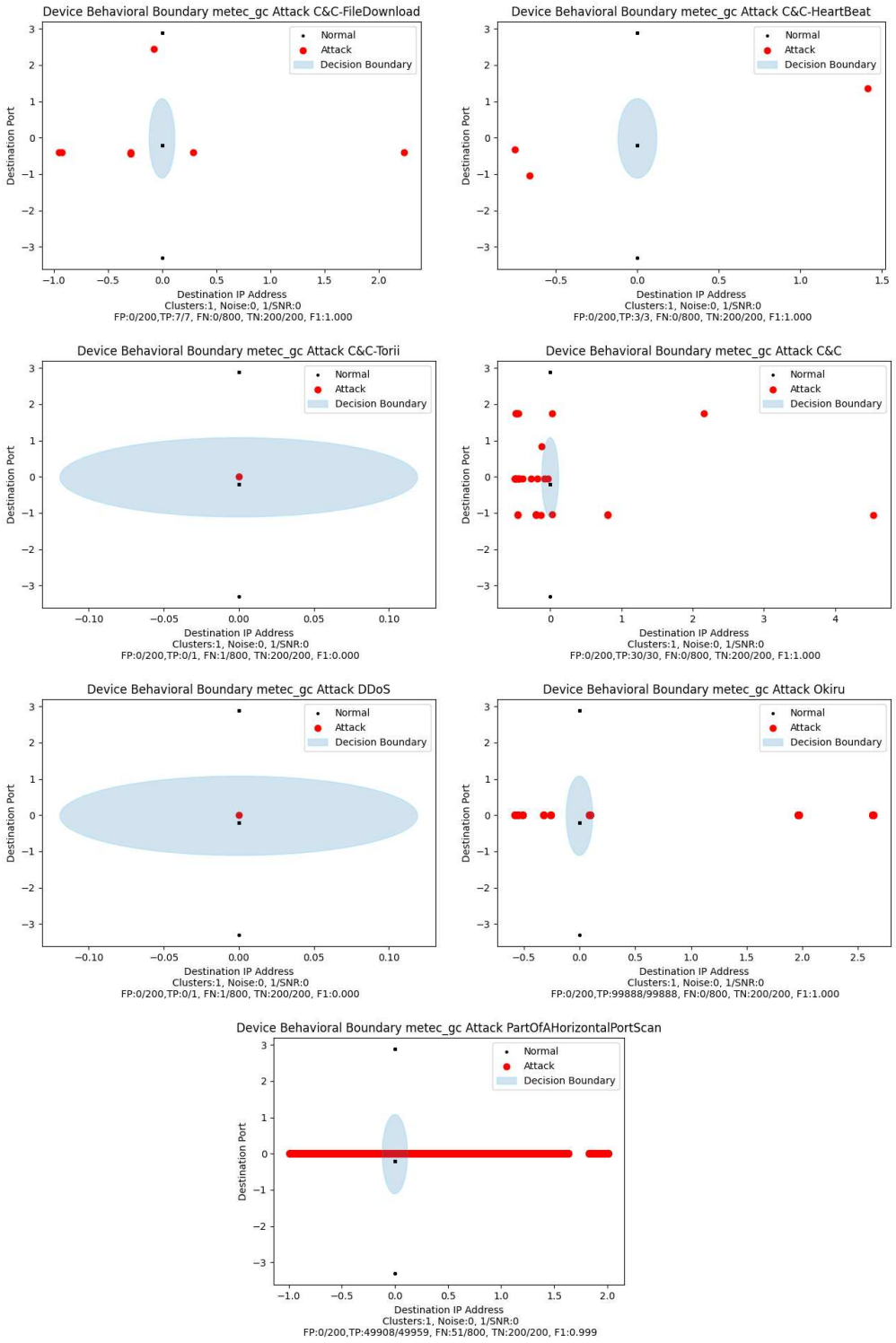


Figure E.20: metec-gc: SCADA

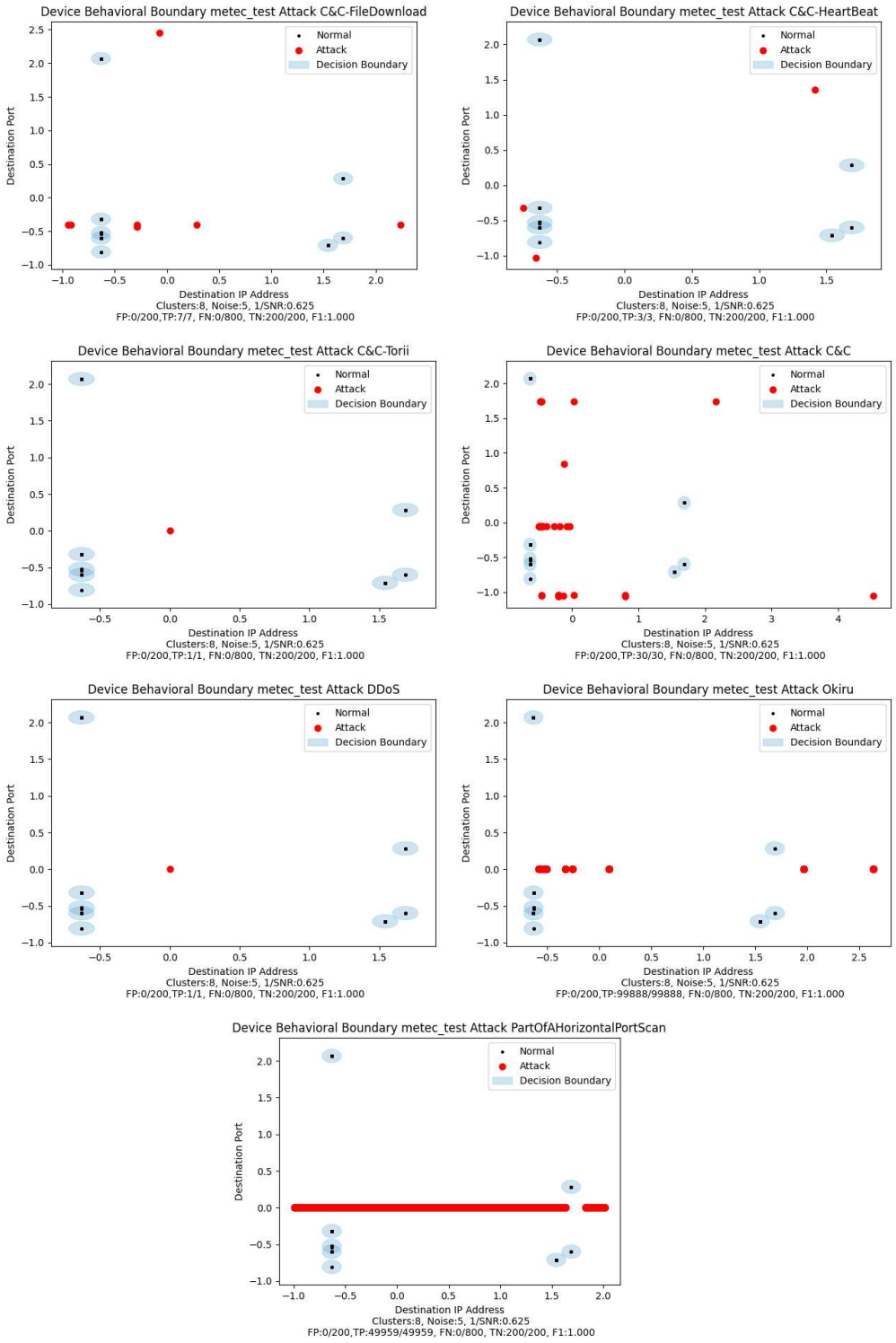


Figure E.21: metec-test: SCADA