A Quantitative Study of DDoS and E-DDoS Attacks on WiFi Smart Home Devices

Bhagyashri Tushir*, Yogesh Dalal†, Behnam Dezfouli*, Yuhong Liu*
*Internet of Things Research Lab, Department of Computer Science and Engineering, Santa Clara University, USA
†Apple Corporation, Cupertino, USA
{btushir, bdezfouli, yhliu}@scu.edu ydalal@apple.com

***Abstract*—Internet of Things (IoT) has facilitated the prosperity of smart environments such as smart homes. Meanwhile, WiFi is a broadly-used technology for the wireless connectivity of IoT devices. However, smart home IoT devices are often vulnerable to various security attacks. This paper quantifies the impact of Distributed Denial of Service (DDoS) and energy-oriented DDoS attacks (E-DDoS) on WiFi smart home devices and explores the underlying reasons from the perspective of attacker, victim device, and access point (AP). Compared to the existing work, which primarily focus on DDoS attacks launched by compromised IoT devices against servers, our work focuses on the connectivity and energy consumption of IoT devices when under attack. Our key findings are three-fold. First, the minimum DDoS attack rate causing service disruptions varies significantly among different IoT smart home devices, and buffer overflow within the victim device is validated as critical. Second, the group key updating process of WiFi, may facilitate DDoS attacks by causing faster victim disconnections. Third, a higher E-DDoS attack rate sent by the attacker may not necessarily lead to a victim's higher energy consumption. Our study reveals the communication protocols, attack rates, payload sizes, and victim devices' ports state as the vital factors to determine the energy consumption of victim devices. These findings facilitate a thorough understanding of IoT devices' potential vulnerabilities within a smart home environment and pave solid foundations for future studies on defense solutions.**

***Index Terms*—Internet of Things (IoT), security, energy consumption, association, disconnection.**

I. Introduction

Internet of Things (IoT) systems facilitate user interactions and/or monitor and control physical environments by sensing, communicating, and analyzing the data collected via sensory devices. The application domains of IoT includes social robots [1], [2], intelligent transportation services [3], [4] and battlefield environments [5]. One important IoT application that influences everyone's daily life is smart home [6]. With the advancement of IoT technologies, smart home systems and their adaptation have significantly increased, providing groundbreaking services such as voice assistants, security cameras, smart entertainment systems, home appliances, and many more. Meanwhile, due to its wide deployment and low cost, WiFi is a broadly used technology for the wireless connectivity of IoT devices [7].

Despite the significant increase of smart home devices [8], these devices show vulnerabilities to diverse security and privacy attacks, such as *Distributed Denial of Service* attacks (DDoS), *energy-oriented DDoS* attacks (E-DDoS), harvesting and forging data, blackmail/extortion, bitcoin mining, stalking, or robbery [9]. The reasons are multi-fold. First, because of competition and revenue gain, many manufactures disregard the security aspects of IoT devices, which demands resources and skills—adding to the cost. Second, consumers are usually not well educated about the potential security issues caused by arbitrarily adding IoT devices into their home networks, not to mention the importance of keeping their IoT devices' security features up-to-date. Third, smart home devices usually share a WiFi Access Point (AP) for Internet access and local interconnection. For example, Google Nest and Google Home can control smart thermostats, lights, cameras, and home appliances, and provide feedback to users' smartphones. The high interconnectivity among smart home devices makes it much easier for attackers to compromise one device to get hold of the other devices and launch various attacks against smart homes [9]. For example, [10] illustrates that IoT devices behind NAT/firewall-enabled home APs are no longer safe. Malicious attackers can inject viruses into smart home devices and create a large-scale botnet, severely threatening everything connected to the Internet.

Among various types of attacks, DDoS and E-DDoS are particularly effective in *service disruption* and *increasing users' electricity cost*. These attacks are often initiated by a cluster of compromised devices—known as a *botnet*. While DDoS attacks launch malicious traffic to exhaust the target device's resources, E-DDOS attacks aim to cost maximum energy consumption on the target side through malicious traffic. For example, high-profile cloud services have become a popular target of DDoS and E-DDoS [11]–[13] attacks launched by compromised IoT devices, leading to a dramatic increase of data centers' service disruptions and energy consumption.

Despite the well-recognized damage of DDoS and E-DDoS attacks on data centers, their impact on smart homes is ignored by existing studies. Compared to data centers, individual household owners are much more vulnerable and sensitive to their energy bills. More importantly, considering the massive amount of smart home devices on the market, the aggregated attack impact cannot be neglected. According to [14], the number of IoT cameras and smart appliances in use by 2020 is around 1 billion and 5 billion, respectively. *Our study reveals that when these devices are under E-DDoS attacks for one month, the approximate increase in the electricity bills can easily reach $253.7 million*. On the other hand, DDoS attacks against these devices can disrupt their services by

This research was fully performed at the Internet of Things Research Lab, Santa Clara University.

disconnecting them from the AP. Thus, well-planned attacks can significantly damage the confidence of the general public in adopting household IoT devices to address their needs, and eventually, causes revenue loss and hurts the market. Therefore, it is essential to better understand the impact of DDoS and E-DDoS attacks on smart home devices.

In contrast to the existing work, in this paper, we focus on the direct effects of these attacks on the IoT devices. In particular, instead of focusing on specific devices, user application, or software stack, we study DDoS and E-DDoS attacks from the WiFi connectivity and energy consumption perspectives. Therefore, the target space is the highest for such attacks. To the best of our knowledge, *this is the first work that empirically measures the impact of these attacks on household IoT devices*. The results show the significant damage possibly caused by these attacks and draw attention to the urgent needs of effective defense solutions. Our research can be used as a framework to test IoT devices' security and can be applied to create security standards for robust, predictable, and tamper-free operations.

The main contributions of this paper are summarized as follows: First, we design a smart home testbed to automate the different attack factors and reliably capture network traffic and victim IoT devices' power consumption in real-time. *Second*, we quantitatively study the impact of DDoS attacks on victim devices' service disruptions by identifying the minimum attack rates and duration. By studying a victim device's internal status, buffer overflow has been validated as the key reason for device disconnections. Furthermore, we identify that the WiFi protected access (WPA) group temporal key (GTK) updating process, which is a security feature, can facilitate DDoS attacks to disconnect victim devices from their associated AP. Its quantitative impact on the victim's disconnection is also studied in detail. *Third*, we identify several critical influential factors, particularly in terms of communication protocol, attack rate, payload size, and victim devices' port state, and analyze these factors considering their impact on the victim devices' energy consumption. *Fourth*, based on the above observations and studies, we summarize effective ways to launch DDoS and E-DDoS attacks against smart home devices.

The rest of the paper is organized as follows: We present the attack scenarios, testbed setup, key influential factors, and automated data collection process in Section II. Section III presents the results of the network scan, impact of DDoS attacks on device disconnection, and impact of E-DDoS attacks on energy consumption. We overview related work in Section IV and conclude the paper in V.

II. Experiment Setup and Design

In this section, we first present possible attack scenarios against smart home. Then, we explain the detail of our testbed design, different attack factors, and the automated data collection process.

A. *Experiment Setup*

There are two possible scenarios when launching attacks on a smart home: *internal* and *external*. In the internal scenario, the attacker has access to the local network. This is possible by hacking the WiFi network or gaining access to a device of the smart home. For example, the attacker may launch internal attacks by remotely accessing a local Linux-based device such as Amazon Echo or Google Home. For the external case, the attacker generates attack packets from outside of the network. For example, if the attacker can identify port mapping inside the home router's NAT table, it can send packets to particular devices from outside of the local network. Both of these scenarios have been studied in literature [15].

The testbed contains various household consumer IoT devices as victim. We chose the most popular smart home devices on the market as victim devices, including Google Mini Home (Google Home), Amazon Echo Dot (Alexa), Nest Indoor Camera (NestCam), and Ring Stick Up Camera (RingCam). Besides, since accessing the internals (e.g., perceived RSSI value, buffer overflow, and perceived noise) of these devices is not possible, we also include an IoT edge device as another victim device to understand the internal status of a victim device when it is under attack. This IoT device is CYW43907 [16], [17], henceforth as the *DevBoard* (development board), an embedded wireless system-on-a-chip (SoC) featuring two ARM Cortex-R4 processors, one used for managing the WiFi subsystem and one used for running applications. Moreover, four separate Linux devices are involved: (i) an attacker to send malicious packets to the victim devices, (ii) an AP, (iii) a sniffer to capture WiFi traffic, and (iv) a command and control center (C&C) to coordinate the testbed components. Further, we use a programmable power switch to turn on/off the connected devices automatically. To measure each victim device's energy consumption in real-time, we use the EMPIOT board developed in our lab [18]. EMPIOT is a low-cost, easy-to-build, and flexible energy measurement platform. This platform samples voltage and current at approximately 500,000 samples per second. These samples are then averaged and streamed at 1 Ksps. The current and voltage resolution of this platform are 100 μA and 4 mV, respectively.

We use various software tools for attack data generation and collection. On the attacker device, we use `nmap` to launch a network scan and identify the victim device's status (e.g., online/offline, MAC address, IP address). We also run TCP/UDP port scan to identify ports' status on the victim devices. We use the `hping3` tool to generate dynamic DDoS and E-DDoS attack traffic by adjusting attack rate, source IP address, destination IP address, payload, attack type, flags of TCP session (SYN, ACK, FIN, push, or urgent) and port types. Second, for complete control over the AP, for example, to monitor the 802.11 events, we install `hostapd` on the AP. `hostapd` is a user-space daemon that provides AP functionality on Linux-based machines [7]. To quantitatively evaluate the impact of the DDoS/E-DDoS attacks, we use `tshark`, running on a sniffer, to capture WiFi traffic. For the control purposes, we use `Paramiko`, a parallel SSH Python library. With this library, the C&C uses SSH to connect to EMPIOT, attacker, AP, and the sniffer to synchronize attack generation and data collection. Last, we developed Python scripts on the C&C for the programmable power switch to automatically

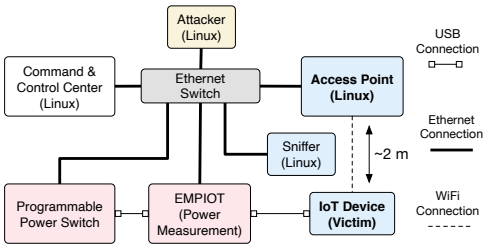

Fig. 1. Interconnections of the testbed components to automate attack generation and data collection.

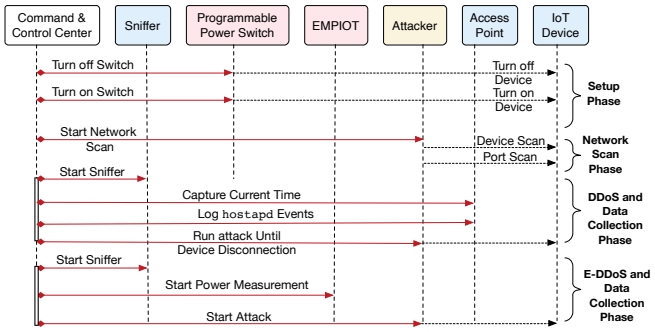

Fig. 2. Interactions between the components of the testbed. Each trial contains three phases: *setup*, *network scan*, and *attack and data collection.*

restart the connected IoT device after each experiment.

Figure 1 shows the connectivity of testbed components. The AP, C&C, programmable power switch, EMPIOT, sniffer, and attacker connect to an Ethernet switch. The victim device connects to the AP through WiFi and is placed close (about 2 m) to the AP to avoid link unreliability. The sniffer is placed between the AP and victim to sniff all the packets exchanged. The victim device's input USB port is plugged into the EMPIOT's output USB port for energy measurement. The EMPIOT's input is connected to the programmable power switch via USB, allowing automatic restarts after each attack. To study the impact of attacks on each victim device independently, we perform experiments on a device-by-device basis. During attacks, we make sure that the victim device is in an idle state so that the energy consumption and disconnections are caused only by the attack traffic. This allows us to differentiate between the energy caused by device operation (e.g., video streaming) and attack.

B. Identify Key Influential Factors

In this work, we mainly focus on the impact of the following factors on disconnection and energy consumption.

Firstly, different types of victim devices differ in their hardware design (e.g., CPU, memory, WiFi chip) and available services (e.g., voice assistants, video cameras), thereby they respond to the same attack in different ways. In this work, we use Google Home and Alexa as the WiFi voice assistant devices, and NestCam and RingCam as the WiFi video cameras.

Secondly, during an attack, the communication protocol and port state (e.g., open, closed) can significantly influence a victim device's response, leading to different service disruptions or energy consumption levels. Therefore, we launch three types of attacks: (i) ICMP Echo Request (ICMP), (ii) TCP-SYN, and (iii) UDP attacks. In the case of TCP-SYN and UDP attacks, we launch attacks against ports with different states—open, closed, filtered, and open/filtered. Note that ICMP attacks do not specify the port number.

Thirdly, although it has been recognized that the impact of DDoS/E-DDoS attacks is directly related to attack rates, there is a lack of quantitative analysis on the correlation between attack rates and the resulted service disruptions and energy consumption of victim devices. For example, what is the minimum attack rate causing device disconnections? Will energy consumption linearly increase versus attack rates? How does such a relationship change across different victim devices? We test different attack rates on each device to answer these questions.

Fourthly, as IoT devices are resource-constrained in terms of processing power and memory, their behaviors may differ significantly versus the payload size of attack packets. Therefore, we also adjust the payload of attack packets to 0 Byte, denoted as P_L (no-payload), and 1400 Byte, denoted as P_H (high-payload).

C. Automated Data Collection

We developed an automated testbed due to the following reasons: Firstly, a reliable evaluation of the key influential factors requires a significant amount of repetitive experiments over a long duration. For example, we need to run 735 trials for E-DDoS attacks where each trial lasts 3 minutes, and 640 trials for DDoS attacks where each trial runs until the victim device is disconnected. Manual initialization and control of these experiments will significantly delay the data collection process. Secondly, highly accurate coordination and synchronization among all the testbed components are critical, making manual control impossible. Thirdly, we need to frequently switch to a different attack configuration. To ensure that we start on a clean slate, the testbed needs to be restarted after each attack trial. A manual restart of all the testing devices is tedious and error-prone. Besides, we observe that many IoT devices change their port state after each reboot, requiring us to identify the attacking ports for each trial automatically.

Figure 2 presents the interactions among the testbed's components. Note that although we draw both DDoS and E-DDoS attacks in the same figure for simplicity reasons, these two types of attacks are conducted in separate sets of experiments. Each trial contains three phases: *setup*, *network scan*, and *attack and data collection*. The double solid lines in Figure 2 represent multiple operations running simultaneously.

During the *setup* phase, C&C restarts the victim device to avoid any sort of influence from the previous trials. Using a Python program, the C&C turns off a particular port of the programmable power switch to shut down the victim device. After waiting for five seconds, C&C turns on the programmable power switch's port to start the victim device, and waits for one minute to ensure that the device is completely restarted and stabilized. Next, the C&C sends a command to the attacker

TABLE I
RESULTS OF NETWORK SCAN

Device	TCP Scan	UDP Scan
Alexa	4 open, 49144 filtered and 16387 closed ports	1 open, 622 open/filtered and 377 closed port
Google Home	6 open and 65529 closed ports	986 closed and 14 open/filtered ports
NestCam	65535 closed ports	1000 closed ports
RingCam	65535 closed ports	1000 closed ports

to initiate the *network scan* phase. Using `nmap`, the attacker performs a device scan and a port scan.

The third phase is *attack and data collection*. To collect data during DDoS attacks, the C&C simultaneously starts the sniffer, connects to the AP to capture its current time, and starts the attack and logs 802.11 events. Specifically, two files are generated for each trial, one at the sniffer side containing 802.11 management packets, and the other at the AP side containing the attack start time and 802.11 events generated by `hostapd`.

In contrast, for E-DDoS attacks, the C&C simultaneously launches commands for EMPIOT, sniffer, and the attacker. Two files are generated for each experiment trial, one EMPIOT file containing the energy consumption data and one `tshark` file containing all the packets exchanged by the victim device. After each trial, the C&C reconfigures the attack parameters.

III. Experimental Results and Analysis

A. *Results of Network Scans*

Device scan gathers information about the victim IoT device, such as the device state (online/offline), IP address, and MAC address. The goal of the port scan is to discover the state of TCP and UDP ports. The possible states are: open, closed, filtered, and open/filtered. When E-DDoS attacks are launched against different ports, depending on their state, the energy consumption could vary. An attacker can successfully identify all active IoT devices in the network and retrieve the state of TCP/UDP ports for each device. Table I reports the state of ports for the devices used in our testbed.

B. *Attack Rates Received by Victim*

In preliminary tests, we find that victim devices seldom receive the actual attack rate sent by the attacker. For example, for TCP-SYN attack on a closed port, Figure 3 shows the average total attack rate received by Google Home versus the average attack rate sent by the attacker, averaged for five repetitions. We can observe that with an increase in the packet rate sent by the attacker, the received packet rate shows an approximately logarithmic increase. Similar patterns are observed for Alexa, RingCam, and NestCam (results not included due to space limitation).

To figure out the reason, we analyzed the `tshark` files collected by the sniffer and observed a significant packet retransmissions rate from the AP to the victim IoT device, as Figure 3 shows. This is because the reception buffer of the 802.11 transceiver in the IoT device is overflowed and

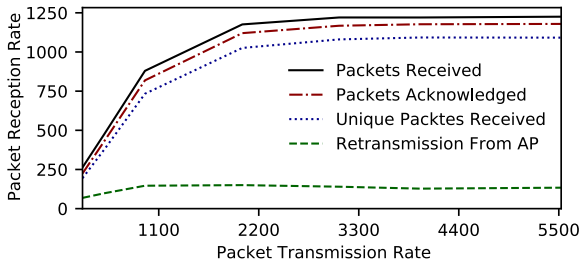

Fig. 3. TCP-SYN E-DDoS attacks on a closed port of Google Home. Packet transmission rate and reception rate are over 500 ms intervals. These results confirm that the packet rate received by the victim (Y axis) does not necessarily follow the attack rate sent by the attacker (X axis).

cannot accept all the incoming packets. Therefore, it cannot acknowledge all the received packets when under attack. As a consequence, the AP retransmits packets until the retry limit (usually 7) is reached, and then removes the packet from the queue allocated to the station in the WiFi transceiver's driver. Meanwhile, as it takes a certain amount of memory and computation time to store and retransmit packets, after some time, buffer overflow happens at the AP, leading to attack packets dropping by the AP. In particular, since the output rate of the AP's wireless interface is lower than the input rate of the wired interface, packet drop happens in the AP's queuing discipline (qdisc) layer and the wireless driver's queues.

From the above analysis, we can see that higher attack rates sent by the attacker, which usually costs more resources at the attacker side, may not necessarily lead to compelling increases in the attack rates received by the victim. Consequently, we will mainly consider the *received attack rate* in this paper.

C. *Quantitative Understanding of DDoS Attacks*

In this section, we study the effect of DDoS attacks and WPA GTK on the disconnection of victim devices from the AP.

1) ***Disconnections caused by DDoS attacks*****:** We consider *threshold attack rate* as the minimum attack rate measured as packet per second (pps) that results in disconnecting the victim device from the AP. The time duration between the initiation of an attack until device disconnection caused by the attack is referred to as the *survival duration*. We set the maximum attack duration to 10 minutes. We have launched ICMP attacks and TCP-SYN/UDP attacks on open, filtered, and closed ports of the victim devices to collect their threshold attack rate and survival duration. Table II presents the results for three types of attacks: ICMP, TCP-SYN on closed port, and UDP on closed port. We use these attack types because: (i) RingCam and NestCam only have closed ports, and (ii) Google Home and Alexa are not disconnected by TCP-SYN/UDP attacks on open or filtered ports even with the maximum receivable attack rates.

As Table II shows, Alexa does not disconnect under ICMP attacks. Google Home and Alexa are disconnected by all the P_L attacks, while Alexa survives a slightly higher attack rate. By looking into the `tshark` files, we observe the 802.11

TABLE II
DDoS THRESHOLD ATTACK RATES AND SURVIVAL DURATION

	Google Home		Alexa		NestCam		RingCam	
	P_L	P_H	P_L	P_H	P_L	P_H	P_L	P_H
Rate	15 k	-	20 k	-	800	100	16 k	800
T_1	7.4	-	-	-	3.6	3.15	6.8	6.19
T_2	8.6	-	6.54	-	4.4	2.45	6.2	7.2
T_3	7.8	-	6.31	-	3.8	2.45	7.1	7.6

P_L and P_H refer to 0 and 1400 Byte attack payloads.
Rate refers to the attack packets per second sent by the attacker.
T refers to the average survival duration (in minutes).
T_1: ICMP, T_2: TCP-SYN closed port, T_3: UDP closed port

handoff process performed by both devices. Specifically, during attack, Google Home and Alexa broadcast probe requests and send deauthentication packets to the AP. The reason code field of the deauthentication packets is 3, suggesting that the victim device wants to leave the current network [19]. On the other hand, in the case of P_H attacks, no such packets are present, and Google Home and Alexa do not disconnect even with the highest attack rate. We will present a detailed study of these observations in the next section. Compared to Google Home and Alexa, RingCam disconnects in all the cases, and disassociation packets with reason code 3 are also observed in all the cases. It is also worth mentioning that Alexa and Google Home include more powerful hardware compared to RingCam and NestCam. Alexa uses a quad-core ARM-Cortex A53 application processor and Google Home uses a dual-core ARM Cortex A7 application processor, and therefore both systems are capable of running the Linux operating system. In contrast, for example, RingCam uses a TI CC3220 SoC along with a Camera Video Processor.

NestCam disconnects in all the cases. With the P_H, however, NestCam requires a lower threshold attack rate and demonstrates a slightly shorter survival duration. For all the attack types and payload sizes, the NestCam sends a disassociation packet with reason code 7 [19] to the AP, indicating that it is no longer associated with the AP and therefore no more packets are expected from the AP. Two potential reasons are: either NestCam does not send deauthentication packets to the AP while leaving the network, or the sniffer fails to capture the deauthentication packets. To identify the reason, we stopped the attack and forced NestCam to leave the network by turning off the AP. Even in this case, no deauthentication packet was captured by the sniffer. Therefore, we conclude that NestCam does not send notifications to the AP when leaving the network.

*2) **Analysis from inside of victim devices*:** Since it is not possible to access the internals of commercial IoT devices, we adopt the DevBoard (discussed in Section II-A) to analyze and reveal the internal status of the victim device during DDoS attacks. In our preliminary test in Section II-B, we mentioned that buffer overflow occurs in the victim device. Meanwhile, since the 802.11 handoff mechanism is usually triggered due to beacon loss and RSSI drop [20], [21], we collect these two parameters along with buffer overflow. We use low-level APIs to interact with the driver and collect the above parameters during system operation. The attack type used is TCP-SYN on closed port. During the experiment, we consecutively switch among three 2-minute attack scenarios: (i) attack with P_L, (ii) attack with P_H, and (iii) very low-rate attack that does not cause buffer overflow (the payload size is 0 Bytes). In between two consecutive scenarios, we also add a 2-minute idle (no attack) interval. The overall experiment lasts for 32 minutes. We also implement two different workloads on the victim device's host processor: (i) a single idle thread, and (ii) a single busy thread that is continuously performing mathematical operations. Figure 4 presents the results collected. The solid black line shows the results when the idle thread is running, and the dash red line shows the results when the busy thread is running.

Figure 4 shows that buffer overflow during P_L intervals is considerably higher compared to the P_H intervals. Although the second row of Figure 4 clearly shows buffer overflow instances, we also collect noise level to verify that packet losses were not caused by low signal to interference-and-noise ratio (SINR) [22]. As the fourth row of this figure shows, there are no evident noise changes during the P_L attack periods.

A buffer overflow indicates the reception of a new packet before processing the previously-received packet. When a packet is received by the transceiver, it generates an interrupt to inform the host processor and request for packet transfer from the transceiver to the operating system's domain. If the request is not handled promptly, the transceiver may receive a new packet while the previously received packet has not been processed yet—causing a buffer overflow. Both incoming packet rate and the load of the host processor intensify buffer overflow. This can be observed by the dash red lines in Figure 4, which correspond to the case where a higher load is imposed on the processor.

While data packets are usually retransmitted in case the transmitter does not receive an acknowledgment or a proper response, broadcast packets are sent just once and their loss adversely affects the operation of victim device. Most importantly, 802.11 devices need to receive beacon packets every 102.4 ms to synchronize their timer with the AP, identify if the AP has any buffered packets for them, and decide for roaming, just to mention a few [23]. The 802.11 drivers usually adopt a moving averaging method to compute their link quality with the AP. In particular, whenever a beacon packet is received, the RSSI of the packet is fed into the averaging method. If no beacon is received when it is expected (102.4 ms after the last beacon), a small RSSI value such as the noise floor [22] is added to the averaging method. Therefore, even if the victim device is very close to the AP, frequent loss of beacons results in a lower RSSI value perceived by the victim over time. And usually, when the RSSI drops below a certain threshold (e.g., -70 dBm), the victim device initiates the roaming process.

Figure 4 also shows that buffer overflow happens in all the P_H scenarios, which also leads to beacon loss and RSSI drop. This explains why NestCam and RingCam are disconnected when they are under the P_H attacks. However, the number of buffer overflows under the P_H scenario is not as high as that of the P_L scenario. This is because it takes a longer time for the P_H attack packets to be transmitted and received by the victim device, and this provides the victim device with a longer time

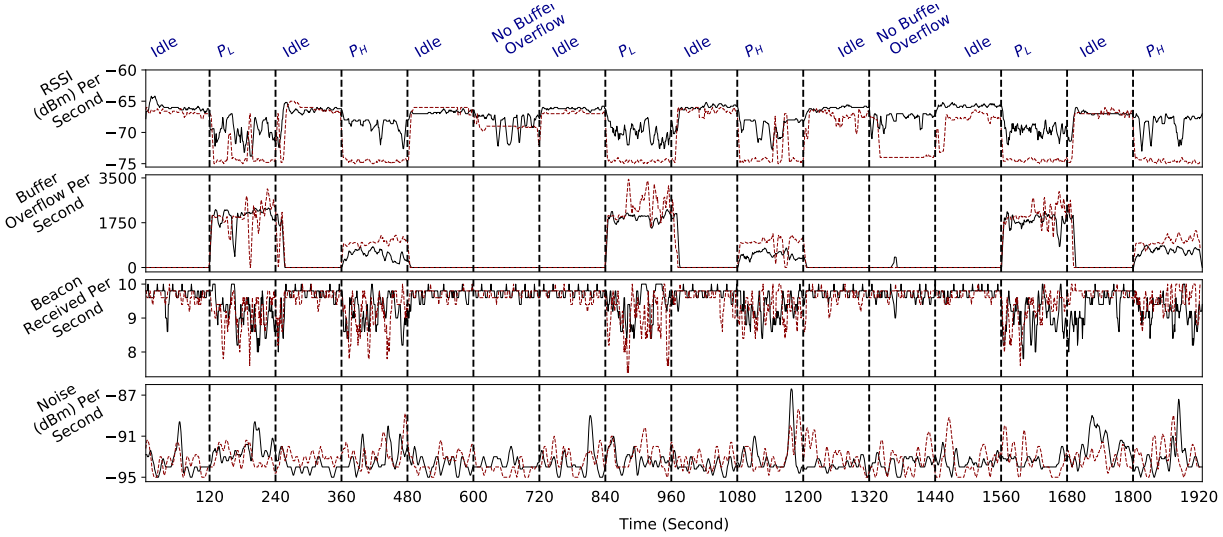

Fig. 4. Internal status of victim device (the DevBoard) under TCP-SYN closed port DDoS attack. The solid black line shows the results when the idle thread is running, and the dash red line shows the results when the busy thread is running. Attacks with no-payload (P_L) and high-payload (P_H) cause buffer overflow in the victim device. Also, increasing the victim device's processor load results in higher number of buffer overflows. These overflows have two major effects: the device cannot accurately measure its RSSI to the AP, and the device cannot respond to AP's time-critical communication requests (such as GTK update) promptly.

to process each packet. This explains why Google Home and Alexa get disconnected under P_L attacks but survive under P_H attacks.

3) ***GTK updates facilitates DDoS*****:** WiFi uses the WPA protocol to provide secure communication between stations and AP. In addition to the pairwise transient key (PTK) assigned to the associated devices, all the devices connecting to the same AP share a group temporal key (GTK) to receive broadcast packets such as address resolution protocol (ARP). GTK is updated periodically (every 10 minutes or once per day, depending on the group key type used) or when a device leaves the network. In the latter case, the update is necessary to prevent the left device from receiving multicast or broadcast packets of the network that the device no longer belongs to. Since the capability of receiving and processing packets by IoT devices drops in the presence of attacks, our hypothesis is that these attacks will prevent the devices from performing GTK update in a timely manner, which in turn results in their disassociation from the AP. In particular, the default `hostapd`'s configuration allows the AP to perform only four tries to update the key. Due to the buffer overflows happening in the victim device, the device may not be able to respond back to the packets sent by the AP to update the GTK; therefore, causing disconnection from the AP. During our experiments, we identify that such GTK updates actually facilitate the DDoS attacks to shorten the survival duration of the victim device. For each IoT device, we record the indices of GTK updates that cause the disconnections.

We run 100 experiments per device. Figure 5 shows the histogram of the indices of GTK update instances that cause victim devices' disconnections. We observe 100% disconnections within the original DDoS attack survival duration in all the four subplots, indicating the impact of GTK on facilitating DDoS attacks. By looking into packet level traces, we

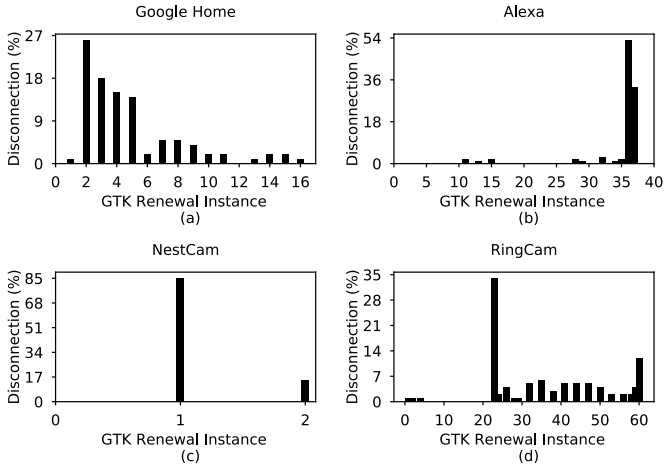

Fig. 5. We ran 100 experiments per device, where each experiment is a TCP-SYN DDoS attack on a closed port. The X axis shows the GTK update instance, and the Y axis shows the percentage of disconnections happening at that particular instance. As this figure shows, GTK facilitates disconnecting the victim devices from the AP.

observed that the AP sends deauthentication packets with the reason code 2 [19]. Taking NestCam as an example, 85% of disconnections happen right after the first GTK update, while the other 15% of disconnections happen right after the second GTK update. It indicates that as long as one or two connected devices move out of the AP's range during the attack, NestCam will be disconnected by the AP due to its failures in updating the GTK. For Google Home, most disconnections are gathered around 2 to 5 GTK updates. However, for Alexa and RingCam, only a very small percentage of disconnections happen before the 20th GTK update. We can conclude that NestCam and Google Home are sensitive to GTK updates when under attack,

while Alexa and RingCam are much less sensitive.

D. Quantitative Understanding of E-DDoS Attacks

In this section, we quantitatively study the impact of E-DDoS attacks on each victim devices' energy consumption.

1) ***Statistical data processing*****:** To study energy consumption versus the incoming attack reception rate of the victim device, we need to time synchronize the data collected from the sniffer (which is used to collect attack rate) and EMPIOT (which is used to collect energy consumption). Although the C&C starts the sniffer and EMPIOT simultaneously, these two devices may not start at the same time due to initialization delays. Therefore, the raw data collected by EMPIOT and `tshark` are not synchronized. We synchronize the two data sequences by correlation analysis. We shift the attack rates data versus energy consumption data in 10 ms slots, and calculate the Pearson correlation values, denoted as r, after each shift. We consider the two sequences synchronized when the maximum correlation value is achieved because the victim device's energy consumption should have a strong correlation with the received attack rate. This is a valid assumption because the energy consumption of WiFi transceiver is considerably higher than the energy consumed by other components such as host processor. For example, considering the DevBoard (CYW43907 [17]) we use in this paper, the current consumption when only the processor is on is 140 mA, and using the transceiver increases this current to around 400 mA. As another example, the current consumed by Avnet BCM4343W is 40 mA when processing, and the current consumption is increased to 350 mA when using the transceiver [18].

The correlation between the energy data and attack rate data is calculated as follows.

$$r = \frac{\sum_{i=1}^{n}(p_i - \overline{p})(v_i - \overline{v})}{\sqrt{\sum_{i=1}^{n}(p_i - \overline{p})^2 (v_i - \overline{v})^2}} \tag{1}$$

where r is Pearson correlation value, n is number of samples, and p and v are the average energy consumption and average attack rate per 10 ms interval, respectively.

After synchronization, we quantitatively evaluate how the victim device's energy consumption changes versus attack rates. To reduce the impact of short-term variations, we split the collected data into a sequence of 500 ms non-overlapping windows and retrieve the average energy and attack rate data for each window. To eliminate energy consumption caused by device itself, we further use the retrieved average energy minus baseline energy, which is collected as the energy consumption of each IoT device during its idle state. Lastly, we apply linear regression to evaluate the relationship between attack rate and victim device's energy consumption. The regression line is computed using the following equation:

$$\Delta p = \alpha_0 + \alpha_1 \times v \tag{2}$$

where Δp is an IoT device's energy increase (percentage), v is E-DDoS attack rate, and α_1 is the coefficient (slope) of the explanatory variable v.

The values of α_1 and α_0 of the regression line are calculated by Equation (3) and Equation (4), respectively:

$$\alpha_0 = \frac{(\sum \Delta p)(\sum v^2) - (\sum v)(\sum \Delta p * v)}{n(\sum v^2) - (\sum v)^2} \tag{3}$$

$$\alpha_1 = \frac{n(\sum \Delta p * v) - (\sum v)(\sum \Delta p)}{n(\sum v^2) - (\sum v)^2} \tag{4}$$

where n is the number of samples.

In addition, by using G*Power [24], we have confirmed that the minimum required sample size is 68 if we aim to achieve the statistical power 0.99 with effect size 0.5 and error probability 0.01. In the regression analysis, for each IoT device, the collected sample data size is 1800 for each specific setting of the influential factors, which is far more than the required size 68. This proves the statistical significance of our regression results.

The collected samples are displayed in Figures 6, 7, 8 and 9 to illustrate the victim devices' energy consumption when attack parameters change. The majority of the collected data samples are concentrated with small variations, which further validates the statistical significance of the observed linear relationship. Although some outliers are also observed in our collected data samples, a robust linear regression algorithm, RANdom SAmple Consensus (RANSAC) [25], has been adopted to minimize the impact of outliers and ensure the reliability of the observed linear relationship.

2) ***Analysis of attack impact*****:** In this section, we study E-DDoS attacks. For each victim device, the E-DDoS attack rates sent by the attacker are below the threshold DDoS attack rates that cause disconnection. In particular, the attack rates applied to Google Home and Alexa are 500, 1000, 2000, 4000, 6000, 8000, and 10,000 pps, for both P_L and P_H. The attack rates sent to NestCam for P_L attacks are 100, 200, 300, 400, 500, 600, 700 pps, as the threshold attack rate is 800 pps. We did not use P_H attack against NestCam because it disconnects with a very small attack rate. The attack rates for RingCam are identical to that of Google Home and Alexa for P_L attacks, and 100, 200, 300, 400, 500, 600, and 700 pps for P_H attacks.

Figures 6, 7, 8 and 9 show the quantitative analysis results for these devices. The solid black lines in each subplot of Figures 6, 7, 8 are the regression lines between the received attack rates and the corresponding percentage energy increase compared to the normal energy consumption of the device. The α values represent the slope of the regression lines.

Google Home: First, we analyze the impact of payload size (P_L and P_H). Taking the ICMP attack as an example, compared to the P_L case (Figure 6 (a)), the maximum attack rate received in the P_H case (Figure 6 (b)) is smaller. However, using P_H causes higher energy consumption at lower attack rates, compared to P_L. Similar observations can be obtained for TCP-SYN and UDP attacks, as the second and third rows of Figure 6 show. This is due to three reasons: First, for each ICMP packet received, the victim replies with a same-size reply packet. Since the victim device contends with the AP to send reply packets, the incoming attack rate of the victim is reduced. Second, energy consumption increases because transmission energy is considerably higher than that of reception

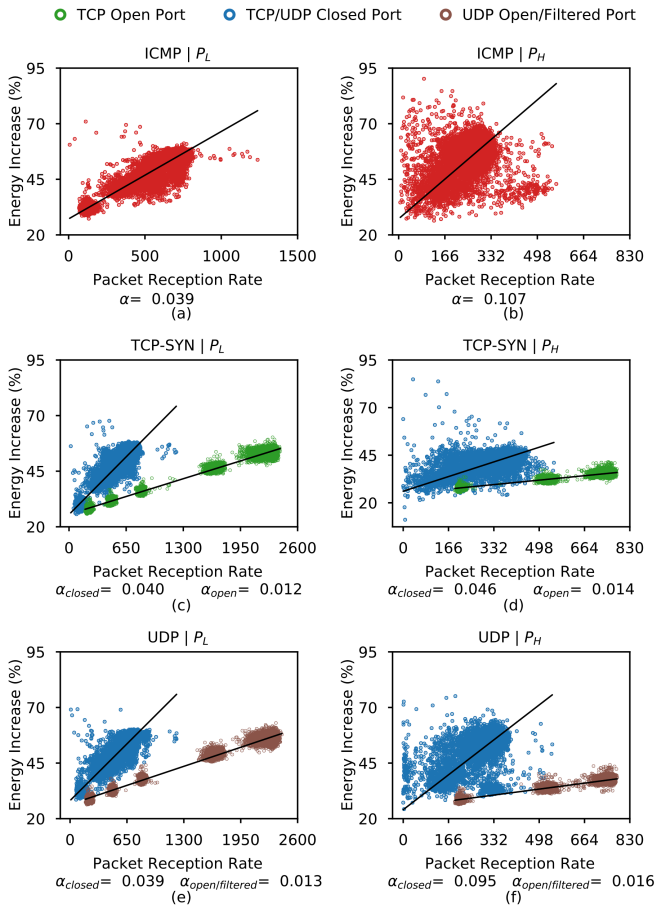

Fig. 6. Google Home under E-DDoS attacks shows maximum energy consumption and received attack rate for high-payload (P_H) ICMP and no-payload (P_L) TCP-SYN open port attacks, respectively.

(e.g., 60 mA for reception vs 300 mA for transmission [17]). Third, transmitting an x Byte packet is not equal with n transmissions of x/n Byte packets. Specifically, the CSMA method employed by 802.11 transceivers requires the device to perform a random backoff before accessing the channel. This means n transmissions of x/n packets includes $n-1$ additional channel access backoffs compared to sending an x Byte packet. Therefore, when sending high-payload packets, idle listening duration drops and the victim device's receiver spends more time in packet reception.

Second, we look into the impact of port state. We focus on TCP-SYN and UDP attacks because ICMP attacks do not specify port number. Considering TCP-SYN attack, Figures 6 (c) and (d) show that the maximum attack rate received by closed ports is smaller than that received by open ports. By looking into the `tshark` files, we observe much fewer SYN-ACK packets (i.e., responses from open port) than SYN-RST packets (i.e., responses from closed port). The reason is that Google Home accepts only a few concurrent connections on each open port, and any excessive received packets will be dropped without further processing. However, a closed port replies with a TCP-RST packet for each received packet. Therefore, we can observe that attacking closed ports leads to a higher α value. Similar observations can be made for UDP closed ports, as Figures 6 (e) and (f) show.

We have also launched UDP attacks against an open/filtered port. The state of such a port cannot be confirmed by network scan using `nmap` because the port does not send any response packet. From Figures 6 (e) and (f), we observe that the maximum received attack rate and α of this port are highly similar to that of the open ports in TCP-SYN attacks (cf. Figure 6 (c) and (d)). Based on these observations, it is highly possible for this port to be open.

Third, we examine the impact of attack type. In terms of the maximum received attack packets, ICMP and TCP-SYN/UDP closed port attacks yield similar results. This is because Google Home responds to all the received packets during these attacks, leading to similar capacity to receive attack packets. Considering the energy consumption of each attack type (α), ICMP and TCP-SYN/UDP closed port attacks show similar values in the case of P_L. However, the value of α in the P_H cases varies significantly, where ICMP achieves the largest α value and TCP-SYN achieves the smallest. The `tshark` traces reveal that the payload sizes of Google Home's response packets are quite different for different attack types. Specifically, (i) an ICMP reply (response to ICMP packets) includes a 1400 Byte payload, (ii) SYN-RST (response of TCP closed port) includes a 0 Byte payload, and (iii) ICMP error reply (response of UDP closed port) includes a 520 Byte payload. This is because (i) ICMP requests and replies always have identical sizes (i.e., 1400 Bytes in our high-payload cases), (ii) SYN-RST packet only updates the TCP header's RST field with no payload required, and (iii) the ICMP error message contains the original ICMP message as much as possible with a maximum payload limit of 576 Bytes [26]. These differences are not observed in the P_L cases, but are very obvious for P_H cases.

Alexa: Figure 7 shows the results using Alexa as the victim. By checking the impact of attack payload on Alexa, we observe a similar trend to that of Google Home: P_H attacks cause smaller maximum received attack rates but result in higher energy consumption.

Next, we analyze the impact of port state. Similar to Google Home, we observe a higher incoming attack rate on open port compared to closed port. By launching UDP attacks against an open/filtered port, we observe similar α values and maximum received attack rates to that of TCP filtered ports. Thus, we consider it is highly possible for the port under attack to be a filtered port. The study on the open/filtered ports of both Google Home and Alexa shows the possibility of using energy analysis to identify the victim device's port state, which cannot be directly determined using `nmap` network scans.

Last, we examine the impact of attack type on Alexa. ICMP, TCP-SYN attacks on filtered ports, and UDP attacks on open/filtered ports all yield similar maximum received attack rates and α values. This is because Alexa does not generate any response when these attack packets are received, thereby leading to a similar capacity to receive attack packets. Further, comparing UDP and TCP-SYN attacks on open ports, the maximum received attack rates with UDP attack is higher. This is because the processing required to handle UDP

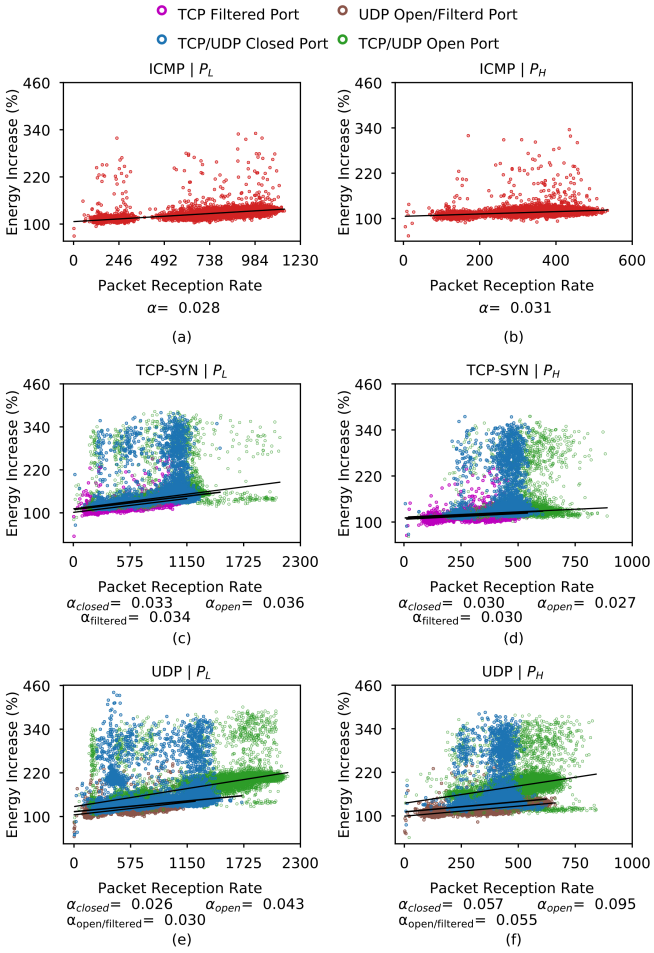

Fig. 7. Alexa under E-DDoS attacks demonstrates maximum energy consumption for high-payload (P_H) UDP attacks on open ports, and demonstrates maximum received attack rate for no-payload (P_L) TCP-SYN/UDP attacks on open ports.

packets is less than that of TCP packets. Specifically, since TCP implements error recovery and rate control, it requires more processing and memory allocation to keep track of per-connection state.

NestCam: We examine the impact of attack type on NestCam, as Figure 8 shows. Similar to Google Home, we find that for P_L attacks, the maximum received attack rates and α value caused by ICMP and TCP-SYN/UDP closed port attacks are very close.

RingCam: RingCam shows a multi-valued association, which means it has two different values of energy consumption versus attack rate, as Figure 9 shows. By `tshark` trace analysis, we observed intervals where the packet transmission rate from the AP significantly drops and instead the reply rate from the victim goes high. This has two effects: First, it causes intervals where transmission power significantly dominates over reception power. Second, two different attack rates (i.e., high and low) are received by RingCam when under a high-rate attack.

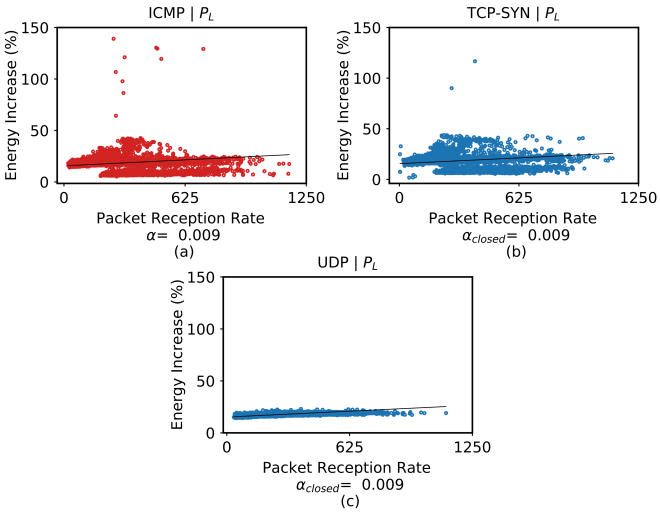

Fig. 8. NestCam under no-payload (P_L) E-DDoS attacks shows similar energy consumption and maximum received attack rate for all the attack types. TCP-SYN and UDP attacks are on closed port.

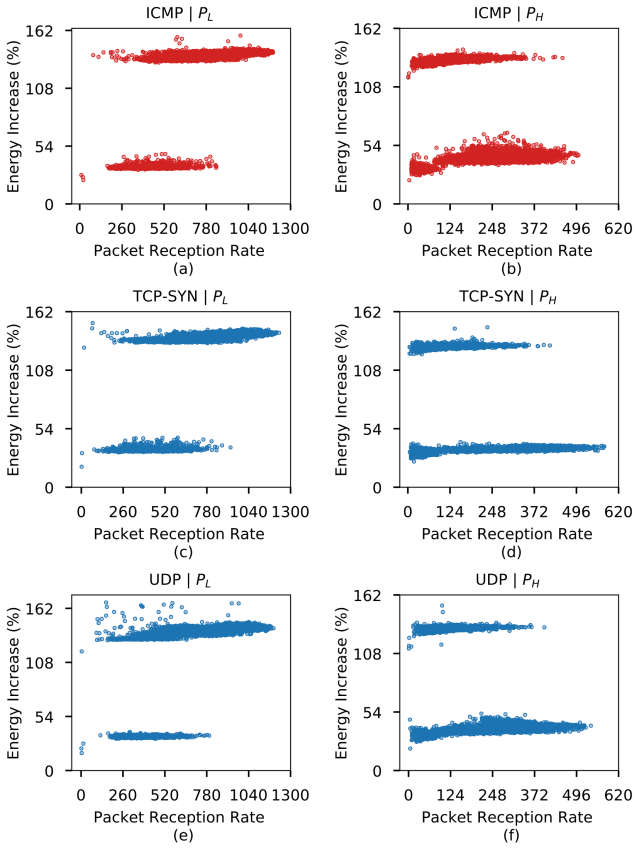

Fig. 9. RingCam under no-payload (P_L) and high-payload (P_H) E-DDoS attacks shows multi-valued association between energy consumption and received attack rates across all attack types. The TCP-SYN and UDP attacks are on closed ports.

E. *Key observations*

In the experiments of this section, we launched moderate attacks against Google Home, Alexa, NestCam, and Ring-

Cam for about 30 minutes and captured energy consumption. During such attacks, the IoT devices continuously receive the packets and spend resources processing these packets. Based on such responses, effective techniques to launch DDoS and E-DDoS attacks against IoT devices are as follows. Our analysis shows that an effective DDoS attack can be launched as a no-payload, TCP-SYN/UDP attack on a closed port or an ICMP attack if the victim responds to ICMP packets. To launch an E-DDoS attack that costs the victim device's maximum energy without being disassociated from the AP, the attacker can launch a high-payload UDP attack against a closed port or an ICMP attack if the device responds to ICMP packets. Furthermore, in the voice assistant category, Google Home is more vulnerable to DDoS attacks, while Alexa is more vulnerable to E-DDoS attacks. In the video camera category, comparing NestCam with RingCam, NestCam is more vulnerable to DDoS attacks and RingCam is more vulnerable to E-DDoS attacks.

IV. **Related Work**

The majority of the existing works strive to propose defense methods against DDoS and E-DDoS attacks, and much less attention has been paid to the quantitative analysis of these attacks. In contrast, by proposing and conducting a pragmatic approach, in this work we quantitatively analyzed and revealed new challenging dimensions considering attack rate, port state, payload size, device's response, and GTK key renewal process.

A. DDoS attacks

The massive deployment and non-secure configuration of smart home IoT devices have made them increasingly attractive to DDoS attacks. Various studies have focused on the impact of DDoS attacks on web servers when the attack is launched by compromised IoT devices. For example, in [27]–[30] the authors discuss the outbreak of the Mirai botnet (and its variants), which compromised smart home IoT devices to launch a DDoS attack against data centers. They claim that even naive approaches can be employed to take control of such devices and create a massive and highly-disruptive army of zombie devices.

The work in [31] analyzes the families of IoT malware that characterize the recent DDoS attacks. The authors in [32], [33], and [34] present the threat of botnet-based DDoS attacks posed to web servers' application layer. They evaluate the performance of web servers under different DDoS attacks and confirm increased CPU usage, slowed response time, and disruption of service to legitimate users. The authors in [15] study reflexive DDoS attacks. In such attacks, the attacker sends a large number of requests to IoT devices with the spoofed IP address of the victim, which is usually a web server. The IoT devices receiving malicious requests send overwhelming response messages to the victim. In [35], the authors estimate that DDoS attacks targeting web servers result in 38 to 114 million dollars in losses. In summary, while the existing work studies the threat of IoT devices posed on web servers, the threat of DDoS and E-DDoS attacks on IoT devices and home users has not been investigated.

Existing studies also show that the consumer IoT devices are not secure. For example, in [36] the authors show that 79.54% of web cameras do not enforce firewall protection, and [37] demonstrate that both commercial and industrial IoT devices are vulnerable to attacks. Our work confirms that Alexa, Google Home, NestCam, and RingCam do not provide firewall protection against DDoS and E-DDoS attacks via ICMP, TCP-SYN, and UDP packets.

Very few studies [38], [39] have been conducted regarding how to launch link-layer DDoS attacks against 802.11 devices.

In [38], the authors examine deauthentication and disassociation DDoS attacks, where the attacker overwhelms the wireless device through fake deauthentication and disassociation packets. They show that by increasing the attack rate, TCP throughput is dropped and UDP packet loss is increased. They have developed a client-device-based queuing model to show that the current IEEE 802.11w standard cannot resolve deauthentication and disassociation at high attacking rates. The work [39] performs association/authentication, disassociation/deauthentication, and probe request DDoS attacks. They propose a detection system for auditing the network and capturing and analyzing packets, which is then used to detect such attacks. In contrast, in this work, we mainly focused on the impact of DDoS and E-DDoS attacks on the resource consumption, response, and connectivity of IoT devices.

B. E-DDoS attacks

Both [40] and [41] examine the fraudulent energy consumption caused by E-DDoS attacks against large-scale cloud services. In [40] the authors present a strategy to launch low-rate E-DDoS attacks on web servers. They demonstrate CPU usage and energy consumption increase versus attack rate, meanwhile they assure that the attack rate is low enough to stay undetected. In [41], the authors present power consumption during CPU- and I/O-bound DDoS attacks on web servers. They claim that CPU-bound attacks achieve higher power consumption; however, I/O bound attacks may slow down the data center responsiveness more than CPU-bound attacks.

To the best of our knowledge, our work is the first to quantitatively analyze the impact of E-DDoS attacks on smart home IoT devices. Although each IoT device may not consume a significant amount of energy, E-DDoS attacks on a large number of such devices in smart homes may lead to approximately $253.7 million in energy loss in a month considering the 5 billion IoT devices in use by 2020.

V. **Conclusion**

In this work, we quantitatively studied the impact of DDoS and E-DDoS attacks on several, broadly-used smart home IoT devices and analyze the underlying reasons for victim devices' various response types. We first designed an automated testing environment for efficient and reliable data collection. For DDoS attacks, we first quantified the threshold attack rate and survival attack duration for each victim device, investigated victim devices' internal status (e.g., perceived RSSI, buffer

overflow, perceived noise, and beacons received) using a WiFi-based IoT development board, and quantitatively studied how the GTK update process facilitates DDoS attacks. For E-DDoS attacks, we identified several key influential factors from the attacker's and victim device's perspectives, including the attack type (i.e., protocol used), attack rate, payload size, and victim devices' port states, and then we quantitatively studied the impact of each on victim devices' energy consumption. The key observations made from this work presents a thorough understanding of the potential vulnerabilities of IoT devices within a home wireless environment. This work provides a solid foundation for future studies on defense solutions.

REFERENCES

[1] Y. Li, Y. Jiang, D. Tian, L. Hu, H. Lu, and Z. Yuan, "Ai-enabled emotion communication," *IEEE Network*, vol. 33, no. 6, pp. 15–21, 2019.

[2] Y. Zhang, Y. Qian, D. Wu, M. S. Hossain, A. Ghoneim, and M. Chen, "Emotion-aware multimedia systems security," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 617–624, 2018.

[3] Y. Zhang, Y. Li, R. Wang, M. S. Hossain, and H. Lu, "Multi-aspect aware session-based recommendation for intelligent transportation services," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[4] H. Lu, Y. Zhang, Y. Li, C. Jiang, and H. Abbas, "User-oriented virtual mobile network resource management for vehicle communications," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[5] K. Lin, F. Xia, C. Li, D. Wang, and I. Humar, "Emotion-aware system design for the battlefield environment," *Information Fusion*, vol. 47, pp. 102–110, 2019.

[6] A. Jalal, M. A. Quaid, and M. Sidduqi, "A triaxial acceleration-based human motion detection for ambient smart home system," in *16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. IEEE, 2019, pp. 353–358.

[7] J. Sheth and B. Dezfouli, "Enhancing the energy-efficiency and timeliness of iot communication in wifi networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9085–9097, 2019.

[8] Y. Zhang, Y. Li, R. Wang, J. Lu, X. Ma, and M. Qiu, "Psac: Proactive sequence-aware content caching via deep learning at the network edge," *IEEE Transactions on Network Science and Engineering*, 2020.

[9] D. Bastos, M. Shackleton, and F. El-Moussa, "Internet of things: A survey of technologies and security risks in smart home and city environments," 2018.

[10] V. Sivaraman, D. Chan, D. Earl, and R. Boreli, "Smart-phones attacking smart-homes," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2016, pp. 195–200.

[11] F. Palmieri, S. Ricciardi, U. Fiore, M. Ficco, and A. Castiglione, "Energy-oriented denial of service attacks: an emerging menace for large cloud infrastructures," *The Journal of Supercomputing*, vol. 71, no. 5, pp. 1620–1641, 2015.

[12] R. V. Deshmukh and K. K. Devadkar, "Understanding DDoS attack & its effect in cloud environment," *Procedia Computer Science*, vol. 49, pp. 202–210, 2015.

[13] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *Proceedings of USENIX Security*, 2017, pp. 1093–1110.

[14] C. Gray, R. Ayre, K. Hinton, and L. Campbell, "'smart'is not free: Energy consumption of consumer home automation systems," *IEEE Transactions on Consumer Electronics*, 2019.

[15] M. Lyu, D. Sherratt, A. Sivanathan, H. H. Gharakheili, A. Radford, and V. Sivaraman, "Quantifying the reflective DDoS attack capability of household IoT devices," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2017, pp. 46–51.

[16] C. Semiconductor. (2020) CYW943907AEVAL1F Evaluation Kit. http://www.cypress.com/documentation/development-kitsboards/cyw943907aeval1f-evaluation-kit.

[17] Cypress Semiconductor. CYW43907: IEEE 802.11 a/b/g/n SoC with an Embedded Applications Processor. [Online]. Available: http://www.cypress.com/file/298236/download

[18] B. Dezfouli, I. Amirtharaj, and C.-C. Li, "EMPIOT: An energy measurement platform for wireless IoT devices," *Journal of Network and Computer Applications*, vol. 121, pp. 135–148, 2018.

[19] A. Arora, "Preventing wireless deauthentication attacks over 802.11 networks," *arXiv preprint arXiv:1901.07301*, 2018.

[20] V. Mhatre and K. Papagiannaki, "Using smart triggers for improved user performance in 802.11 wireless networks," in *Proceedings of MobiSys*, 2006, pp. 246–259.

[21] H. Wu, K. Tan, Y. Zhang, and Q. Zhang, "Proactive scan: Fast handoff with smart triggers for 802.11 wireless lan," in *Proceedings of INFOCOM*. IEEE, 2007, pp. 749–757.

[22] B. Dezfouli, M. Radi, S. Abd Razak, T. Hwee-Pink, and K. A. Bakar, "Modeling low-power wireless communications," *Journal of Network and Computer Applications*, vol. 51, pp. 102–126, 2015.

[23] S. Liu, V. Ramanna, and B. Dezfouli, "Empirical Study and Enhancement of Association and Long Sleep in 802.11 IoT Systems," *Global Communications Conference (GLOBECOM)*, 2020.

[24] F. Faul, E. Erdfelder, A. Buchner, and A.-G. Lang, "Statistical power analyses using g* power 3.1: Tests for correlation and regression analyses," *Behavior research methods*, vol. 41, no. 4, pp. 1149–1160, 2009.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[26] C. Systems. (Dec, 2018) Requirements for IP Version 4 Routers. https://tools.ietf.org/html/rfc1812#section-4.3.2.3.

[27] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[28] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *Proceedings of MILCOM*. IEEE, 2017, pp. 267–272.

[29] A. Marzano, D. Alexander, O. Fonseca, E. Fazzion, C. Hoepers, K. Steding-Jessen, M. H. Chaves, Í. Cunha, D. Guedes, and W. Meira, "The evolution of bashlite and mirai iot botnets," in *Proceedings of ISCC*. IEEE, 2018, pp. 00 813–00 818.

[30] B. Tushir, H. Sehgal, R. Nair, B. Dezfouli, and Y. Liu, "The impact of dos attacks on resource-constrained iot devices: A study on the mirai attack," in *The 11th International Conference on Ubi-Media Computing*. UMEDIA, 2018.

[31] M. De Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "Analysis of DDoS-capable IoT malwares," in *Proceedings of FedCSIS*. IEEE, 2017, pp. 807–816.

[32] K. Singh, P. Singh, and K. Kumar, "Impact analysis of application layer DDoS attacks on web services: a simulation study," *International Journal of Intelligent Engineering Informatics*, vol. 5, no. 1, pp. 80–100, 2017.

[33] M. Jiang, C. Wang, X. Luo, M. Miu, and T. Chen, "Characterizing the impacts of application layer DDoS attacks," in *Proceedings of ICWS*. IEEE, 2017, pp. 500–507.

[34] R. R. Zebari, S. R. Zeebaree, and K. Jacksi, "Impact Analysis of HTTP and SYN Flood DDoS Attacks on Apache 2 and IIS 10.0 Web Servers," in *International Conference on Advanced Science and Engineering (ICOASE)*. IEEE, 2018, pp. 156–161.

[35] E. Alomari, S. Manickam, B. Gupta, S. Karuppayah, and R. Alfaris, "Botnet-based distributed denial of service (DDoS) attacks on web servers: classification and art," *arXiv preprint arXiv:1208.0403*, 2012.

[36] N. Vlajic and D. Zhou, "IoT as a land of opportunity for DDoS hackers," *Computer*, vol. 51, no. 7, pp. 26–34, 2018.

[37] J. Wurm, K. Hoang, O. Arias, A.-R. Sadeghi, and Y. Jin, "Security analysis on consumer and industrial iot devices," in *21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2016, pp. 519–524.

[38] C. Liu and J. Qiu, "Performance study of 802.11 w for preventing dos attacks on wireless local area networks," *Wireless Personal Communications*, vol. 95, no. 2, pp. 1031–1053, 2017.

[39] M. A. C. Aung and K. P. Thant, "Detection and mitigation of wireless link layer attacks," in *Proceedings of SERA*. IEEE, 2017, pp. 173–178.

[40] M. Ficco and F. Palmieri, "Introducing fraudulent energy consumption in cloud infrastructures: a new generation of denial-of-service attacks," *IEEE Systems Journal*, vol. 11, no. 2, pp. 460–470, 2015.

[41] F. Palmieri, S. Ricciardi, and U. Fiore, "Evaluating network-based dos attacks under the energy consumption perspective: new security issues in the coming green ict area," in *International Conference on Broadband and Wireless Computing, Communication and Applications*. IEEE, 2011, pp. 374–379.