Detecting Hidden Wireless Cameras through Network Traffic Analysis

KC K. Cowan

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Applications

September 2, 2020

Blacksburg, Virginia

Keywords: wireless cameras, network traffic analysis, privacy, detection

Detecting Hidden Wireless Cameras through Network Traffic Analysis

KC K. Cowan

(ABSTRACT)

Wireless cameras dominate the home surveillance market, providing an additional layer of security for homeowners. Cameras are not limited to private residences; retail stores, public bathrooms, and public beaches represent only some of the possible locations where wireless cameras may be monitoring people's movements. When cameras are deployed into an environment, one would typically expect the user to disclose the presence of the camera as well as its location, which should be outside of a private area. However, adversarial camera users may withhold information and prevent others from discovering the camera, forcing others to determine if they are being recorded on their own. To uncover hidden cameras, a wireless camera detection system must be developed that will recognize the camera's network traffic characteristics. We monitor the network traffic within the immediate area using a separately developed packet sniffer, a program that observes and collects information about network packets. We analyze and classify these packets based on how well their patterns and features match those expected of a wireless camera. We used a Support Vector Machine classifier and a secondary-level of classification to reduce false positives to design and implement a system that uncovers the presence of hidden wireless cameras within an area.

Detecting Hidden Wireless Cameras through Network Traffic Analysis

KC K. Cowan

(GENERAL AUDIENCE ABSTRACT)

Wireless cameras may be found almost anywhere, whether they are used to monitor city traffic and report on travel conditions or to act as home surveillance when residents are away. Regardless of their purpose, wireless cameras may observe people wherever they are, as long as a power source and Wi-Fi connection are available. While most wireless camera users install such devices for peace of mind, there are some who take advantage of cameras to record others without their permission, sometimes in compromising positions or places. Because of this, systems are needed that may detect hidden wireless cameras. We develop a system that monitors network traffic packets, specifically based on their packet lengths and direction, and determines if the properties of the packets mimic those of a wireless camera stream. A double-layered classification technique is used to uncover hidden wireless cameras and filter out non-wireless camera devices.

Dedication

This work is dedicated to my late Nanay Socorro, my parents, my sisters, my boyfriend, and all of the friends I made along the way at Virginia Tech. Thank you for being my greatest fans and always seeing the best in me.

Acknowledgments

Thank you to my advisor, Dr. Hicks, for accepting me onto his team and guiding me through my research. I would also like to thank my committee members, Dr. Gračanin and Metin Ahiskali, who lent a listening ear and provided feedback and direction when I found myself stuck. I thank my parents, Wayne and Aida Cowan, for never failing to support and encourage me through the highs and the lows. Thank you to my boyfriend, Nick Ekanger, for talking through every bug, pulling all-nighters with me, and making sure I knew that I had someone in my corner. Finally, thank you to everyone who offered their advice, patience, and kind words throughout my research process.

Contents

List of Figures **ix**

List of Tables **x**

1 Introduction **1**

1.1 Motivation 2

1.2 Problem Description 3

1.3 Research Objectives 3

2 Background **4**

2.1 Network Traffic Analysis 4

2.2 Networking 5

2.2.1 TCP/IP Model 5

2.2.2 IEEE 802.11 8

2.3 Wireless Cameras 10

2.3.1 System on a Chip (SoC) 11

2.3.2 Image Sensor Processor (ISP) 12

2.3.3 Codec 12

2.3.4 Networking Protocol 18

3 **Threat Model and Requirements** **20**

3.1 Attack Model 20

3.2 Design Requirements and Limitations 21

4 **Methodology** **23**

4.1 Wireless Traffic Sniffer 24

4.2 Data Collection and Processing 24

4.3 Machine Learning Models 26

4.3.1 Classification Techniques Used by Previous Work 26

4.3.2 Support Vector Machines 27

4.4 Implementation 30

5 **Results** **32**

5.1 Training Dataset 32

5.1.1 Data Parsing 32

5.1.2 SVM Classifier 33

5.1.3 Rolling Window Classification 35

5.2 Testing Dataset 38

5.2.1 SVM Classifier 38

5.2.2 Rolling Window Classification 39

6 **Discussion** **42**

7 Limitations **46**

8 Literature Review **48**

8.1 Applications of Network Traffic Analysis 48

8.2 Wireless Camera Detection 50

9 Conclusions and Future Work **52**

Bibliography **54**

List of Figures

2.1 General process for conducting network traffic analysis. 5

2.2 Architecture stack for the TCP/IP model. 6

2.3 Architecture stack for the OSI model. 8

2.4 General structure of an 802.11 data frame [19]. 9

2.5 Standard hardware components of wireless cameras [11]. 11

2.6 Example of a Group of Pictures (GOP) defined by the function $G(3, 9)$ [1]. . 14

3.1 Attack scenario where an Airbnb host has installed multiple cameras and routers throughout the home. Host only reported cameras in public spaces on listing. 21

4.1 Framework for this research's approach to network traffic analysis on wireless camera network traffic. 23

List of Tables

2.1 Address field values for different To and From DS bit combinations [19]. . . 10

5.1 Summary of bidirectional packet features collected from a wireless camera and YouTube video stream. All packet lengths are given in bytes. 33

5.2 Summary of packet feature values for both a Wireless Camera (WC) and YouTube (YT) video stream when packets are grouped into 500, 1,000, 5,000, and 10,000 ms bins. All packet lengths are given in bytes. 34

5.3 Summary of metrics for SVM training using seed values 0, 21, and 42 with 500, 1,000, 5,000, and 10,000 ms bin sizes. 35

5.4 Summary of metrics for SVM tested against initial dataset using seed values 0, 21, and 42 with 500, 1,000, 5,000, and 10,000 ms bin sizes. 36

5.5 Summary of metrics for rolling window reclassification with window size of 5. 37

5.6 Summary of accuracy, precision, recall, and number of false positive values when using the rolling window classification on the training dataset classified with a 10,000 ms bin size and seed value of 0. Window sizes used: 7, 9, and 11. 38

5.7 Summary of average SVM classifier metrics across the three different seed values for all bin size designations of the testing dataset. 40

5.8 Summary of accuracy, precision, recall, and number of false positive values for rolling window classification using window sizes of 5, 7, 9, 11, 13, 15, 17, and 19. 41

List of Abbreviations

BSSID	Basic Service Set Identifier
CCD	Charge-Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
DL	Downlink
DS	Distributed System
GOP	Group of Pictures
IP	Internet Protocol
ISP	Image Sensor Processor
MAC	Media Access Control
MCU	Microcontroller Unit
NAL	Network Abstraction Layer
PHY	Physical
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
SoC	System on a Chip
SVM	Support Vector Machines

TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UL	Uplink
VCL	Video Coding Layer
WLAN	Wireless Local Area Network

Chapter 1

Introduction

By 2021, wireless camera sales in the United States are projected to reach $32 billion, over three times the sales in 2016 [4]. While cameras have primarily been used for surveillance and anti-theft measures, their applications in public spaces expand beyond such duties. Monitoring systems, especially for smart cities, have arisen due to the combination of multiple wireless cameras with additional processing hardware capable of quickly analyzing video imagery. Identifying ideal travel times based on traffic congestion and locating available parking spots are only some of the benefits derived from using wireless cameras to analyze everyday life in cities [24, 27].

However, the popularity and accessible cost of cameras has led to their integration into the home, providing homeowners with additional security when they are away. In general, cameras may be found indoors, monitoring residents as they move in and out of the home, as well as guests who may have been left alone. With the rise of homestay websites such as Airbnb and CouchSurfing, more homeowners have installed cameras within the guest area to ensure that the home's integrity is maintained. While wireless cameras should be used as an additional layer of protection, adversaries may take advantage of the technology and use cameras as tools for privacy invasion. Due to their small size, cameras may be well concealed, leaving people unaware of the unwarranted surveillance.

1.1 Motivation

Typically, homestay companies require hosts to disclose the presence of a camera, with the stipulation that the device is only placed in public areas (i.e., living room, kitchen) [5]. However, hosts may easily withhold this information from the listing, leaving guests none-the-wiser of the wireless camera's presence, whether it is in a public or private area. In fact, multiple instances have been reported where guests have found cameras hidden in vents, smoke alarms, and even shampoo bottles, the latter resulting in imprisonment [15, 30, 36].

Using wireless cameras to invade privacy is not limited to homestays. Cameras may be hidden in everyday items that are strategically installed in areas so that their placement seems accidental. Remote-controlled drones may be outfitted with wireless cameras and flown around homes to observe residents through open windows. From a broader viewpoint, wireless cameras may be installed in parks, restaurants, public restrooms, beaches, changing rooms, and many other public spaces [13, 35]. All of these locations are susceptible to adversaries wishing to invade others' privacy, and with the small size of cameras on the market today, it is easy to discretely hide cameras and record unsuspecting people.

With the possible placement of multiple cameras throughout a home, across a beach, etc., one would want to prove that a device is possibly installed. To do this, a large collection of data is needed to notice the camera's unique features, meaning that there would need to be multiple data collection points. One approach would be to deploy multiple wireless camera detectors throughout the area, allowing them to collect as much diverse data as possible, thereby increasing the chances of device identification. For such an approach to work, these detectors would need to be small and cost-effective, making them portable and easily accessible. Fortunately, such devices do exist and may be used as traffic monitors, supplemented by additional machine learning techniques on a local machine.

1.2 Problem Description

For this research, we consider a scenario where a guest is staying at an Airbnb location. As stated, under Airbnb policy, hosts are allowed to install cameras in public, non-invasive areas but must report their presence on the listing. However, hosts may easily withhold this information or lie about the camera's location, thus leaving guests uninformed. We operate under the assumption that a host may have placed a wireless camera in the home without notifying the guest. The camera may be found in a public or private area. In either case, the camera is well-hidden so that it is not visible to the naked eye. Our goal is to determine if a camera is present, and, if possible, the room in which the camera has been placed.

1.3 Research Objectives

Developing a wireless camera detector can be done in a three-fold process. To start, we wish to create a wireless traffic sniffer, a program that observes and collects the packets traveling over the surrounding network. Data from both camera and non-camera devices will be gathered. We will preprocess this data to extract relevant features and form groupings based on the packets' timestamps. We aim to train a classifier that will use these features to accurately differentiate between traffic patterns for camera and non-camera network streams. Finally, we will determine the best application of the classification technique used, deciding if a single-level or hierarchical method produces the best results.

Chapter 2

Background

2.1 Network Traffic Analysis

Network traffic analysis provides in-depth information about a network and its users, identifying patterns in usage as well as any possible flaws or exploits [10]. Using this knowledge, network administrators can detect when and where a network is failing, including situations where unknown traffic has entered into the network stream or unprecedented congestion occurs. While network traffic analysis has obvious advantages, it may be exploited by adversaries looking to infiltrate a network to collect personal data, or simply to render the network useless for others. With the rise of mobile devices over the past two decades, using network traffic analysis to monitor user behavior has left people susceptible to attacks from those who recognize the patterns of specific applications and may extract credentials and private information from the generated network traffic [14].

In general, network traffic analysis is conducted in a four-step process (Figure 2.1). Network traffic may be collected using a self-developed packet sniffer or with popular monitoring tools like Wireshark [7, 26]. The data will be run through a preprocessing unit that will extract significant features that may be useful for the analysis, at which point behavioral patterns and network performance are uncovered. Finally, the trained model is used to evaluate future systems and identify characteristics of the network, as well as the people who use it.

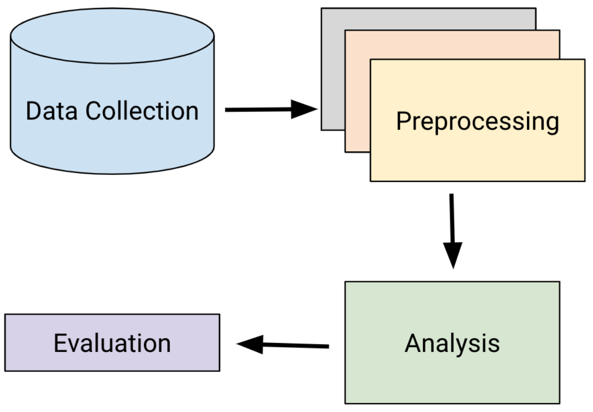

Figure 2.1: General process for conducting network traffic analysis.

Router-based monitoring techniques dominate the industry when collecting and analyzing network data. The Simple Network Monitoring Protocol (SNMP) runs at the application layer as a part of TCP/IP, passively calculating network statistics through administrator requests. Remote Monitoring (RMON) is similar to SNMP, but instead of requiring specific requests on the administrator's end, alarms can be set in the system that are tripped when suspicious activity is detected. Finally, Netflow actively collects and analyzes network traffic as it enters the interface, preemptively characterizing the network without explicit administrator interference. Non-router based techniques offer less flexibility and may be either active (probing the network to measure traffic behavior between two points) or passive (sniffing existing network packets at only one point).

2.2 Networking

2.2.1 TCP/IP Model

The Internet protocol suite (Figure 2.2), also called the TCP/IP model, was implemented by DARPA in the mid-1970s as a standard for Internet communication between networked devices [9]. The model utilizes the TCP and IP protocols as the foundations for reliable message transmission across a network. The architecture itself serves as an abstraction of the expected behaviors and relationships between communicating devices, allowing manufacturers to follow a set of standards when designing such machines. As network communication exists between various hosts with possibly different compositions, a standardized framework model allows for seamless communication and provides a base framework of design for developers constructing such systems.

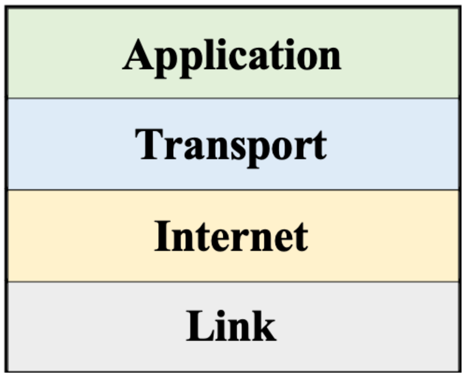

Figure 2.2: Architecture stack for the TCP/IP model.

There are four layers to the TCP/IP model (from top to bottom): Application, Transport, Internet, and Link. Within the Application layer, data begins in its raw form and is reformatted in preparation for handling by the network. The Transport layer will handle additional formatting before adding on a transport header, later establishing the connection for transmission of a segment (TCP) or datagram (UDP). The Internet layer provides the actual capabilities for transmitting packets, though unreliably. Finally, the Link layer encapsulates the information collected between the layers before physically moving packets between different hosts' Internet layers.

The Application layer allows a user and his computer to interface with the network, containing protocols necessary for user applications on a network. As the top of the stack, this layer relies only on the assistance of the layer below it, the Transport layer, to accurately provide support to users wishing to communicate through a network. There are two types of protocols in the Application layer: user and support. The former group assists users with network interfacing by providing direct services to the user (e.g., File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP)), while the latter pertains to any common functions within the system, including Domain Name System (DNS) protocols.

The Transport layer provides a connection between entities, handling the reliable transfer of data between them. The flow control through the network, as well as basic error checking, are conducted here. Two major protocols are available: TCP and UDP. TCP relies on a connection between communicating devices and ensures the reliable transmission of segments. Oppositely, UDP is connectionless and can be seen as less reliable when handling datagrams across the network.

The Internet layer handles the actual transmission of packets, but there is no guarantee for a reliable transfer through routing and IP addressing. The reliability of packet movement depends on the behaviors and settings of the layers above. This layer also deals with any fragmentation or security checks. Because IP is connectionless, it remains agnostic when handling data for upper layer protocols. Examples include the Internet Control Message Protocol (ICMP), used for error and congestion reporting, and Internet Group Management Protcol (IGMP), applicable to the establishment and management of group hosts in IP multicasting.

Within the Link layer, a link is directly established between two nodes to allow for immediate data transmission. When reading data from the network, this layer takes the information and compiles it into a frame, forming a consolidated package that will be communicated between devices; the MAC address of a frame is attached here. The IEEE 802 standard maps to the Link layer, providing additional behaviors and formatting guidelines when executing major

Local Area Network (LAN) protocols. The 802.11 standard, which handles Wireless Local Area Networks (WLANs), will be discussed in the next section.

The Open Systems Interconnection (OSI) model (Figure 2.3) was developed soon after by the International Organization for Standardization (ISO) and introduces an additional three layers: Presentation, Session, and Physical [20]. The Presentation and Session layers take over some of the duties defined with the Application layer, formatting data and establishing sessions for future connections. The Physical layer handles the actual transmission of bits between devices. In addition, the Internet layer is renamed as the Network layer, but the duties remain the same.

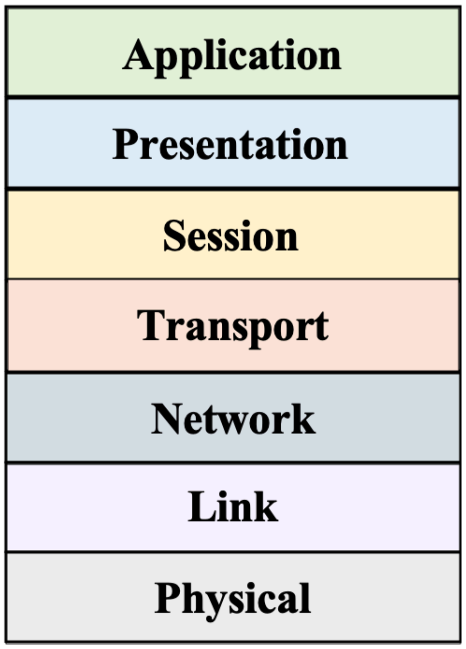

Figure 2.3: Architecture stack for the OSI model.

2.2.2 IEEE 802.11

The IEEE 802.11 standard handles communication between WLANs [19]. The data format handled by the Link layer, once encapsulated with a MAC address and other pertinent information, is called a frame. 802.11 frames are sent between communicating devices, containing all of the information necessary to understand the destination, source, and purpose of a message. There are three types of 802.11 frames: control, management, and data. Management frames handle the initial connection between access points and stations, while control frames deal with channel access and frame acknowledgement. Data frames contain the actual information that is being exchanged between hosts. For this research, we focus on data frames as wireless camera video sequences are transmitted through this frame type. It is important to note that 802.11 frames may be encrypted. However, encryption impacts the payload of the frame, while this research only makes use of the information found in the frame header.

All 802.11 frames follow a standard format, with a data frame's fields shown in Figure 2.4. The Frame Control field is a 16-bit stream that contains general setup and format information, including the type and subtype of the frame, the transmission direction, and the protocol version. Two important bits within this field indicate what type of hosts are communicating and in which direction, labeled as To and From DS. Based on the setting of these bits, one can know if a frame originated from a station and is being sent to an access point, or vice versa. For the purpose of this research, we want to know if a station is sending a frame to an access point, represented by setting To DS = 1 and From DS = 0.

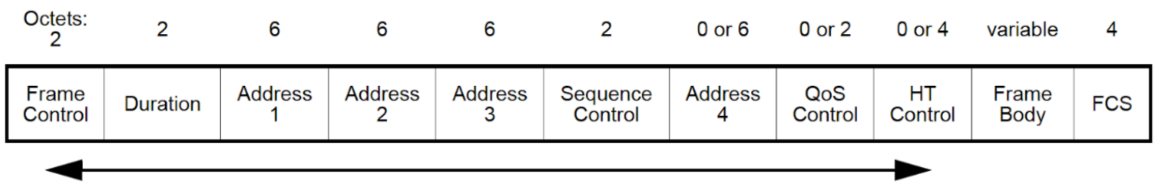

Figure 2.4: General structure of an 802.11 data frame [19].

Four address fields may be found in all frames, but all of the addresses may or may not be used. In general, frames are sent between access points and stations. Access points, which contain a station, distribute a wireless connection to a variety of stations, managing them as necessary. A station is a device that connects to an access point, utilizing the network connection provided and executing behaviors directed by the user. Based on the setting of the To and From DS flags, the usage and meaning of the address fields will vary. Table 2.1 summarizes the address assignments according to the flags' settings.

The BSSID describes a section of a service set, a group of wireless network devices identified by the same SSID, which commonly refers to the network's name. BSSIDs are formatted similarly to MAC addresses but specifically serve as the MAC address of the radio interface to which a client is connected.

Table 2.1: Address field values for different To and From DS bit combinations [19].

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	Destination	Source	BSSID	N/A
0	1	Destination	BSSID	Source	N/A
1	0	BSSID	Source	Destination	N/A
1	1	Receiver	Transmitter	Destination	Source

2.3 Wireless Cameras

In general, wireless cameras are composed of the same hardware: a lens, an image sensor, a dedicated System on a Chip (SoC) for encoding and transmission preparation, and a Wi-Fi chip [3, 18]. Figure 2.5 depicts these components and how they interact. When recording an image, the lens focuses the light reflected off the captured objects onto the image sensor. The image sensor is separated into multiple units, called pixels. Each pixel records the intensity of the light that reaches it, converting the light into electronic signals for the camera's image processing circuitry to then handle and prepare for transmission. There are two main types of image sensors: a charge-coupled device (CCD) and a complementary metal oxide semiconductor (CMOS). CCD sensors are geared towards more sophisticated cameras, specifically those used for high-quality broadcasting. In contrast to CCD, CMOS sensors are cheaper and require less power, making them the perfect choice for network cameras, smartphones, and other consumer-dedicated products. CMOS sensors process each pixel individually, handling each charge before transmitting a single digital bit. Once the image has been converted into bits readable by the SoC, it is then processed, compressed, and fragmented before transmission over the network using the Wi-Fi chip. Once the video has been transmitted, it can be reviewed in real-time or stored for future viewings.

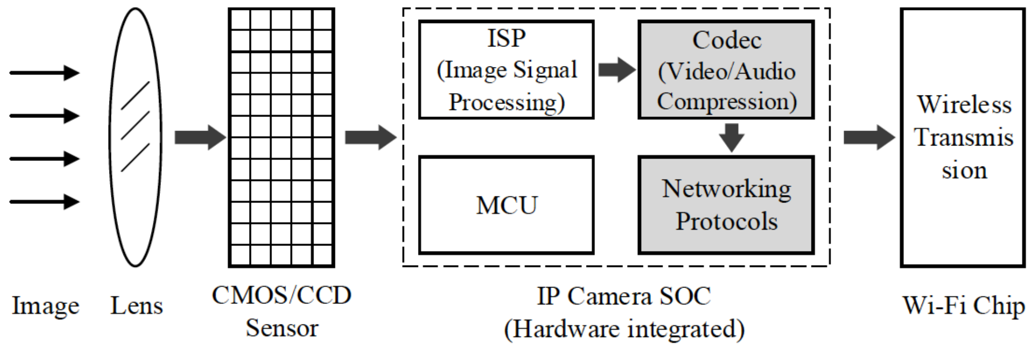
ISP (Image Signal Processing)
Codec (Video/Audio Compression)
MCU
Networking Protocols
Wireless Transmis sion
Image
Lens
CMOS/CCD Sensor
IP Camera SOC (Hardware integrated)
Wi-Fi Chip

Figure 2.5: Standard hardware components of wireless cameras [11].

2.3.1 System on a Chip (SoC)

Wireless cameras typically have dedicated hardware for transmitting video data over the network. SoCs contain the same general components: a microcontroller unit (MCU), an image sensor processing (ISP) module, a codec module, and a networking protocols module. While the MCU runs the SoC, the three other modules are responsible for processing the received data, compressing the images into smaller frames, and preparing these frames for transmission to some remote storage.

2.3.2 Image Sensor Processor (ISP)

Using the information provided by the image sensor, the ISP will interpolate the image, building the frame and providing the colors based on the color of the surrounding pixels through a process called demosaicing [22]. The ISP also handles clean-up duties, including focusing, noise reduction and filtering, and sharpening. By refining the provided image, the system has a clearer, more accurate depiction of the recorded scene that will provide a more attractive representation of the video when reviewed on a remote machine.

2.3.3 Codec

Once an updated, improved frame has been built, it must be compressed to a smaller size, simplifying transmission and utilizing less network bandwidth. The most popular standard for video compression is the H.264 Video Coding Standard. H.264 was introduced in 2003 in response to the rising need for higher compression for large-scale applications such as video conferencing, Internet streaming, and broadcast television [17, 33]. The standard was designed with flexibility and customizability in mind, allowing users to integrate H.264 with a variety of applications and networks while also modifying it to meet their needs.

H.264 covers two main layers: the Video Coding Layer (VCL) and the Network Abstraction Layer (NAL). The VCL handles the efficient representation of the data, while the NAL takes the VCL representation and formats it so that it can be transmitted through different mediums. The main goal of the NAL is to provide "network friendliness", allowing for customization of the VCL for various purposes, specifically for the video representation of both conversational and non-conversational applications. The VCL handles the compression and segmentation of the video images, producing the formatted results utilized in this research's approach.

Video Coding Layer (VCL)

The H.264 VCL follows the block-based hybrid approach for video coding. Each coded picture is represented through block-shaped units of associated luma and chroma samples, called macroblocks. Luma samples depict the brightness of the image, or the "black-and-white" portion, while chroma samples handle the necessary color information for the image. The algorithm makes use of both intra-compression and inter-compression of the video frames, where the former handles statistical redundancy and the latter handles temporal redundancy. Statistical redundancy monitors the similarities between neighboring pixels within a single frame, while temporal redundancy marks the similarities between neighboring frames within a video sequence.

The color space of the samples follows a YCbCr distribution, where Y represents the luma samples and Cb and Cr represent the chroma samples, specifically concerning the extent to which the image's color deviates from gray towards blue and red, respectively. The human eye is more sensitive to the brightness of an image than to the color, so H.264 utilizes a sampling structure where the number of chroma component samples is one-fourth the number of luma component samples, halving the number of samples in both the horizontal and vertical dimensions. This is called 4:2:0 sampling with 8 bits of precision per sample.

Macroblocks are the basic building blocks for the video coding process. Macroblocks are formed when pictures are partitioned into fixed-size, rectangular picture areas that cover 16x16 luma samples and 8x8 chroma samples for each chroma component (i.e., Cb and Cr). Slices, sometimes called frames, contain a sequence of macroblocks that are processed according to the rectangular pattern of the image's capture and reconstruction. Since pictures can be split into one or more slices, a picture is a collection of one or more slices within the H.264 standard.

All slices can be coded using the following encoding types [2]:

- Intracoded (I) slices: all macroblocks within the slice are encoded using intra-prediction. These are the largest slices of the encoding types.
- Forward Predictive (P) coded slice: macroblocks can be coded with intra- or inter-prediction, making use of motion changes from preceding I and P slices.
- Bidirectional (B) Predictive coded slice: macroblocks can also be coded with intra- or inter-prediction, but instead makes use of any motion changes from both preceding or subsequent I and P slices. These are the smallest of the encoding types.

The combination of these slices forms a Group of Pictures (GOP) (Figure 2.6), which always starts with I frames [1]. GOPs can be described with the function $G(m, n)$, where m represents the distance between I and P slices and n represents the distance between I slices.

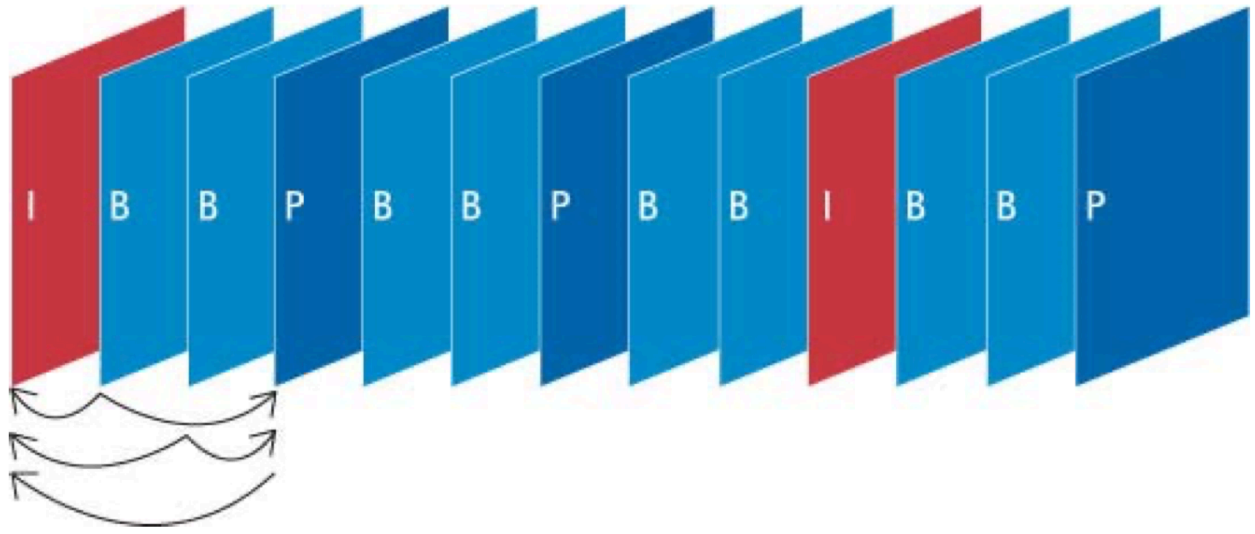

Figure 2.6: Example of a Group of Pictures (GOP) defined by the function $G(3, 9)$ [1].

All luma and chroma samples are either spatially or temporally predicted. In general, I slices are encoded using intra-compression, while P and B frames make use of inter-compression.

When using intra-compression, algorithms are to individual blocks of a video frame, not the entire macroblock. The video frame is divided into 16x16 pixel macroblocks, which are composed of four 8x8 luma samples and some number of 8x8 chroma blocks. The Discrete Cosine Transform (DCT) is applied to each block and breaks the image into separate parts based on its contribution to image's quality. Quantization then compresses the resulting coefficients into a single quantum value and replaces the original coefficients with 0, providing more room for further compression. The codec at the receiving end will then handle the data in reverse. Inter-compression, on the other hand, applies to full macroblocks, using predictive encoding on some frames to account for the minor changes between them. Errors in a frame, depending on its type, can have very little or very great consequences that propagate throughout the frame sequence. If an error is found while encoding an I frame, all future P and B frames are impacted; a similar result occurs for encoding errors in P frames. However, errors while encoding B frames only affect the relevant B frame, reducing its impact on the entire sequence.

Network Abstraction Layer (NAL)

The NAL was designed with the intent to enable "network friendliness", permitting customization of the VCL usage in order to operate with a wide variety of transport layers and protocols. For wireless video transmission, the NAL will map the H.264 VCL data to the RTP protocol, or some variant of it.

NAL units, the data format at this level, are composed of coded video data structured as a packet with some number of bytes. The header byte indicates what type of data is encompassed within the NAL unit, with the remaining bytes containing the actual payload. When necessary, the payload may be interleaved with emulation prevention bytes, which helps break up the byte pattern to prevent the random appearance of the start code prefix.

This format is generalized, allowing the NAL units to be used for both packet-oriented and bitstream-oriented transport systems. Packet-oriented transmission is organized according to the system's transport protocol and allows for identification of the NAL unit boundaries within the packet itself, removing the need for a start code prefix. This avoids unnecessary data transmission, reducing strain on the network when packets are transmitted.

NAL units can be either VCL or non-VCL. The former handles data that represents the values of the samples in the video pictures, while the latter contains additional administrative information, such as parameter sets and supplemental enhancement information. Parameter sets contain important header information that is expected to rarely change and offers decoding guidelines for a large number of VCL NAL units. Sequence parameter sets apply to a series of consecutive coded video pictures, also called coded video sequences, while picture parameter sets apply to one or more pictures within the coded video sequence, each handled individually. By using sequence and picture parameter sets, transmission for information that rarely changes is separated from the transmission of actual coded data. Parameter sets can be sent ahead of the VCL NAL units that they pertain to and can be retransmitted for robustness against loss of data. Each VCL NAL unit contains an identifier referring to the content of the relevant picture parameter set, with each parameter set then storing an identifier referring to the content of the relevant sequence parameter set. This organizational structure allows for a simple mapping between a small amount of data (the identifier) and a large amount of data (the parameter set) without repeating information.

A specified combination of the NAL units forms an access unit, which, when decoded, produces a single decoded picture. The base structure of an access unit is a set of VCL NAL units, constructing the primary coded picture. The VCL NAL units contain slices or slice data partitions that represent the sample of the video picture. The access unit delimiter prefixes the access unit, signaling the start of the unit. Supplemental enhancement information can also precede the primary coded picture. After the primary coded picture, additional VCL NAL units containing redundant representations of the same areas of the same video picture can be found. If the coded picture is the last of the coded video sequence, the end-of-sequence NAL unit marks the terminus. An end of stream NAL unit can be found if the coded picture serves as the conclusion of the entire NAL unit stream.

A coded video sequence contains a series of sequential access units in the NAL unit stream, using only one sequence parameter set. Each coded video sequence can be independently decoded, assuming the necessary parameter set information is provided. The sequence start is marked with an instantaneous decoding refresh (IDR) access unit, which holds an intra-picture, a coded picture that can be decoded without decoding any previous pictures in the NAL unit stream. By having an IDR, one knows that the pictures following the intra-picture will not need to refer to any pictures preceding the intra-picture when decoding. An NAL unit stream contains at least one coded video sequence.

When the NAL unit stream is being transmitted, it is segmented into 188-byte packets, forming a Packetized Elementary Stream (PES) [12]. These packets will then be grouped together and encapsulated to form some application-layer protocol packet, usually RTP. The UDP or TCP header, depending on which Internet protocol is used, will then be appended to the beginning of the packet. 802.11 Ethernet frames have a Maximum Transmission Unit (MTU) of 1,500 bytes, meaning that no more than 7 PES may be encapsulated into a single frame. The varying characteristics of I, P, and B slices lead to a unique distribution of packet lengths when observing a wireless camera stream. Since I slices are self-contained, they tend to be very large. When segmented into PES, this could result in multiple, full-length 802.11 frames being needed for full transmission. This leads to a steady number of consecutive, MTU-length packets, followed by a much smaller packet. A similar process exists for P slices, but there would be a fewer number of MTU-length packets, but a greater occurrence of this phenomenon as more P slices can be found than I slices. Finally, since B slices rely on both preceding and succeeding slices, they tend to be very small and typically fit into one 802.11 frame.

2.3.4 Networking Protocol

With compressed frames prepared for transmission, it is up to the networking protocol module to execute the necessary processes for packetizing and sending the data to an awaiting destination. Implemented at the application layer, the Real-Time Transport Protocol (RTP) handles the delivery of real-time media, namely audio and video [31]. RTP typically relies on UDP to make use of multiplexing and checksum services, allowing for a multicast stream of data. As stated above, part of the compression process involves the NAL encapsulating the VCL-outputted slices, forming NAL units (NALUs), which are more suitable for transmission over packet networks. An RTP header is appended to the beginning of the NALU, forming an RTP packet that will then be transmitted between hosts. Parameter sets can be sent ahead of these packets, whose headers may contain a list of sequence and picture parameter sets, as well as a codeword specifying the exact parameter set(s) to use.

The RTP payloads may be structured three different ways: single NALU packet, aggregation packet, and fragmentation unit [38]. A single NALU packet contains only one NALU inside the payload, while an aggregation packet combines multiple NALUs into a single RTP payload. Fragmentation units are formed oppositely from aggregation packets, breaking a single NALU into multiple chunks that will then be sent using multiple RTP packets.

The Real-Time Streaming Protocol (RTSP) may augment RTP by handling setup and control duties when delivering data with real-time properties, including pausing, positioning, and rewinding playback [32]. It is meant to be used as a “network remote control” for multimedia servers. Using a connected-oriented (TCP) or connectionless (UDP) session, clients first request information about streaming media from servers using media descriptions. RTSP assists with establishing the sessions through which the server may respond with the correct media, allowing clients to control the media's playback. Depending on the type of session connection used, clients and servers can communicate through a series of messages. A persistent connection, where the connection between the client and server remains open once established, is recommended for all transactions between clients and servers. In the case of transient connections, where each transaction has its own connection, servers will have no way to send a request to the client because the session context will have become obsolete.

Chapter 3

Threat Model and Requirements

3.1 Attack Model

We consider a scenario where a guest is residing in an Airbnb with wireless cameras present. Figure 3.1 presents an overview of the device configurations. Within the home, the home-owner (the adversary) has installed multiple security cameras in both public (green) and private spaces (red), only alerting the guest of the cameras in the public spaces. The router in the living room is available for guests to utilize. All devices with a red border are unknown to the guest. We make the following assumptions:

- In this case, the cameras are connected to an unknown router. However, the guest has no prior knowledge of which access point the cameras are connected to and should assume that the cameras may be communicating with a network separate from the one provided to the guest.
- Each camera is uploading a continuous video stream to the adversary's machine.
- The adversary is the only individual who has access to the camera feed.
- The adversary alone has the credentials to access the camera and the network upon which it is connected.
- The cameras are well-concealed and invisible to the naked eye.

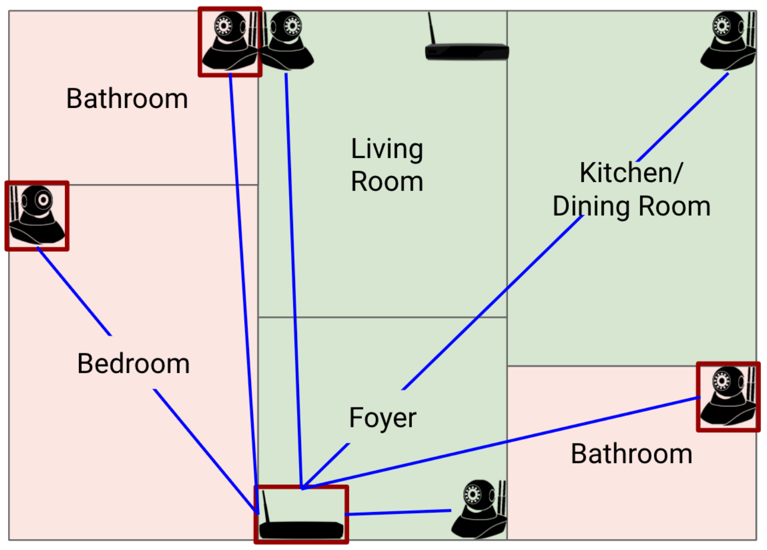

Figure 3.1: Attack scenario where an Airbnb host has installed multiple cameras and routers throughout the home. Host only reported cameras in public spaces on listing.

3.2 Design Requirements and Limitations

When the wireless camera detection system is deployed within a possibly hostile environment, its functionality must fit within a specific set of limitations:

- There is no guaranteed access to any portion of the network packet aside from the PHY/MAC headers. In addition, the network packet, and therefore the payload, may be encrypted. All of the necessary data needed to identify a wireless camera should be extracted from the headers.
- The adversary who implemented the camera has full control over the configuration of cameras within the home. This limitation also implies that more than one camera may be installed. The system must be able to observe multiple wireless camera flows.
- The adversary retains sole access and maintenance of the Wi-Fi network(s) installed in the home. Cameras may or may not be connected to a network different from the one

provided to the guest. The system must perform independent of a network connection.

Chapter 4

Methodology

The major components of the project include the wireless traffic sniffer, data processing unit, and classifiers. The sniffer collects the necessary information from each network traffic packet, storing the data in a text file that will then be passed to the data processing unit. This section of the system pipeline handles the grouping and organization of the packets. Finally, the processed data is used to build and train the classifier that will detect a wireless camera in later collections. The Amcrest ProHD 1080P Wi-Fi Camera is used for all training and testing. Figure 4.1 (based on Figure 2.1) presents the network traffic analysis pipeline through which the wireless camera network data is processed and analyzed to assist with wireless camera detection.

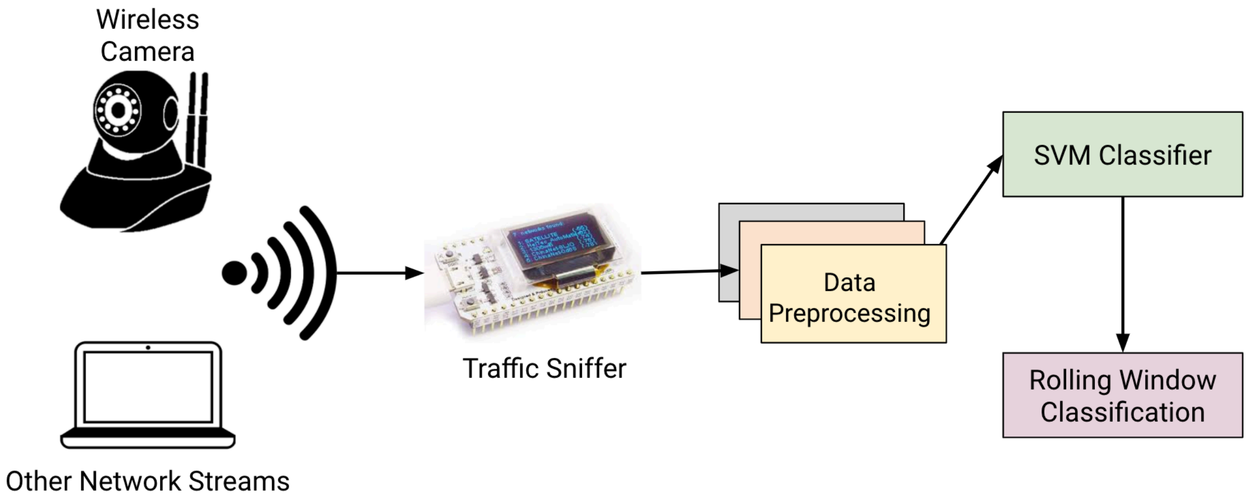

Figure 4.1: Framework for this research's approach to network traffic analysis on wireless camera network traffic.

4.1 Wireless Traffic Sniffer

To gather network information from communicating devices within the area, a wireless traffic sniffer was developed to observe the packets and their flows. When gathering the packets, the sniffer can either survey the entire collection of network traffic or lock onto a MAC address of interest. The former would be more applicable in the case where multiple cameras could be found in the home. The amount of time spent on each channel is malleable to the user's desires, but 5 to 10 seconds would allow for enough time to collect large amounts of data for the current channel while enabling quick channel rotations. When the sniffer locks onto a specific communication stream, it does so based on the number of consecutive maximum length packets, as this would be indicative of a possible I or P frame. The sniffer would then cease any channel rotation and solely focus on the given network stream.

For each packet, the following fields and flags were extracted: destination address, source address, channel, packet length, and direction. The direction was determined based on the To and From DS bit settings in the Frame Control field of the packet. In addition, the local UNIX timestamp, which was relative to when the ESP32 was powered on, was calculated for each packet, as well. All of this information was written out as a single line to a text file for data processing later in the system pipeline.

4.2 Data Collection and Processing

Two sets of data were collected and analyzed for this research. The training data for the classifier was gathered through the reconfigured wireless traffic sniffer, which only handled uplink and downlink packets from the wireless camera and a YouTube video streams. An hour's worth of data was collected for both devices. The packets were compiled into groups based on their local timestamp values, forming bins with timestamp ranges of 500, 1,000, 5,000, and 10,000 ms. The streams for the two devices were collected separately, with the wireless camera data being monitored first. To collect the testing data, the sniffer was then set to its locked-on mode and analyzed the entirety of a network stream for 10 minutes, stopping on the relevant channel when the camera was detected and only collecting the camera's network packets. The camera was marked as a device of interest during the first round of packet sniffing. Each channel was monitored for 10 seconds before the sniffer moved on. Using less time for data collection more accurately simulates a scenario where one is trying to detect a hidden camera.

Each bin maintains a list of packet lengths specific to the direction of the packets. A variety of uplink and downlink packets may reside within the same bin, so the average value of their lengths was maintained to monitor stability, as well as assist with choosing the optimal bin size. A master list of all packet lengths within the bins was maintained as well. In addition, a list containing values representing the number of consecutive maximum length packets was also present for each bin. The current packet's length was specifically checked to see if it is greater than or equal to the MTU (1500 bytes), contributing to both the respective bin's and the overall consecutive maximum length packets lists. Anytime a non-maximum length packet was read, the counter variable used to represent such a value was added to the lists and reset to 0.

Once this data was collected and calculated, the consecutive maximum length packets lists for each bin were checked, counting the number of possible I and P frames within each bin. Based on previous data collections with the camera, we observed that 6 or less consecutive maximum length packets indicated a P frame, while anything greater would most likely be an I frame. Because of this, the threshold value used was 6 maximum length packets in a row, where more than 6 would indicate an I frame and anything less than or equal to 6 would indicate a P frame. Since a Group of Pictures (GOP) contains more P frames than I frames, when looking at a typical wireless camera stream, we expected to see a greater proportion of P frames than I frames, especially when the content of the video contained no movement. The I and P frames were counted for the overall data collection, as well.

When writing out data for the classifier, the device address, access point address, average total packet length, average uplink (UL) and downlink (DL) packet lengths, proportion of UL packets, proportion of DL packets, proportion of I frames, and proportion of P frames were all calculated and exported. The wireless camera was marked as being part of class 1 (camera) and the YouTube stream was marked as class 0 (non-camera). The combined results of the wireless camera and YouTube stream were then used to train the classifier. For the testing dataset, the devices were marked as being part of class 0 or class 1 depending on the MAC address of interest that was recorded during the packet sniffing process. Because of this, all streams not belonging to the wireless camera were marked under the non-camera class. These class settings were used for testing the classifiers' performance later.

4.3 Machine Learning Models

4.3.1 Classification Techniques Used by Previous Work

Previous research done in detecting hidden wireless cameras have utilized a variety of classification techniques for their systems, both inside and outside the realm of machine learning. Cheng et al. [11] ran their data through 5 different supervised learning algorithms: SVM, AdaBoost, GradientBoosting, Random Forest, and ExtraTrees, randomly allocating 30% of their data towards training. Results were shown to be strong across all 5 methods, supporting the authors' decisions for feature selection. ExtraTrees was the method used throughout the duration of their experimentation due to its higher accuracy value and robustness for a single classification algorithm. In [39], a threshold-based classification technique was tested against a neural network, with the former producing errors 24% of the time. However, the optimal versions of each classifier were used to test their system, with the neural network boasting significantly greater accuracy, precision, and recall scores. The human presence detection system in [21] used a Long Short-Term Memory (LSTM) network, a model geared towards time sequences of data points. Their classifier read in the packet length sequence for the flows to determine if its pattern matched that of a wireless camera. Accuracy, precision, and recall values were all at or above 98%.

4.3.2 Support Vector Machines

For this research, the main attributes observed from the network stream were the presence of the correct To and From DS bit settings, the distribution of packet lengths over a given number of packets, and the proportion of I and P frames. Based on the few variables used for consideration, it was determined that a Support Vector Machine (SVM) classifier was the most reasonable training algorithm for this project.

SVM is a supervised machine learning algorithm that makes use of kernel tricks to produce a model that best predicts target values according to the testing data that was used to train the model [8]. In general, the goal is to determine the most accurate hyperplane, or boundary, between classes of data so that the data can be best classified through kernel function manipulation into some higher dimensional space. A kernel function computes the dot-product between the input vector and each support vector. When the model is first generated, data points are plotted around a linear boundary line, whose equation is determined by whatever kernel function is used (i.e., linear, polynomial, gamma, etc.). Support vectors are then identified as the vectors on either side of the boundary that are the closest to the line, and these support vectors drive the orientation of the boundary line. Therefore, manipulation of the support vectors directly modifies the boundary line. The goal when using as SVM classifier is to maximize the distance between support vectors and the boundary line through minimization of the weight vector. By maximizing this distance, the model is further reinforced to ensure classification with more confidence.

There are a few kernel functions that are widely used for SVM classifiers. In each case, x_i indicates a training vector, data upon which the model has already been adjusted, and x_j indicates the target vector, incoming data from new sources. The most popular kernel functions $k(x_i, x_j)$ are as follows:

- Linear: $\mathbf{x}_i^\mathsf{T} * \mathbf{x}_j$
- Polynomial: $(\gamma \mathbf{x}_i^\mathsf{T} \mathbf{x}_j + r)^d$
- Gaussian: $exp(\gamma ||\mathbf{x}_i^\mathsf{T} - \mathbf{x}_j||^2)$

γ, r, and d are kernel parameters, which can be adjusted to better manipulate the kernel to the shape of the data.

The dataset was split into the different bin sizes mentioned in Section 4.2. An SVM classifier using a linear kernel was built for each bin size, using all of the columns except for the MAC addresses as training features; the "Class" column was used as the benchmark for the classifier's performance metrics. The dataset was then split so that 90% was used for training, with the remaining 10% serving as the test points. When the flag `random_state` is not specified within the splitting function, a random seed is used for splitting. For consistency between the different bin sizes, the `random_state` flag was set to 0, 21, and 42 each time the data is separated.

When the model was run against the test dataset, the accuracy, precision, and recall of the model were calculated. The precision (P) of the model was based on the following equation:

$$P = \frac{TP}{TP + FP} \tag{4.1}$$

where TP represents the number of true positives and FP represents the number of false positives.

The model's recall was calculated using the number of correct classifications, defined as:

$$R = \frac{TP}{TP + FN} \tag{4.2}$$

with FN being the number of false negatives.

Finally, the accuracy of the model was determined using the given equation:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.3}$$

where TN is the number of true negatives. By using all three metrics, we observed the classifier's strengths and weaknesses, mainly focusing on the number of false positives. False positives were given priority over false negatives with the classifier because of the trade-off between them. False negatives can be filtered out using additional classification techniques so that the true positive is eventually uncovered, thus increasing the latency of the classifier. However, a longer detection rate introduces lesser consequences than an incorrectly classified device, as false positives could lead to accusations that may have detrimental effects on both parties.

4.4 Implementation

To collect a large amount of data over a widespread area, utilizing multiple packet capture units is optimal. In order for such a system to be realistically used by everyday consumers, the units must be cost-effective as well as manageable in size, making them easy to transport. An example of such a device, used in this research, is the ESP32. The ESP32 is a dual-core MCU integrated with Wi-Fi and Bluetooth capabilities, part of Espressif Systems' series of chips with similar capabilities [16]. The chip and its variations have been integrated with other modules to form development boards from various companies, including HiLetgo and Heltec. The ESP32 can be programmed using the ESP Internet of Things (IoT) Development Framework (esp-idf) or through its Arduino variant. C, C++, and Arduino are the languages mainly used for coding. For this project, the ESP32 was programmed using C and the esp-idf libraries. The HiLetgo ESP32 OLED and HiLetgo ESP-WROOM-32 development boards were used for programming and testing. The wireless sniffer used in this project is modeled after the sniffer developed in [28].

To enable wireless traffic sniffing on the ESP32, the chip must be set to promiscuous mode during the Wi-Fi settings initialization process, allowing it to monitor the network traffic and register the packets. The chip will repeatedly cycle through channels 1 to 11. As each packet is recorded, it is read in as a **void*** type, so it must be cast to the promiscuous packet type defined by the esp-idf. Only 802.11 data frames are monitored. From there, each packet can be separated into its payload and 802.11 header. The radio header information of the promiscuous packet may be accessed using an esp-idf structure. The packet payload and 802.11 header structures are user-defined, allocating bits for the necessary fields. Major fields include the Frame Control and address fields.

One major limitation of the ESP32 is its inability to handle languages outside of the C family.

Because the SVM classifier is developed using Python, two separate machines must be used for data collection and processing. This prevents consolidation of the entire system onto one device, requiring explicit user participation in collecting deployed ESP32 development boards to analyze the network traffic data that was observed. A local machine was used to run the data preprocessing and SVM training programs, both of which were developed in Python.

As mentioned, only the Amcrest ProHD 1080P Wi-Fi Camera was used for this research. The restriction on cameras used was due to financial limitations. Amazon's top-rated cameras all provide free access to the camera feed as long as the recording is viewed through a mobile application on a smartphone connected to the same network as the camera. Because of this, the network traffic patterns differed from what was expected, presenting video data as a constant stream of DL packets when the application was open; video transmission did not occur if the application was inactive.

Chapter 5

Results

5.1 Training Dataset

The wireless camera and YouTube video stream traffic were collected separately, each over the span of one hour. 166,142 wireless camera and 44,192 YouTube video network packets were captured by the sniffer, encompassing all UL and DL communications within the time frame.

5.1.1 Data Parsing

The information collected by the sniffer was passed through a parser that grouped packets into 500, 1,000, 5,000, and 10,000 ms bins. Table 5.1 and Table 5.2 summarize the major findings of the parser, with Table 5.1 focusing on the metrics of the entire dataset and Table 5.2 specifically highlighting the average proportion of UL packets, DL packets, I frames, and P frames for each bin size. It should be noted that Table 5.2 averages the grouped results, meaning that the dataset for these metrics is smaller and varied greatly with a change in bin size. As the bin size grew larger, we see very few surprising changes in the average values for the YouTube stream. The DL packets sent to the laptop were generally sized at 1432 bytes, with a few instances of smaller packets appearing. By increasing the bin size and grouping more of these maximum-sized packets together, we understandably saw a steady increase in the average overall and DL packet lengths.

Table 5.1: Summary of bidirectional packet features collected from a wireless camera and YouTube video stream. All packet lengths are given in bytes.

	Video Stream Type	
Feature	Wireless Camera	YouTube Video
Total Number of Packets	166,142	44,192
Avg. Total Length	353.108	1339.443
Avg. UL Length	523.366	426.221
UL Ratio	0.587	0.058
Avg. DL Length	111.095	1398.127
DL Ratio	0.413	0.942
No. of I Frames	294	0
I Frames Ratio	0.165	0.0
No. of P Frames	1,488	114
P Frames Ratio	0.835	1.0

However, the size of the bin had a prominent effect on the wireless camera values, specifically with the proportion of I and P frames. Between all four bin sizes, the average packet lengths and average proportions of UL and DL packets remained relatively stable. For the 500 and 1,000 ms bins, the proportion of I and P frames appeared to be 0.5, each. The proportion slightly increased in favor of P frames for the 5,000 ms bin and we saw a more expected distribution when using the 10,000 ms bin. Since we expected to see a greater number of P frames than I frames due to the characteristics of a Group of Pictures (GOP), the 10,000 ms bin provided the best representation of wireless camera packets within a network.

5.1.2 SVM Classifier

Using the combined data points from the wireless camera and YouTube video stream, a dataset was built to train the SVM classifier. As stated, the wireless camera was marked as part of class 1, with the YouTube stream falling under class 0. To monitor stability in

Table 5.2: Summary of packet feature values for both a Wireless Camera (WC) and YouTube (YT) video stream when packets are grouped into 500, 1,000, 5,000, and 10,000 ms bins. All packet lengths are given in bytes.

	Bin Sizes (ms)							
	500		1000		5000		10000	
Avg. Features	WC	YT	WC	YT	WC	YT	WC	YT
Total Length	343.278	1091.163	348.971	1029.887	350.580	1282.171	350.486	1302.336
UL Length	500.067	451.966	511.578	459.017	512.645	409.233	510.687	410.388
UL Ratio	0.588	0.112	0.587	0.155	0.584	0.062	0.582	0.069
DL Length	111.384	1158.735	112.529	1110.099	116.226	1338.216	117.521	1367.912
DL Ratio	0.412	0.888	0.413	0.845	0.416	0.938	0.418	0.931
No. of I Frames	255	0	270	0	292	0	287	0
I Frames Ratio	0.037	0.0	0.077	0.0	0.161	0.0	0.154	0.0
No. of P Frames	1,660	114	1,594	114	1,498	114	1,520	114
P Frames Ratio	0.238	0.061	0.446	0.093	0.806	0.169	0.810	0.284

precision, accuracy, and recall, the SVM classifier for each bin size was run three times. The dataset was split so that 90% of the values were dedicated towards training the model, with the remaining 10% serving as the testing set to calculate the relevant diagnostic values. To maintain consistency for each bin, we used three different seed values – 0, 21, and 42 – for the data splitting and compared the accuracy, precision, and recall between each seed value for all four bin sizes. Table 5.3 presents these values.

While the accuracy, precision, and recall values scored close to 100%, a significant number of false positives were produced when the trained model was run against the testing set. When using the model to identify possible cameras, this resulted in a non-zero number of devices that were not wireless cameras to be classified as such. If this were a real-life scenario, this misclassification could lead to issues of distrust. To see how the model would classify the entire dataset, all of the data points were passed as input to the model with each seed for all bin sizes. The results (Table 5.4) show a proportionally similar number of false positives and false negatives, both of which were still significant.

Table 5.3: Summary of metrics for SVM training using seed values 0, 21, and 42 with 500, 1,000, 5,000, and 10,000 ms bin sizes.

		Bin Sizes			
	Model Metrics	500 ms	1000 ms	5000 ms	10000 ms
Seed = 0	Accuracy (%)	98.955	98.287	100.0	98.529
	Precision (%)	99.703	99.706	100.0	97.222
	Recall (%)	98.968	97.977	100.0	100.0
	No. of False Positives	2	1	0	1
Seed = 21	Accuracy (%)	99.071	99.572	99.2	100.0
	Precision (%)	99.552	100.0	98.684	100.0
	Recall (%)	99.256	99.438	100.0	100.0
	No. of False Positives	3	0	1	0
Seed = 42	Accuracy (%)	99.652	98.929	100.0	98.529
	Precision (%)	99.854	99.706	100.0	97.5
	Recall (%)	99.708	98.834	100.0	100.0
	No. of False Positives	1	1	0	1
Average	Accuracy (%)	99.226	98.929	99.733	99.019
	Precision (%)	99.703	99.804	99.561	98.241
	Recall (%)	99.311	99.416	100.0	100.0
	No. of False Positives	2	1	0.333	1

5.1.3 Rolling Window Classification

Because the number of false positives was still not 0 or a number close to 0, it was important to try additional reclassification methods in conjunction with the current SVM classifier to reduce this number. The classification values provided when the SVM classifier read the entire dataset as input was written as an “Evaluated Class” column into the dataset. Using these values, a “Rolling Class” was added and built. Initially, the first 5 values in the “Evaluated Class” were grouped and the class value majority was determined, requiring 3 or more instances of class 1 (wireless camera) to be identified as a camera. Whatever

Table 5.4: Summary of metrics for SVM tested against initial dataset using seed values 0, 21, and 42 with 500, 1,000, 5,000, and 10,000 ms bin sizes.

		Bin Sizes			
	Model Metrics	500 ms	1000 ms	5000 ms	10000 ms
Seed = 0	Accuracy (%)	99.082	98.544	99.116	99.110
	Precision (%)	99.645	99.333	99.718	98.365
	Recall (%)	99.190	98.704	98.745	100.0
	No. of False Positives	24	23	2	6
Seed = 21	Accuracy (%)	99.036	98.501	99.116	99.110
	Precision (%)	99.557	99.304	99.718	99.169
	Recall (%)	99.220	98.675	98.745	99.169
	No. of False Positives	30	24	2	3
Seed = 42	Accuracy (%)	99.047	98.629	99.116	99.110
	Precision (%)	99.601	99.420	99.718	98.365
	Recall (%)	99.190	98.732	98.745	100.0
	No. of False Positives	27	20	2	6
Average	Accuracy (%)	99.055	98.588	99.116	99.110
	Precision (%)	99.601	99.352	99.718	99.300
	Recall (%)	99.200	98.704	98.745	99.723
	No. of False Positives	27	22.333	2	5

the class was, this value was assigned as the rolling class value for the fifth data point. The window was then shifted down 1 and re-evaluated the majority for the second through sixth data points, writing the newly determined value in the "Rolling Class" column of the sixth instance. This was repeated through the entirety of the wireless camera portion of the dataset, then restarted for the YouTube video stream section. In almost all combinations of seed value and bin size, the number of false positives dropped to 0, with two instances of 1 false positive being identified, raising the precision to 100% or near 100%. Table 5.5 showcases the results for this round of classification.

Table 5.5: Summary of metrics for rolling window reclassification with window size of 5.

		Bin Sizes			
	Model Metrics	500 ms	1000 ms	5000 ms	10000 ms
Seed = 0	Accuracy (%)	99.406	98.990	99.593	99.848
	Precision (%)	100.0	100.0	100.0	99.718
	Recall (%)	99.248	98.643	99.295	100.0
	No. of False Positives	0	0	0	1
Seed = 21	Accuracy (%)	99.418	98.990	99.593	100.0
	Precision (%)	100.0	100.0	100.0	100.0
	Recall (%)	99.263	98.643	99.295	100.0
	No. of False Positives	0	0	0	0
Seed = 42	Accuracy (%)	99.406	99.033	99.593	99.848
	Precision (%)	100.0	100.0	100.0	99.716
	Recall (%)	99.248	98.701	99.295	100.0
	No. of False Positives	0	0	0	1
Average	Accuracy (%)	99.410	99.004	99.593	99.899
	Precision (%)	100.0	100.0	100.0	99.811
	Recall (%)	99.253	98.662	99.295	100.0
	No. of False Positives	0	0	0	0.667

To observe how varying window sizes impacted the reduction of false positives, additional classification was done using window sizes of 7, 9, and 11. We chose the 10,000 ms bin with a seed value of 0 as it was one of only two combinations that produced a non-zero number of false positives. Table 5.6 presents the accuracy, precision, recall, and number of false positives for each window size.

The rolling window classification test for the bin size-seed combination also included window sizes 13, 15, 17, and 19. However, as shown by Table 5.6, accuracy, precision, and recall reach 100% when the window size was 11, bringing the number of false positives to 0. It is worth noting that the precision and accuracy values remained relatively stable for three

Table 5.6: Summary of accuracy, precision, recall, and number of false positive values when using the rolling window classification on the training dataset classified with a 10,000 ms bin size and seed value of 0. Window sizes used: 7, 9, and 11.

	Window Sizes		
Model Metrics	7	9	11
Accuracy	99.849	99.848	100.0
Precision	99.719	99.718	100.0
Recall	100.0	100.0	100.0
No. of False Positives	1	1	0

different window sizes, highlighting the importance of using large-sized windows in order to account for any misclassified data that may have passed through undetected.

5.2 Testing Dataset

The testing data was collected over a 10 minute period, recording 30,684 packets, 30,241 of which were delivered between the wireless camera and its access point. The packet sniffer only had to monitor the camera's channel once to determine that it was a possible recording device before locking onto its network stream. We wanted to test the strength of the SVM classifier built in Section 5.1.2 for each bin size-combination pair against the test dataset as it was split into 500, 1,000, 5,000, and 10,000 ms bins.

5.2.1 SVM Classifier

Table 5.7 summarizes the average accuracy, precision, recall, and number of false positives for each bin size-seed value pair used to train the SVM classifier, which was then tested on the dataset for all bin sizes. One obvious finding was the dramatic reduction in accuracy and precision as the testing dataset was split using the larger bin sizes and even across each bin as the SVM classifier's bin size increased. A possible explanation for such a result is the different expectations concerning the values for certain fields. When the data was separated into 500 ms bins, we expected the 500 ms classifier to perform well because with the way the packets were grouped, the characteristics of the testing data would be more similar to the characteristics of the training data. This trend, however, was not consistent as the data was split into larger bin sizes. A decrease in data points may have possibly affected the classifier's ability to recognize the features of the data points, especially in the 10,000 ms testing data bins, where network streams preceding the wireless camera had no more than 2 data points. These groupings of all collected packets into these data points may have reflected patterns similar to that of a wireless camera, especially concerning the UL and DL proportions, thereby causing confusion with the classifier.

5.2.2 Rolling Window Classification

As seen by the results of Section 5.2.1, an additional level of classification was necessary to reduce the number of false positives and increase the precision. For this method, we chose the best combination of SVM classifier and testing dataset bin size separation, using the classifier trained by training data grouped into 500 ms bins and split according to the seed value of 0. Choosing 0 out of the three seed values used had no effect on the SVM classifier's performance as all three seed values produced the same results for this combination. We ran the rolling window classification method using window sizes of 3, 5, 7, 9, and 11. If there are too few data points for a given network stream, the majority value served as the class assignment; an even number of camera and non-camera classifications were assigned as a non-camera to reduce the chance of false positives. Table 5.8 shows the results.

As the window size increased, the number of false positives decreased until it fell to almost 0.

Table 5.7: Summary of average SVM classifier metrics across the three different seed values for all bin size designations of the testing dataset.

		Bin Sizes			
	Average Model Metrics	500 ms	1000 ms	5000 ms	10000 ms
500 ms	Accuracy (%)	98.358	97.942	95.953	94.409
	Precision (%)	96.293	92.892	73.239	60.227
	Recall (%)	99.607	99.806	100.0	100.0
	No. of False Positives	39	39.333	38	35
1000 ms	Accuracy (%)	98.472	98.061	99.593	94.728
	Precision (%)	96.568	96.166	74.286	61.628
	Recall (%)	99.607	99.806	100.0	100.0
	No. of False Positives	36	37	36	33
5000 ms	Accuracy (%)	81.092	80.102	79.766	76.997
	Precision (%)	67.354	56.921	35.374	26.904
	Recall (%)	99.607	99.806	100.0	100.0
	No. of False Positives	491	389	190	144
10000 ms	Accuracy (%)	40.349	27.976	12.993	9.531
	Precision (%)	39.440	26.731	11.293	8.559
	Recall (%)	100.0	100.0	100.0	100.0
	No. of False Positives	1561.667	1411.667	817	566.333

Even with a window size of 5, we noticed an improvement in the precision when compared to the SVM classifier, indicating a significant drop in false positives. With a locked-on dataset, the larger window sizes had a greater impact when handling network traffic from an actual camera as there are a greater number of data points. The rolling window method does plateau at 1 false positive. However, secondary checks for the supposed false positive's proportion within the entirety of the dataset can easily remedy this issue, as it will be much less than that of the wireless camera due to the sniffer's locked-on configuration. Another important result of increasing the window size was the decrease in false negatives. As stated previously, false negatives were not as large a concern for this research as false positives, but

Table 5.8: Summary of accuracy, precision, recall, and number of false positive values for rolling window classification using window sizes of 5, 7, 9, 11, 13, 15, 17, and 19.

	Window Sizes							
Model Metrics	5	7	9	11	13	15	17	19
Accuracy	98.975	99.119	99.400	99.563	99.733	99.819	99.908	99.953
Precision	97.961	98.340	98.631	99.017	99.407	99.603	99.801	99.900
Recall	99.605	99.604	100.0	100.0	100.0	100.0	100.0	100.0
No. of False Positives	21	17	14	10	6	4	2	1

reductions in both better fortified the classifier against misclassified data and highlighted proper labels for network streams sooner.

Based on the data in Section 5.2.1, it is unlikely that the rolling window classification would produce as favorable results with the larger bin size combinations due to the much greater number of false positives. Since these values would cluster together for network streams, the rolling window would immediately group and find their majority value to indicate a wireless camera. However, conducting additional tests to gather these results highlighted the best bin size-seed value combination for training the SVM classifier.

Chapter 6

Discussion

While the 10,000 ms bins for the data collections provided the most accurate representation of a wireless camera stream, the reliability of this group size would vary based on the surveying technique used during the sniffing portion of the system. If the sniffer was set so that it would routinely cycle between each channel without noting a stream of interest, it is more likely that one would see a fewer number of maximum length packets. Because of this, one would need to set the channel rotation rate so that the the amount of time spent on a given channel is equal to or greater than the size of the bin that is being used to parse the data and train the model. Otherwise, the bins will contain too few data points and most likely not contain enough packets to capture the maximum length packets. This issue is negligible when the sniffer is set to lock onto a specific stream as an infinite number of packets for the desired stream would be collected, thus increasing the amount of available data. However, we do note the failures of the 10,000 ms bin, both as a grouping for test data and a training size for the SVM classifier. By having so few data points that may vary widely in the I and P frame proportions, the classifier relied heavily on features other than the I and P frame proportions, thus directing the classifier towards the incorrect labels.

When searching for a typical wireless camera stream, the main characteristic the classifier looks for is the presence of possible I and P frames. The additional attributes taken in by the SVM classifier simply support and strengthen the model's parameters when analyzing network packets. A major finding of the project showed that, with proper filtering and postprocessing, a model could be created that would successfully classify a collection of network traffic packets as being from a camera or some other device that is not a camera. While testing against the training set would seemingly be enough to showcase a high resiliency of the SVM classifier against false positives and negatives, the second round of classification mentioned in Section 5.1.2 highlights that the model itself is not perfect, yet still proportionally strong when compared to the initial training. The utilization of the rolling window method for classification further fortifies the model against such errors.

Depending on how the sniffer is set to observe data, the system may suffer when the camera's frame rate reduces. If the frame rate drops to as low as 1 frame per second (FPS), spending 10 seconds on a single channel while completing an overall network traffic survey could easily result in the defining features of wireless camera traffic, namely the consecutive maximum length packets, being overlooked. For such situations, it is imperative that a locked-on sniffer setting is used. One can extend the amount of time spent on a given channel so that the chances of recording multiple maximum length packets in a row increases. While it is possible for the overall survey method to detect these, it is likely that the next time the channel is monitored, the sniffer will miss the necessary packets, therefore reducing the amount of usable data. Because of this, the locked-on method is the superior choice, as it only takes one instance of maximum length packet detection for the sniffer to focus only on the specific stream, thereby guaranteeing further recordings of the possible I and P frames. The amount of time spent on a channel would need to increase to at least 20 seconds in to provide an ample amount of time for detection.

Motion cameras, when recording movement by either an object within the frame or even the camera itself (meaning actual jostling of the camera), should present the same network traffic patterns as wireless cameras that are continuously streaming. When testing with available cameras, a major disadvantage was that the cameras could only be accessed through a mobile device that was connected to the same network, with cloud services requiring additional fees. The network traffic generated through this basic use of the cameras did not reflect the true behavior of a wireless camera. For example, instead of being uploaded to a separate location, shown in the network traffic as an UL from a station to some access point address, the traffic would appear as a DL from an access point to a station, which would be the mobile device. These inconsistencies made it difficult to replicate the typical behavior of a wireless camera set to observe individuals based on their movement. Despite this issue, the system itself could be adapted to recognize such a burst in network traffic, but it would take some time using the sniffer, as well as consistent movement on the user's end, to accurately detect this data.

To detect hidden SSIDs, the ESP32 can be configured as an access point scanner that, when a specific setting is toggled, may uncover a hidden SSID. However, the settings for the access point scanner functionality of an ESP32 differ from the initialization settings that are required to enable wireless traffic sniffing. There is not enough information describing how these two configurations relate to one another. However, it is possible to collect the two types of data (wireless traffic and available access points) separately and compare the MAC addresses to determine if a hidden SSID can be found within the area.

As of now, the current ESP32 model only supports the 2.4 GHZ band for 802.11n, meaning cameras found on channels within the 5 GHz band will not be observed by the wireless traffic sniffer. In addition, while introduced with 802.11n, beamforming Wi-Fi has become a standard process for 802.11ac, although it is not a requirement. Beamforming Wi-Fi focuses signals from a router to specific devices, aiming to strengthen the connections and reduce the wasted, weaker signals that propagate out in all directions. The ESP32 standard does not specifically address beamforming. It is noted that only 802.11b/g/n are supported by the ESP32, so as of now, it can be assumed that beamforming cannot be observed by the chip.

Chapter 7

Limitations

A major characteristic of wireless camera traffic that makes this research possible is its unique network patterns. In general, only wireless cameras will exhibit a patterned distribution of consecutive maximum length packets, simplifying the process of actually identifying the network traffic. However, because of these particular patterns, it is possible to confuse the wireless camera detector by injecting network packets that mimic a camera's behavior. A situation like this could be used by guests who wish to exploit a host and falsely accuse them of installing a network camera within the home. One possible countermeasure for such an act would be to move within the room in which the camera is supposedly placed and monitor for surges in I frames, as standardized, motion-less wireless camera traffic would not reflect the change in the traffic pattern.

As mentioned, the detector is currently designed to detect continuous, live-streamed video traffic. When a wireless camera is consistently recording and uploading video data to a local machine, the network traffic patterns are easier to record and identify. Cameras that record when motion is detected exhibit similar behaviors, just inconsistently. The burst of I and P frames will only be seen when the camera registers movement, and depending on which channel the detector is on at the time of action, the wireless camera traffic may be missed. One way to address this issue would be to have the user spend some amount of time moving around the room in which they believe a camera to be, allowing the detector enough time to cycle through all of the possible channels, increasing the chances of detection.

As mentioned in Section 4.3, we focus primarily on reducing the number of false positives over the number of the false negatives. In general, both values would optimally approach zero to warrant the most accurate system possible. However, reducing the number of false positives takes precedence because of the issues that may be caused by a false accusation. The number of false negatives increases the time taken to detect a camera, but it has a lesser impact on the system's overall reputation. We do observe a drastic reduction in false positives when the bin sized is increased, but using this configuration will have to take the channel rotation times into consideration.

Chapter 8

Literature Review

8.1 Applications of Network Traffic Analysis

Network traffic analysis provides access to network information that vastly improves performance and knowledge of user behaviors. The results of network traffic analysis may be used to recognize network patterns and information about users, raising awareness for abnormal situations. Taylor et al. [37] developed a smartphone application identification system based on side-channel data from application network packets, such as packet size and direction. Using these features, their system could determine what type of application was being accessed, correctly classifying the applications with a 96% accuracy. They also monitored the changes in these features over time across devices and versions, producing favorable results as well. Identifying application features not only enables determining user behavior, but also uncovering possible malware on mobile devices. In [6], the authors compared examples of regular and malware network traffic on Android devices and extracted multiple relevant features, including the average bytes per second and the incoming and outgoing packet ratios. Using a rule-based approach to identify the malware, the authors' system boasted a 90% detection rate.

Recognizing specific devices on a network also assists with diagnosing a network's performance and security. In their work, Shahid et al. [34] addressed the advantages and disadvantages of identifying vulnerable Internet of Thing (IoT) devices on a network. On one hand, by marking which devices open the network up for exposure, network administrators may proactively block network access and reduce the chance of an exploit. However, if an adversary were to have first-hand knowledge of these vulnerable devices, they may utilize their flaws and gain unauthorized access to a network. A survey of network traffic analysis conducted by Conti et al. specifically addresses the dangers of mobile devices as they generate an abundance of data that may expose private information via side-channel analysis [14]. They found that most network traffic analysis works focus on identifying user behavior through applications used, leakage of Personal Identifiable Information (PII), and other privacy-invading purposes. In addition, Android devices presented the greatest vulnerabilities to attacks due to the openness of Android's software.

Previous network traffic analysis has monitored the behaviors of wireless camera traffic and users, relating these patterns to their subscription type. Li et al. conducted an in-depth analysis of the per-user network characteristics for owners of a specific camera brand, identifying patterns indicative of specific usage behaviors [23]. The authors explored a variety of user combinations, monitoring both premium and non-premium camera owners, who may watch any footage through a continuous live-stream, motion detection alerts, or through replay of temporarily stored footage for future observation (only available to premium users). While premium users tended to be very active when accessing and watching videos, 60% of the uploaded replay videos were left unseen, wasting resources and unnecessarily accessing the network. In addition, the cameras were only accessed only from 1 or 2 remote sites, some of which indicated a possible surveillance usage. The authors also found that using changes in the traffic patterns, specifically with any surges in packets or changes in data rate, an adversary could possibly identify the presence and movements of someone being observed by the camera. They continued on to say that routine behaviors, past, present, and future, may even be detectable by an adversary.

8.2 Wireless Camera Detection

A variety of techniques have been applied to detect wireless cameras within an environment. Utilizing network traffic and comparing packet features to the expected behaviors of wireless camera traffic has shown to be a reliable form of detection. In [11], the authors made use of the packet length distribution (PLD), recording both its pattern and stability, as well as the stability of the bandwidth to help identify wireless cameras, marking changes in packet length to classify I, P, and B frames. Camera localization was possible and required user-assistance through movement in a room, monitoring the spike in the number of packets being sent to accommodate the changing imagery. All detection was done through a mobile application with an accuracy of 99% within 2.7 seconds. Instead of explicitly extracting streaming characteristics, Wu et al. compared the streams of network-enabled recording devices with the streams of a smartphone [39]. Using the collected data, the authors compared timing patterns, as well as the general similarities in data packets. Without a neural network model to assist with classification and instead relying on thresholds, an accuracy of 87% was achieved. However, the introduction of the neural network increased the accuracy to 97% and boasted promising results for both indoor and outdoor scenarios.

Monitoring features outside of a network stream have been shown to produce favorable results. In [25], two heavily user-driven approaches were introduced: Blink and Flicker. Blink, a mobile application, requires the individual to turn the lights off and on in a room containing a camera, during which the application will make use of the mobile phone's light sensor and wireless radio to observe the response of the wireless camera. Flicker developed Blink one step further by using a portable circuit to stimulate a camera with human-invisible flickering LEDs. Flicker worked in scenarios whether cameras were live streaming or not. Blink typically performed well in daytime-scenarios over multiple data collections, while

Flicker reached detection rates of 99% over various network settings, but did initially suffer when the network was congested. Radio Frequency (RF) waves given off by a wireless camera may also be observed for identification. Karthikeyan and Sasirekha designed a system that utilized an RF antenna to observe the RF waves given off by the hidden camera, alerting to the presence through a mobile notification [29]. A separate bug detector identified the location of the camera, which was a major contribution the authors offered as an improvement upon previous localization methods.

In [21], wireless camera detection was not the sole purpose of the experiment; detection was only a stepping stone towards identifying human presence within a home. The first part of their system detected the presence of the camera by monitoring I, P, and B frames and variations in bit rate, similar to [11]. Then, they used the fluctuation in bit rate as an indication of some human presence within the environment. Not only were the authors able to determine if an individual was present, but based on the collection of previous data, they built Activity of Daily Living (ADL) profiles for the residents. They achieved a recall of 98.3% when detecting the camera and successfully observed a human presence 97.2% of the time.

Chapter 9

Conclusions and Future Work

We designed and implemented a wireless camera detector that can determine the presence or absence of a hidden wireless camera within a given area. Because of the wireless camera's unique network traffic characteristics, the detector can easily and quickly identify a camera's network packets patterns within a network traffic stream. However, the variability of these patterns may result in false positives when engineered traffic is injected into a network. These issues are limited to those wishing to exploit individuals for an invasion of privacy and may be countered through further testing.

We see that an SVM classifier used as the first level of identification produced strong accuracy, precision, and recall values, all calculated at or above 97%. However, with intentions to reduce the number of false positives and raise the precision to 100%, a secondary level of classification, the rolling window classification, was necessary. The rolling window method greatly reduced the false positive value as the window size was increased, which also resulted in a decreased number of false negatives.

While the system itself effectively detects a continuously streaming wireless camera, improvements may be made to the system to expand its scope. While multiple ESP32 development boards may be distributed to collect network traffic data, a separate machine is needed to process the data. In the future, we would like to consolidate the system so that data collection, preprocessing, and analysis can be done on a single development board. In addition, we would like to enable camera localization, preferably without user-assistance. However, having the user move throughout the home to trigger surges in camera network traffic and detecting these surges is currently possible. With regards to motion, adapting the system to recognize motion-detection cameras that make use of mobile devices when accessing the live feed would develop the system's detection range. When a mobile device is used, the network traffic characteristics vary slightly because of its inconsistency and the modified direction of the packets. Training the system to recognize such features would allow for detection of both live and non-live wireless camera streams.

Additional improvements could be made that make the system more than just a wireless camera detector. Using the features indicating a wireless camera, we would like to monitor surges in I and P frames to determine if someone is present in a room with a wireless camera. While this system is intended for detecting hidden cameras, it can be used by those who installed the cameras to determine if unexpected individuals are within the vicinity without having to access the camera stream. Another possible development would be to assist those whose privacy is being invaded by implementing camera jamming, either by flooding the camera or access point to which it is connected with an excess of packets, performing a Denial of Service (DoS) attack. Because the system accesses the MAC addresses for the camera and its network, packets may be formed using the correct address settings.

Bibliography

[1] An explanation of video compression techniques. Technical report, Axis Communications, 2008. URL https://www.accentalarms.com/specsheets/axis/_wp_videocompression.pdf.

[2] An explanation of video compression techniques. *Axis Communications*, pages 1–16, 2008 (Accessed September 16, 2020).

[3] *CCTV Technology Handbook*. U.S. Department of Homeland Security (DHS), 2013.

[4] Smart home security cameras market size, share and trends analysis report by product (wired, wireless), by application (doorbell camera, indoor camera, outdoor camera), by region, and segment forecasts, 2020 - 2027, May 2020 (Accessed September 16, 2020). URL https://www.grandviewresearch.com/industry-analysis/smart-home-security-camera-market.

[5] *What are Airbnb's rules about security cameras and other recording devices in listings?* Airbnb, (Accessed on September 11,2020). URL https://www.airbnb.com/help/article/887/what-are-airbnbs-rules-about-security-cameras-and-other-recording-devices-in-listings.

[6] A. Arora, S. Garg, and S. K. Peddoju. Malware detection using network traffic analysis in android based mobile devices. In *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, pages 66–71, 2014.

[7] P. Asrodia and H. Patel. Network traffic analysis using packet sniffer. *International Journal of Engineering Research and Applications (IJERA)*, 2(3):854–856, May 2012.

[8] Asa Ben-Hur and Jason Weston. A user's guide to support vector machines. *Methods in molecular biology (Clifton, N.J.)*, 609:223–39, 01 2010.

[9] R. Braden. Requirements for Internet Hosts – Communication Layers. RFC 1122, RFC Editor, Oct 1989. URL https://tools.ietf.org/pdf/rfc1122.pdf. (Accessed on September 16, 2020).

[10] Alisha Cecil. A summary of network traffic monitoring and analysis techniques, (Accessed September 11, 2020). URL https://www.cse.wustl.edu/~jain/cse567-06/ftp/net_monitoring.pdf.

[11] Y. Cheng, X. Ji, T. Lu, and W. Xu. On detecting hidden wireless cameras: A traffic pattern-based approach. *IEEE Transactions on Mobile Computing*, 19(4):907–921, 2020.

[12] *Fundamentals of Digital Video.* Cisco, 2008 (Accessed on September 16, 2020). URL https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Video/pktvideoaag.html.

[13] Charlie Cöe. *Horrified tourist finds pervert's secret camera hidden inside a water bottle behind her on the beach - and trawls his footage of scantily-clad women and children to find out who he is.* Daily Mail, 2019 (accessed September 11, 2020). URL https://www.msn.com/en-au/news/australia/horrified-tourist-finds-perverts-secret-camera-hidden-inside-a-water-bottle-behind-her-on-the-beach-and-trawls-his-footage-of-scantily-clad-women-and-children-to-find-out-who-he-is/ar-AAIYrlp.

[14] M. Conti, Q. Q. Li, A. Maragno, and R. Spolaor. The dark side(-channel) of mobile

devices: A survey on network traffic analysis. *IEEE Communications Surveys Tutorials*, 20(4):2658–2713, 2018.

[15] Emily Dixon. *Family finds hidden camera livestreaming from their Airbnb in Ireland*. CNN, 2019 (accessed September 10, 2020). URL https://www.cnn.com/2019/04/05/europe/ireland-airbnb-hidden-camera-scli-intl/index.html.

[16] *ESP32 Technical Reference Manual*. Espressif, 2020 (Accessed on September 16, 2020). URL https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.

[17] H.264 (06/2019). ITU-T H.264: Advanced video coding for generic audiovisual services. Standard, International Telecommunication Union, June 2019.

[18] *Hi3516C Professsional HD IP Camera SoC Brief Data Sheet*. Hisilicon, 2012 ((Accessed on September 16, 2020). URL https://www.burglaryalarmsystem.com/pdf/Hi3516C.pdf.

[19] IEEE Std 802.11-2016. IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Standard, IEEE, 2016.

[20] ISO/IEC 7498-1:1994. Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model. Standard, International Organization for Standardization, November 1994.

[21] Xiaoyu Ji, Yushi Cheng, Wenyuan Xu, and Xinyan Zhou. User presence inference via encrypted traffic of wireless camera in smart homes. *Security and Communication Networks*, 2018:1–10, 09 2018.

[22] R. Kimmel. Demosaicing: image reconstruction from color ccd samples. *IEEE Transactions on Image Processing*, 8(9):1221–1228, 1999.

[23] J. Li, Z. Li, G. Tyson, and G. Xie. Your privilege gives your privacy away: An analysis of a home security camera service. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 387–396, 2020.

[24] David Lima, Eliana Almeida, and Andre Aquino. Evaluation of parking space detection systems using wireless cameras. In *Anais do VII Simpósio Brasileiro de Computação Ubíqua e Pervasiva*, pages 181–190, Porto Alegre, RS, Brasil, 2015. SBC.

[25] Tian Liu, Ziyu Liu, Jun Huang, Rui Tan, and Zhen Tan. Detecting wireless spy cameras via stimulating and probing. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '18, page 243–255, New York, NY, USA, 2018. Association for Computing Machinery.

[26] D. Mistry, P. Modi, K. Deokule, A. Patel, H. Patki, and O. Abuzaghleh. Network traffic measurement and analysis. In *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pages 1–7, 2016.

[27] Davide Moroni, Gabriele Pieri, Giuseppe Riccardo Leone, and Marco Tampucci. Smart cities monitoring through wireless smart cameras. In *Proceedings of the 2nd International Conference on Applications of Intelligent Systems*, APPIS '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450360852.

[28] Łukasz Podkalicki. *ESP32 – WiFi sniffer*, 2017 (Accessed September 16, 2020). URL https://blog.podkalicki.com/esp32-wifi-sniffer/.

[29] K. Sasirekha R. Karthikeyan. Identification of hidden camera using mobile rf signal. *International Journal of Pure and Applied Mathematics*, 166(16):1915–1918, 2018.

[30] Alexis Rivas. *Airbnb Couple Finds Hidden Cameras in Bathroom, Bedroom: Lawsuit.* NBC San Diego, 2019 (accessed September 10, 2020). URL https://www.nbcsandiego.com/news/local/airbnb-guests-find-hidden-cameras-in-bathroom-bedroom-lawsuit/2109603/.

[31] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, RFC Editor, July 2003. URL https://tools.ietf.org/pdf/rfc3550.pdf. (Accessed on September 16, 2020).

[32] H. Schulzrinne, A. Rao, R. Lanphier, and M. Westerlund. Real-Time Streaming Protocol Version 2.0. RFC 7826, RFC Editor, Dec 2016. URL https://tools.ietf.org/pdf/rfc7826.pdf. (Accessed on September 16, 2020).

[33] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h.264/avc standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, 2007.

[34] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar. Iot devices recognition through network traffic analysis. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 5187–5192, 2018.

[35] Samantha Smith. *Virginia Tech freshman arrested after investigation into hidden bathroom cameras on campus.* WSLS 10, 2019 (accessed September 11, 2020). URL https://www.wsls.com/news/2019/02/28/virginia-tech-freshman-arrested-after-investigation-into-hidden-bathroom-cameras-on-campus/.

[36] *Airbnb host busted secretly filming guests showering.* Sunshine Coast Daily, 2018 (accessed September 10, 2020). URL https://www.sunshinecoastdaily.com.au/news/airbnb-host-busted-secretly-filming-guests-showeri/3563702/.

[37] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security*, 13(1):63–78, 2018.

[38] Y.-K. Wang, R. Even, T. Kristensen, and R. Jesup. RTP Payload Format for H.264 Video. RFC 6184, RFC Editor, May 2011. URL https://tools.ietf.org/pdf/rfc6184.pdf. (Accessed on September 16, 2020).

[39] K. Wu and B. Lagesse. Do you see what I see? Detecting hidden streaming cameras through similarity of simultaneous observation. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10, 2019.