International Journal of Emerging Science and Engineering (IJESE)
ISSN: 2319–6378 (Online), Volume-12 Issue-11, October 2024

The Combination of Cyber Security, Excessive Programming, and Available Technology to Create an Effective Weapon

Osamah Ibrahim Mohammed Mohammed

Abstract: ***This research presents a model for designing a military terrestrial drone from available technology by using Raspberry Pi, the proposed model should fulfill the general requirements by providing a secure remote connection, a camera control application with a reliable targeting system, and motion control. We suggested SSH connection with cyber-attack simulation by Ettercap and sslstrip tool to test its reliability, and mark vulnerabilities if exist, as for the camera application and targeting system, python was our main language to write it and we used the SSD object detection method of machine learning for the camera to be able to identify and track objects it sees, python libraries like Tkinter along with PIL library was used to create the application GUI interface and its functions along with OpenCV to provide algorithms for video stream processing and displaying, we used OpenCV also for object detection algorithms writing, and socket was used to create a client-server connection on raspberry pi and windows controlling machine to transfer the targeting control signal, as for the controlling of motion functions its done through SSH PutTTY app and the python curses library. The research's goal is to prove a theory regarding the military technology in the Middle East of mostly using available and civilian-accessed technology provided in stores to manufacture weapons that could show great performance on the battlefield, we improvised this principle in drone manufacture due to its significant role in modern days war.***

Keywords: OpenCV, Raspberry Pi, SSH, SSD

I. INTRODUCTION

In our day drones have proven to be a superior weapon that can act as a main factor for winning wars, while the effect is major, the technology for building such machines can be very simple, yet due to lack of resources for using the technologies we are discussing in this research in the military field makes the work environment a bit improvisational and casts shades on its potential abilities.

PuTTY is a free Windows application that allows users to access the Raspberry Pi terminal through a secure encrypted connection called SSH [1]. SSH is a standard application-layer (under the TCP/IP stack) protocol for remote login over insecure networks for transferring commands. SSH protocol provides a good service in terms of contacting machines running on Linux operating systems [2].

Manuscript received on 10 October 2024 | Revised Manuscript received on 14 October 2024 | Manuscript Accepted on 15 October 2024 | Manuscript published on 30 October 2024.
*****Correspondence Author(s)
Osamah Ibrahim Mohammed Mohammed*, Student, Altınbaş University, Istanbul, Turkey. Email: alskryasamt5@gmail.com, ORCID ID: 0000-0002-2298-873X

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open-access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

We went on to perform an attempt of penetration for the SSH connection by running a scenario of an MITM attack ARP poisoning attack and sslstrip for decoding of SSH encryption.

One of the known techniques for sniffing traffic on LAN networks is ARP poisoning, which uses IP forwarding to sniff and retrieve traffic [5]. The attack is done by a third-party terminal with sslstrip and Ettercap tool on Kali Linux OS. As for the motor functions, camera control app, and targeting system we used excessive programming represented by Python language on both Raspberry Pi and the control machine which is a Windows computer. The targeting system is built by Python OpenCV library object detection, which mainly depends on machine learning to verify objects thus tracking their movement.

Detection of moving and steady objects on images and video streams is mostly done by implementing networks like CNN, ANN, DNN, and other kinds of deep learning algorithms [3]. The model uses OpenCV to capture and broadcast the IP-CAM video stream which will be mounted on the drone's turret, OpenCV also provides object detection capability if combined with a well-trained machine learning weights file with its configuration file, and transmits coordination data to the raspberry pi which holds turret control code by using python socket. Our research objectives are to utilize the best technologies that suit our research aim with simplicity and availability as conditions. Modify and test these technologies so they fit our needs. This study will provide an educational and extremely practical training value for beginners in cyber security, python, and machine learning besides a good demonstration of the usage of these three branches in the military field. Some limitations are that the military field is wide and precise so it requires a lot of experience to work in and a very small margin of error, The usage of civilian technology in the military despite modifying it has its drawbacks.

II. RELATED WORK

Hossain et al [16]. discussed how to design a video surveillance mini rover system that's capable of sending video stream and receiving control commands remotely they used Raspberry Pi 2 operated by Raspbian OS, SSH through PuTTY with Xming server to access the Raspberry Pi desktop, OpenCV to capture the video stream and a control application written by java. They also had to use another terminal as a cloud server written by Java to transfer commands between the control terminal and the rover. GASSER et al [8].

performed research for examining SSH vulnerabilities, they performed a constant scan on SSH for 7 months, including DNS scans, AS and WHOIS lookups, and a geo-IP database, in their Research they explained the SSH operating method to key exchange, the key is a password sent to the server and has to be approved to start the connection, they also agree with what GONZALEZ et al [7]. suggested for SSH being a target to attackers trying to guess the password by dictionary attacks. One of the MITM attacks forms called ARP poisoning, this kind of attack achieves its goal by using IP forwarding to manipulate traffic on LANs to modify the operation of Address Resolution Protocol (ARP) [9]. , Fairweather et al [10]. presented a new model for protecting against SSL striping attacks depending on surveys and analysis of already existing solutions, they describe SSL stripping attack as a type of MITM attack that forces the user to use certain protocols for connection that generally considered unsecured, to achieve that he/she uses a simple public available yet powerful tool called sslstrip. Both HTTP and SSH follow PKI, PKI follows the principle of encryption through the key exchange to allow users to send and receive data through networks that are considered insecure the creation and distribution of private and public keys is done only by a trusted source [11], which is the principle of both HTTPS and SSH. This is also supported by the SSH hacking experiment of Alsaadi et al, [1] where they used Sslstrip to decrypt the SSH packets and they succeeded. Megalingam et al [17]. also demonstrated a Raspberry Pi Robot Operating System through SSH connection, the robot's CPU is accessed from a robot operating console. Commands are passed from the console to the robot's computing unit through SSH, the system consists of 3 units, the control terminal, router, and the robot, the authors also suggest adding Machine learning or artificial intelligence algorithms as future work so the robot can recognize what it sees. According to Kumar et al [13]. who presented a model of object detection that can read users' faces and help them in document searching through OpenCV and TensorFlow libraries, OpenCV is a machine learning library that acts as a modern solution for computer vision problems. This also agrees with Shariff et al [12]. opinion that OpenCV is a library that is mostly used in real-time applications for understanding the context images and is most suitable for face recognition and object detection projects. According to [4]-[15]; SSD is a form of object detection that depends on a single network to perform all of the image processing stages including the resampling of features, it eliminates the pre-processing procedure called proposal generation that includes resizing image scales with continues passing of different scale windows on the image to detect the regions with most possibility of object existence, that's why we can't consider it as object proposal method. One of the SSD issues is that due to its frozen reference training module it can generate multiple bounding boxes around each detected object, one of the methods used to solve this issue is the NMS, according to Ye Pan and Feng Dong [14]; Non-Maximum Suppression (NMS) Is a procedure of canceling bounding boxes which has lower thresholds that surround a single object and maintaining only boxes with higher threshold. The threshold is an indication of detection accuracy and the NMS eliminates the box with a lesser threshold. This is also supported by what procedure Wei Liu et al [15]; took to reduce the multiple bboxes, they stated that due to the multiple bboxes generated by SSD, it was essential to use NMS under a threshold of 0.01 to reduce them. Radhika K A et al [6]. created an IoT system of a Raspberry Pi terrestrial drone controlled by another Raspberry Pi joystick the system is client-server based, and the connection is done by a wireless TCP/IP socket written by Python. The system is interesting because it uses Logitech Driver Force GT to build a control unit that simulates a real driving cockpit and uses 2 Raspberry pi's as client and server yet our SSH system provides both data confidentiality and integrity beside it is originally designed for commands transferring on Linux-like operating systems and we are considering socket in our system for transmitting targeting system control signal.

III. METHODOLOGY

In this section, we will declare how we used the available technology and the research we discussed in the previous section to develop our military terrestrial drone, and what kind of modifications we performed on them to make them suitable for our goal.

A. Ssh Security Experiment

This experiment is a replica of what Alsaadi et al [1]. did in their paper. Yet, their experiment was in 2013 and we are performing it today on more modern technology to keep track of the vulnerabilities they detected.

i. Nessus Analyzing:

According to Williams et al [18]. Nessus is a wide-range used vulnerability scanner due to its gold standard among the information security community, it has good credentials for scanning Internet of Things (IoT) systems and can scan and detect vulnerabilities, on ports, networks, and operating systems software's and systems that usually contain different hardware and software components, the user of Nessus can configure his plugins and test the system without the necessity to stop or interact with its operation. For our research, we performed a Nessus scan but instead of Kali 4.2 and Raspbian Jessy of Linux 4.4, we did it on Raspbian gnu Linux 10 buster operating system.

ii. Nmap Analyzing:

According to Shah et al [19]. Nmap is an open-source tool that is used in network security for port and network scanning, It is considered number 1 in this field and can check for the condition of TCP/UDP ports whether they are open or closed.

We used Nmap in our experiment to first analyze the Raspberry Pi SSH port and as a way to search for IP addresses on the same LAN for the MITM attack simulation.

iii. MITM Attack Simulation:

The main purpose of this simulation is to test the SSH connection security level and whether the system can counter a full scenario hacking attempt done by Ettercap and sslstrip.

The Ettercap is a sniffing tool mostly activated on Linux systems and can be used to perform sniffing by many methods like MITM attacks DDOS attacks packet filtering, DNS spoofing, etc. [28].

© Copyright: All rights reserved.

The attack scenario is performed by three terminals one is our Windows machine control terminal the other is our microcontroller Raspberry Pi 4 and the attack is performed by a Kali Linux run on VMware virtual machine.

- On the attacker machine we start by opening a root terminal and initialize the IP forwarding to the attacker machine by this command "sudo echo 1 > /proc/sys/net/ipv4/ip_forward".
- We redirect SSH requests to port 10000 by this command "iptables -t nat -A PREROUTING -i eth0 -p tcp --destination-port 22 -j REDIRECT --to-port 10000"
- We activate Ettercap by this command "Ettercap_g".
- From Ettercap GUI we start by performing a HOST search and then determine 2 IPs from the host list 1st is the network IP which ends in number 1 the other is the target machine IP.
- We select the 2 IPs as target1 and target2 Then we start the MITM attack by selecting ARP poisoning from the attack list.
- After activating the ARP poisoning, we open a new terminal on the Kali machine and activate sslstrip by this command "sslstrip -a -k -f -l 10000".
- We monitor the data flow through Wireshark by setting it to SSH protocol monitoring.
- the original experiment required the attacker to operate on the Fedora 20 operating system but since it's obsolete and we faced some malfunctions in sslstrip installation on its new version Fedora 35 we used Kali Linux.

Due to SSH key exchange and session ID forwarding mechanism, ARP poisoning will not have a result and no packets will be displayed on Wireshark if the attack occurs after the establishment of an SSH connection. This confirms the opinion of GASSER, HOLZ et al [8]. that there is no possibility for an MITM attack on the SSH if the client public key is accepted by the server and the connection is already established.

B. Camera Application and Targeting System Design and Testing

For its compatibility with both Windows and Linux operating systems, and supporting all the libraries we used Python as the main programming language for our monitoring system.

i. Camera control app:

We require an application that displays camera stream, takes pictures, starts/stops the camera, and activates an IP camera that requires an IP address to start streaming, this required us to design an internal window for IP and port number, the window opens by activating the IP button on the main window and automatically close after inserting the IP and port number and clicking on BROADCAST button then the IP camera stream starts, in case either of the IP and the port number is missing, an error message will show requesting them to be written. We used the toplevel () function from the Tkinter library to enable us to create the sub-window for the IP. As for the application itself, we created all the buttons, labels, text fields, and entries from the Tkinter library, and we also used the OpenCV library for drawing lines and dots we used to create the camera viewfinder sight and time/date indicator. The video stream algorithms were also written by OpenCV, cv2.VideoCapture() to capture the stream frame by frame and cap.read() to display the stream on the main window. The application is designed to start/stop the stream by button and to take photos from the stream and store them on the PC hard drive. The image capture depends mainly on the PIL (python imaging library) also known as 'Pillow' This library adds image processing capabilities to the Python interpreter. It provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities, The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image-processing tool [20].

The 'Image' module of this library provides a class with the same name which is used to represent a PIL image. The module also provides several factory functions, including functions to load images from files, and to create new images [20]. After obtaining the captured image it gets displayed on the main window by loading it on the image label, which we created by the Tkinter library after configuring the label to the image properties. By the usage of the 'file dialog' module of Tkinter, we were able to browse through files and directories of the machine's hard drive thus setting paths for storing and displaying images we captured The images displayed on the main window are done by first pressing the BROWSE2 button and selecting the desired image from the storage, the **Image.open()** enables us to turn it into a PIL image from the path we selected and The **ImageTk.photoImage()** turns it into a bitmap image and it is displayed on the label the same way we explained previously.

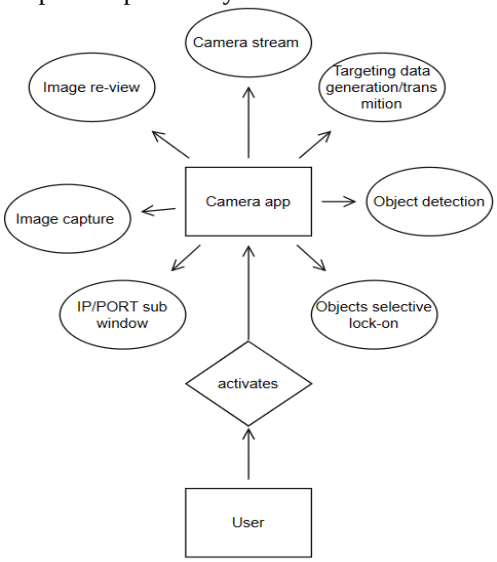

Figure 1: Camera Application Entity Relationship Diagram

ii. Targeting system design:

'coco.names' is a class file that represents a handful of objects we encounter daily and contains image annotations in 80 categories, with over 1.5 million object instances [21]. We used this class to connect the object we detect to its name by subtracting 1 from the class ID value of the detected object since its class numbers start from zero by using this line 'classNames[classIds[i]-1]'

Retrieval Number:100.1/ijese.D40540412423
DOI: 10.35940/ijese.D4054.12111024
Journal Website: www.ijese.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

Yet we faced a single problem since our 'coco.names' class file is formed of 80 members who are considered few, the objects we detect come up with classes sometimes higher than 80 which leads to list index being out of range thus app crash A solution we came up with is to add more elements to the class "blanks" to cover up for the shortage of elements and avoid this issue. OpenCV uses "cv2.dnn_DetectionModel (weightsPath,configpath)", for the launch of the weights file which represents a model trained with multiple pictures used for comparing with objects on the video frames to generate detection data depending on the threshold level in our model, we are using pre-trained face detection convolutional neural network (CNN) of "cv2.dnn_DetectionModel ()", Deep Neural Network (DNN) module [22], to start with detection we used this function "classIds, confs, bbox = net.detect(frame,confThreshold= 0.45)" while classIds represent the IDs of objects within, bbox represent the bounding boxes which we will use to surround the detected objects and the confs represent the correct percentage of each object to be identical to its class ID number. as for NMS solution for the Overlapping objects, open CV provides a module for NMS "cv2.dnn.NMSBoxes(bbox, confs,thres,nms_threshold)", using 0.2 nms_threashold value. Now according to the bboxes values the 'cv2.dnn.NMSBoxes()' will eliminate all bboxes with low thresholds and maintain only higher threshold boxes. Yet our weights file is trained to detect a lot of objects (persons, cars, animals. Etc…) but not a specific one, By using the 'if, else' module, since we already have a method for giving every object we detect a name from the coco.names file depending on its class ID, we can use the if condition to make it detect only the IDs with the name (person) by applying this condition ' if d == 'person' ' while 'd = classNames[classIds[i]-1]' to our 'for (i) in (indices):' which is the loop responsible for displaying bounding boxes and detection data for each detected object, and by adding "float(str(round(confs[i],2))) >= 0.5" we can rise detection accuracy to only objects with name 'person' since confs represent the correct percentage of each object to be identical to its class ID number. As for how the tracking objects on the frame first we determined the middle points of our screen by dividing the row and column value of the screen on 2 we got the zero point which is the middle point of the screen, after setting the zero point the motion of the object on the frame is detected according to the object position from or to the frame, if the x-medium was lower in value than the zero point the argument 'position' will increase above 90 which is the zero point of servo motor, according to that incrementation we can make the servo move in the direction required until the incrementation stops and the object is back on the zero point and vice versa. We used sockets to send position data from the Windows machine to the Raspberry Pi for turret servos guidance, The sockets are defined as a bridge's endpoint where the server can send data to multiple clients and a client connects to that server after obtaining its IP address and the port number, The connection could be achieved through LANs or wide area networks if either term is on a different network [23]. For our design, the server is implemented by the Windows machine through the server python socket code we wrote and the client code written on our raspberry pi, for the server site we are using the sending algorithm 'data = position.encode('utf-8')' to send the 'position' data to the raspberry after encoding it. The encoding method we are using is utf-8, the utf-8 (Unicode transformation format) it's a commonly used encoding and it's Python default, it can handle any shape of data including ASCII data, due to its unique data shape it's immune to data loss, it can represent characters by unique bytes of its own for each character so it doesn't face issues with operating on different hardware's [24]. On the client side, we have to determine our server IP and the port number as demonstrated in the message arriving through the web and enter the controlling of the servo as long as the connection between client and server is established.

The security depends on a unique form of key exchange called meeting ID it is a symmetric form of encryption where the meeting ID which is10-11 digit id associated with meetings determines whether the connection is to be established or not, if the client and server own a similar meeting ID then the encryption key is delivered to both of them and they can both decrypt the transition yet if not and for an example, an MITM attacker entered between them with different meeting ID he will only get the encrypted transition since he will get a different thus useless encryption key [23]. Socket supports bidirectional communication on different devices in the same way it does on similar ones so it's perfectly suitable for Windows laptops and Linux-based Raspberry Pi OS [25]. The server socket takes its IP from its host machine while the client reaches the server through that IP [26][29][30][31][32][33]. And on any common port number between them. The only thing remaining is how to do target selection, in case the system is detecting multiple objects and only one of them is a target. Our object detection system can mark multiple targets with bounding

```
for e in indices:
    q+=1
if 0<c:
    if keyboard.is_pressed('f'):
        c -=1
if c<e:
    if keyboard.is_pressed('g'):
        c += 1
```

Figure 2: Argument C Increment/ Decrement

boxes by using the for-loop illustrated in for each bounding box detected by our algorithm 'indices = cv2.dnn.NMSBoxes(bbox,confs,thres,nms_threshold)' the for-loop increments by value (i), which makes (i) the indicator of how many objects detected within a single frame, the (i) number is unique for each object and by using it within the algorithm we improvised in **figure 2**, we were able to design a quick-aim targeting system activated by 2 directions by keyboard buttons, the system simply increment or decrement the argument (c) according to the pressing of (g) and (f) buttons from the keyboard of windows machine, and by that value, we can control the (i) by an 'IF' condition to display the targeting circle coordinated by x_medium value only on a single object which is the target.

Retrieval Number:100.1/ijese.D40540412423
DOI: 10.35940/ijese.D4054.12111024
Journal Website: www.ijese.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

iii. Motional Functions:

Motional functions of our project include the control of turret rotation which is composed of 2 servo motors along with a pan tilt platform that makes the turret structure, also the control of robotic chassis movement which is composed of a metal 10x5 cm chassis body with 2 DC motors and an L289n Motor driver unit for the control of motion and speed of DC motors, all connected to the raspberry pi microcontroller unit and controlled by windows machine keyboard. The controlling code is written in Python and uses the curses library to translate keystrokes from the keyboard to the Python code. Curses is a C-based-UNIX-like library its function is to allow the user to interact with his code by terminal input devices such as the keyboard and enable him/her to move the cursor or scroll the window and do similar functions [27]. In our case, the keyboard of the Windows machine operates as an input device for the Raspberry Pi through SSH. The main insertions we used to make this library operate on our code, the 'screen = curses.initscr()' is for initializing the library and determining the terminal type, 'curses.noecho()' it disables the echo mode of the terminal and provides a different interface to display the keys, the 'curses.cbreak()' eliminates the need to press enter to perform application tasks instead instantly while the 'screen.keypad(True)' is responsible for activating curses keys and other special keys on the keyboard [27]. We also use the GPIO library to activate the raspberry pi pins, As for the turret control, we use PWM pins to get analog-changed signals to control the servo motors motion, we start by assigning each of our servos to a PWM pin and then initialize them to 0 to control the signal of a PWM pin we use "ChangeDutyCycle()" this function controls the PWM in the range from 0-100 so we need to set it to an integer value that increases or decrease between 0-100 We set two variables and initialized them to 0 then we control the signal by increasing or decreasing the argument values according to the buttons pressed on the keyboard, Since this time, we are using alphabetic characters (W, S, A, D) instead, For arrow buttons we are using the ord(), function to return the number values of characters and for each button the arguments increase or decrease moving the servos in a direction, since the "Change Duty Cycle()" function operates only between 0-100 we had to limit the arguments between 0-100 by IF conditions to avoid app crash. Since we are using curses which require us to disable some functions of the terminal window to operate, to restore the terminal as it was before operating the code, we must turn on the terminal functionalities before closing the code, This is done by using lines on **figure 3**, we use the "Try-finally" function and place the lines we mentioned in the final section, the final section activates when the try section suffers a bug leading to an app crash which is what happens when we activate close on the terminal so the reactivating lines operate directly and the terminal reoperates as normal.

```
93  finally:
94      curses.nocbreak();screen.keypad(0);curses.echo()
95      curses.endwin()
96      GPIO.cleanup()
```

Figure 3: Terminal Functions Activating

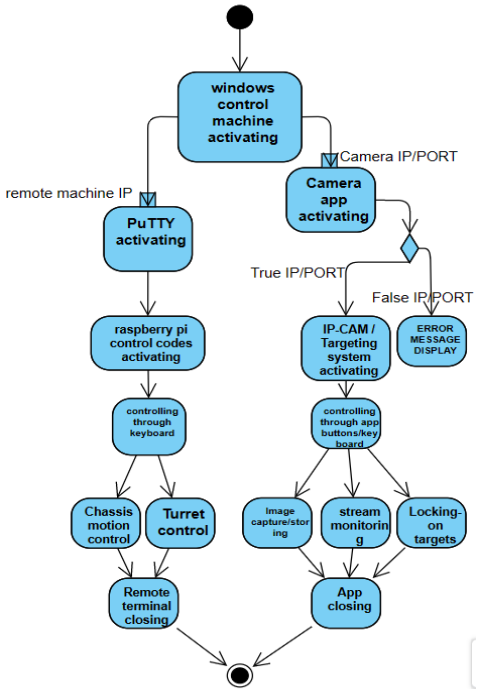

Figure 4: Proposed System Activity Diagram

IV. RESULTS

A. Ssh Security Test

First, our Nessus vulnerabilities testing resulted in only the SSL certificate being untrusted which means that the browser's listed certificate providers do not contain the official certificate authority (CA) within or the SSL certificate is issued by the server itself. The Nmap analysis proved that the SSH gateway is open by default on the Raspberry Pi which could make a vulnerability. As for the MITM attack simulation, our results showed a complete interruption for the SSH connection packets by the Ettercap ARP poisoning attack yet the sslstrip failed to downgrade version 2 of SSH connection to the unsecured version 1, so it was able to maintain its encryption and we did not lose the confidentiality of the username and password there are 2 versions of SSH, SSHv1 and SSHv2, SSHv1 is considered now unsecured yet the SSHv2 is more secure, both versions of SSH are available on Putty 0.78 version which we are using, yet our raspberry pi 4 with Raspbian gnu Linux 10 buster OS contain only SSHv2 which makes any attempt of contacting it with SSHv1 with no avail.

B. Camera Application

For our camera app, the results showed complete success in broadcasting the IP-CAM stream, the sub-window also showed the desired performance by closing automatically after the broadcast and rejecting the activation if either the IP or port number is missing and showing an error message. We were also able to capture images during the streaming and save them on the machine's hard drive and display them on the previewing label by only setting the saving path first **Figure 5**.

Retrieval Number:100.1/ijese.D40540412423
DOI: 10.35940/ijese.D4054.12111024
Journal Website: www.ijese.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

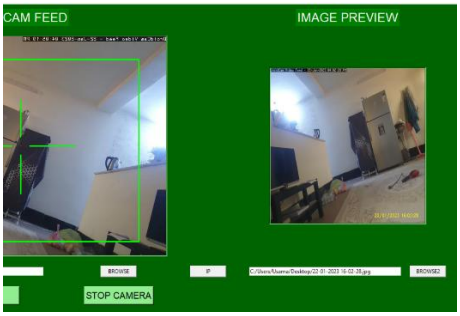

Figure 5: Captured Image Previewing on Preview Label

C. Targeting System

The SSD model of Object detection was successful in detecting objects and classifying them by name, as we see in **Figure 6a,** the camera successfully detected the infant picture and gave a classification of the person we can see on the terminal screen the name person and the ClassIds number which is "1" the same number of person sequence on the coco.names class, also the object detection was shorted to only persons due to the IF condition we improvised as we mentioned in the previous chapter, if we check **Figure 6b,** we can see that the terminal is detecting multiple IDs for multiple objects yet only the infant is surrounded by bounding box and given the name and threshold data. As for our targeting system, the control signal showed a complete interaction with the changing position of the targeted object, our targeting indicator represented by the red dot in the middle of the bounding box in **Figure 6a,b** showed success in switching between multiple detected objects and generating targeting data for each object targeted by red dot, the targeting data showed constant changing with object targeted by red dot motion then transferred to raspberry pi by client-server socket system with no delay or missing data.

Figure 6a: Only Person Detected/Marked by Bbox

Figure 6b: Multiple Objects Detected, Only Person Marked by Bbox

D. Motion Functions

We were able to control the activation and deactivation of the GPIO pins of the Raspberry Pi from the Windows control machine keyboard and through the SSH connection between the machine and the Raspberry Pi. The first results we got are for the rover control which showed success in the activation and deactivation of control pins for each direction of motion and stop phases, for each FORWARD, BACKWARD, RIGHT, LEFT and STOP phase GPIO pins showed success In activating and controlling motor polarity through the motor driver to move in the desired direction or stop the same goes for curses library which was successful in translating each keystroke on the keyboard whether it was direction arrows or normal letters with ASCII codes, also the deactivating of terminal undesired functions and restoring them through the curses library and the "try-except" module was a success.

V. CONCLUSION

From the principle of available technology, the SSH is quite suitable for our goal of controlling a military terrestrial drone, it's quite secure and very flexible in terms of command transferring. And with SSHv2 officially adopted by Raspberry Pi modern operating systems the possibility of a downgrading attack is low. Object detection showed a great performance in designing our targeting system yet it suffered some limitations in terms of lag in detection that led to target tracking being sloppy, we could also mention that the operating of object detection takes a huge amount of processing power along with network bandwidth while operating with an IP camera which causes some lag in its performance.

The OpenCV was quite useful in terms of both image processing and object detection since it was able to handle the IP camera video broadcasting, setting the camera viewfinder, providing a full DNN object detection model along with NMS suppression for detection processing, drawing bounding boxes, displaying detection data, and targeting dot. Raspberry Pi was generally used as the microcontroller unit of all the terrestrial drone designs, we went through in our literature review, yet its functions in such projects are quite small compared to its capabilities and price we suffered some limitations in controlling the GPIO pins since they require a STOP phase to change the pins condition instead of adapting a push-button module of changing signal when the operator lifts his finger from the button. In general, the system has proven its ability to answer our research questions and proven that the concept of improvising in available technology could result in the development of effective weapons yet in this research we could say that the researcher set the main basis and more future development required to fulfill all of our research aims.

DECLARATION STATEMENT

I must verify the accuracy of the following information as the article's author.

Retrieval Number:100.1/ijese.D40540412423
DOI: 10.35940/ijese.D4054.12111024
Journal Website: www.ijese.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been funded by any organizations or agencies. This independence ensures that the research is conducted with objectivity and without any external influence.
- **Ethical Approval and Consent to Participate:** The content of this article does not necessitate ethical approval or consent to participate with supporting documentation.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Authors Contributions:** The authorship of this article is contributed solely.

REFERENCES

1. H. H. Alsaadi, M. Aldwairi, M. Al Taei, M. AlBuainain, M. AlKubaisi. Penetration and Security of OpenSSH Remote Secure Shell Service on Raspberry Pi 2, 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2018. https://doi.org/10.1109/NTMS.2018.8328710
2. I. H. Huang, W. J. Tzeng, S. W. Wang, C. Z. Yang, "Design and Implementation of a Mobile SSH Protocol," TENCON 2006 - 2006 IEEE Region 10 Conference, 2006. https://doi.org/10.1109/TENCON.2006.343956
3. T.Sunitha, Dr. G.Lavanya, "A Novel Approach for Object Detection using Integrated Approach," 2022 International Conference on Electronics and Renewable Systems (ICEARS), 2022. https://doi.org/10.1109/ICEARS53579.2022.9752373
4. X. Wua, , D. Sahoo b, S. Hoi, "Recent advances in deep learning for object detection," Neurocomputing, Vol. 396, pp. 39-64, 2020. https://doi.org/10.1016/j.neucom.2020.01.085
5. A. Ornaghi M. Valleri, "Man in the middle attacks," 2003 Blackhat Conference Europe, 2003
6. K. A. Radhika, B. L. Raksha, B. R. Sujatha, U. Pruthviraj, K. V. Gangadharan, "IoT Based Joystick Controlled Pibot Using Socket Communication," 2018 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), 2018. https://doi.org/10.1109/DISCOVER.2018.8674130
7. S. González, Á. Herrero, J. Sedano, U. Zurutuza, E. Corchado, "Different approaches for the detection of SSH anomalous connections Different approaches for the detection of SSH anomalous connections," Logic Journal of the IGPL, Vol. 24, no. 1, pp. 104-114, 2016. https://doi.org/10.1093/jigpal/jzv047
8. O. Gasser, R. Holz, G. Carle, "A deeper understanding of SSH: Results from Internetwide scans," 2014 IEEE Network Operations and Management Symposium (NOMS), 2014. https://doi.org/10.1109/NOMS.2014.6838249
9. F. Callegati, W. Cerroni, M. Ramilli "Man-in-the-Middle Attack to the HTTPS Protocol" IEEE Security & Privacy, Vol. 7, no. 1, pp. 78-81, 2009. https://doi.org/10.1109/MSP.2009.12
10. D. Fairweather, H. Mozer, S. Rinehart, D. Shin, "An Enhanced Approach To Preventing the SSLstripping Attack," 2015 International Conference on Information and Communication Technology Convergence (ICTC), South Korea, 2015. https://doi.org/10.1109/ICTC.2015.7354559
11. Z. Guo, T. Okuyama, M. Finley, "A New Trust Model for PKI Interoperability," Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services - (icas-isns'05), 2005. https://doi.org/10.1109/CSITSS54238.2021.9683011
12. A. Shariff, R. Bhatia, R. Kuma, S. Jha, "Vehicle Number Plate Detection Using Python and Open CV," 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), India, 2021.
13. R. G. Kummar, S.J. Shetty, S. N. Vishwas, P J. Upadhya, J. R Munavalli, "Edu-bot: An AI based Smart Chatbot for Knowledge Management System," 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), 2021.
14. Y. Pan, F. Dong, "Suppression and Enhancement of Overlapping Bounding Boxes Scores in Object Detection," 2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), United Arab Emirates, 2019. https://doi.org/10.1109/ISSPIT47144.2019.9001826
15. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, A. Berg, "SSD: Single Shot MultiBox Detector," arXiv platform. Computer Vision and Pattern Recognition, pp. 21-36, 8 Dec 2015. https://doi.org/10.1007/978-3-319-46448-0_2
16. N. Hossain, M. Kabir, T. Rahman, M. Hossen, F. Salauddin, "A Real-time Surveillance Mini-rover Based on OpenCV-Python-JAVA Using Raspberry Pi 2," 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Malaysia, 2015. https://doi.org/10.1109/ICCSCE.2015.7482232
17. R. K. Megalingam, S. Tantravahi, H. Tammana, N. Thokala, H. Puram, N. Samudrala, "Robot Operating System Integrated robot control through Secure Shell (SSH)," 2019 3rd International Conference on Recent Developments in Control, Automation & Power Engineering (RDCAPE), India, 2019.
18. R. Williams, E. McMahon, S. Samtani, M. Patton, H. Chen "Identifying Vulnerabilities of Consumer Internet of Things (IoT) Devices," 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, 2017. https://doi.org/10.1109/ISI.2017.8004904
19. M. Shah, M. Junaid, S. Ahmed, H. Khan, K. Saeed, A. rehman, "Penetration Testing Active Reconnaissance Phase – Optimized Port Scanning With Nmap Tool," 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2019. https://doi.org/10.1109/ICOMET.2019.8673520
20. J. A. Clark "Pillow (PIL) fork 9.4.0 documentation," 2023. [Online]. FAvailable: https://pillow.readthedocs.io/en/stable/index.html. [Accessed Jan 20 2023].
21. S. Annotate "Introduction to the COCO Dataset," 2021. [Online]. Available: https://opencv.org/introduction-to-the-coco-dataset/. [Accessed Jan 20 2023].
22. Wei Lin. "Extracting Coefficients of OpenCV Face Detection DNN model. towards data science," 2020. [Online]. Available: https://towardsdatascience.com/extractingcoefficients-of-opencv-face-detection-dnn-model-7f3d944898b9. [Accessed 13 Feb 2023].
23. B. Liu, R. Rajasekar, A. Shukla, A. Solomon, L. Xu, Integrating "Natural Language Processing and Computer Vision into an Interactive Learning Platform," 2020 IEEE MIT Undergraduate Research Technology Conference (URTC), 2020. https://doi.org/10.1109/URTC51696.2020.9668869
24. A. Kuchling. A. Belopolsky, G. Brandl, A. Kuchling, and E. Melotti. "Unicode how to,"2023.[Online].Available: https://docs.python.org/3/howto/unicode.html#:~:text=UT%20F%2D8%20is%20one%20of,used%20%20than%20UTF%2D8.). [Accessed 20 Feb 2023].
25. G. Satyanarayana, D. Chakraborty, S. Das, "Application Oriented Sensor Database System," 2017 International Conference on Networking and Network Applications (NaNA), Nepal, 2017. https://doi.org/10.1109/NaNA.2017.64
26. N. Singh, V. Mahajan, A. Aniket, S. Pandya, R. Panchal, U. Mudgal, M. Bhatt, "Identification and Prevention of Cyber Attack in Smart Grid Communication Network," International Conference on Information and Communications Technology (ICOIACT), Indonesia, 2019.
27. A. Kuchling, E. Raymond, "Curses Programming with Python," 2023, [Online]. Available: https://docs.python.org/3/howto/curses.html#:~:text=Before%20doing%20a%20ny. [Accessed 20 Feb 2023].
28. B. Pingle, A. Mairaj, A. Javaid, "Real-world Man-in-the-middle (MITM) Attack Implementation Using Open-Source Tools for Instructional Use," 2018 IEEE International Conference on Electro/Information Technology (EIT), 2018 https://doi.org/10.1109/EIT.2018.8500082
29. Kalra, Y., Upadhyay, S., & Patheja, Dr. P. S. (2020). Advancements in Cyber Attacks and Security. In International Journal of Innovative Technology and Exploring Engineering (Vol. 9, Issue 4, pp. 1520–1528). https://doi.org/10.35940/ijitee.d1678.029420
30. Alhathally, L., AlZain, M. A., Al-Amri, J., Baz, M., & Masud, M. (2020). Cyber security Attacks: Exploiting weaknesses. In International Journal of Recent Technology and Engineering (IJRTE) (Vol. 8, Issue 5, pp. 906–913). https://doi.org/10.35940/ijrte.e4876.018520
31. Altwairqi, A. F., AlZain, M. A., Soh, B., Masud, M., & Al-Amri, J. (2019). Four Most Famous Cyber Attacks for Financial Gains. In International Journal of Engineering and Advanced Technology (Vol. 9, Issue 2, pp. 2131–2139). https://doi.org/10.35940/ijeat.b3601.129219

© Copyright: All rights reserved.

32. Onome, Dr. O. A. (2022). Advanced Cyber Exploitation and Mitigation Methodology. In International Journal of Emerging Science and Engineering (Vol. 10, Issue 4, pp. 8–15). https://doi.org/10.35940/ijese.c2525.0310422
33. Balasubramanian, Dr. K., Arun, M., & Sekar, Dr. K. R. (2022). An Improved RSA Algorithm for Enhanced Security. In Indian Journal of Cryptography and Network Security (Vol. 2, Issue 2, pp. 1–4). https://doi.org/10.54105/ijcns.b1421.112222

AUTHOR PROFILE

Osamah Ibrahim Mohammed Mohammed graduated from Al-iraqia university-college of Engineering ib Baghdad/Iraq with a bachelor's degree in network engineering, and is currently, a master's degree student in Altinbas University at the Istanbul/Turkey College of Engineering Electrical and Computer Engineering section, participating in Baghdad international exhibition for engineering projects and achieved 4th place, made multiple types of research regarding cybersecurity computer architecture programming and robotics during master study period achieving 3.94 GPA in his preparing year of master study, working in android applications development and python/ java programmer as a freelancer.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of the Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)/ journal and/or the editor(s). The Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP) and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© Copyright: All rights reserved.