Research Article

# Artificial Intelligence-Supported Detection Systems on Embedded Devices

Feyza Alnıaçık[1a], Furkan Yıldırım[2b], Serkan Gönen[3c] Birkan Alhan[4d] Mehmet Ali Barışkan[3e] Hasan Hüseyin Sayan[5f] and Ercan Nurcan Yılmaz[5g]

[1] Fetih Plastik San. Tic. Ltd. Şti., Istanbul, Turkiye
[2] Castrol Ltd. Şti., İstanbul, Turkiye
[3] Department of Software Engineering University of Istanbul Gelisim University, Istanbul, Turkiye
[4] Garanti BBVA Teknoloji, Ankara, Turkiye
[5] Gazi University, Faculty of Technology, Ankara, Turkiye

sgonen@gelisim.edu.tr

DOI : 10.31202/ecjse.1312555
Received: 11.06.2023 Accepted: 18.12.2023
**How to cite this article:**
Feyza Alnıaçık, Furkan Yıldırım, Serkan Gönen, Birkan Alhan, Mehmet Ali Barışkan, Hasan Hüseyin Sayan, Ercan Nurcan Yılmaz, " Artificial Intelligence-Supported Detection Systems on Embedded Devices", El-Cezeri Journal of Science and Engineering, Vol: 11, Iss:1, (2024), pp.(109-119).
ORCID: [a]0000-0002-8952-7313; [b]0009-0002-5022-585X. [c]0000-0002-1417-4461; [d]0000-0003-1511-0109; [e]0000-0002-8039-2686; [f]0000-0002-0692-172X; [g]0000-0001-9859-1600.

**Abstract :** With the transition to the information society , all areas of our lives rapidly shift to the digital environment. From education to health, citizenship procedures to social life, all areas of our lives interact in the digital cyber environment. In this process, smart cities, smart networks, and smart factories, especially critical infrastructures required for social life, have become open to the intranet and the internet for efficient efficiency, speed, remote maintenance , and maintenance . Along with this process, these systems have faced new threat surfaces . One of the components that play an essential role in the operation of these systems is embedded systems. These systems contribute significantly to the effective operation of essential infrastructures. However, interruption in these systems can lead to significant negative consequences, including economic damage and human life. Although there are many studies on the functioning of embedded systems, there are not enough studies on the cyber security analysis of these systems . For this reason , attack and detection analyses for embedded systems have been carried out in this study on the test environment created using real systems. The study aims to detect passive attack, which is more difficult to detect than active attacks on the system, by using artificial intelligence algorithms . The analysis results have shown that the attack has been detected in a high ratio . It has been evaluated that the study will significantly contribute to other studies on the security of embedded systems.

**Keywords :** Embedded System Security, Vulnerability Analysis, Artificial Intelligence, Cyber Security, Attack Detection.

## 1 Introduction

With the ever-changing and developing technology, the number of devices used in businesses is diversifying and increasing daily. Many jobs that require human resources have become more accessible and more efficient with computer-based techniques and systems. Many system designs exist in computer-based systems, such as artificial intelligence, cloud technologies, embedded systems, sensors, communication, and network technologies. With the increase in mechanization in production activities, communication systems between machines are also developing. This is how the Internet of Things (IoT) emerged.

Another type of system used in computer-based system designs is embedded systems. With its widespread use, the importance of security in embedded systems has increased rapidly. Embedded system boards produced in many different brands and models are shaped for every purpose and budget. These easily accessible cards are the most widely used products, such as Raspberry Pi and Arduino.

Developing embedded system applications efficiently with these single-board computers has become possible. With the increased use of these systems, efficiency, low cost, and speed significantly contribute to human life. One of the most critical issues in this field is that the manufacturers of embedded systems, which are rapidly being used in areas that will affect human life, especially in industry and critical infrastructures, due to their benefits such as low cost, speed, and efficiency, are not made the necessary cyber security tests by the manufacturers. The necessary precautions are not taken to find vulnerabilities in cyber security.

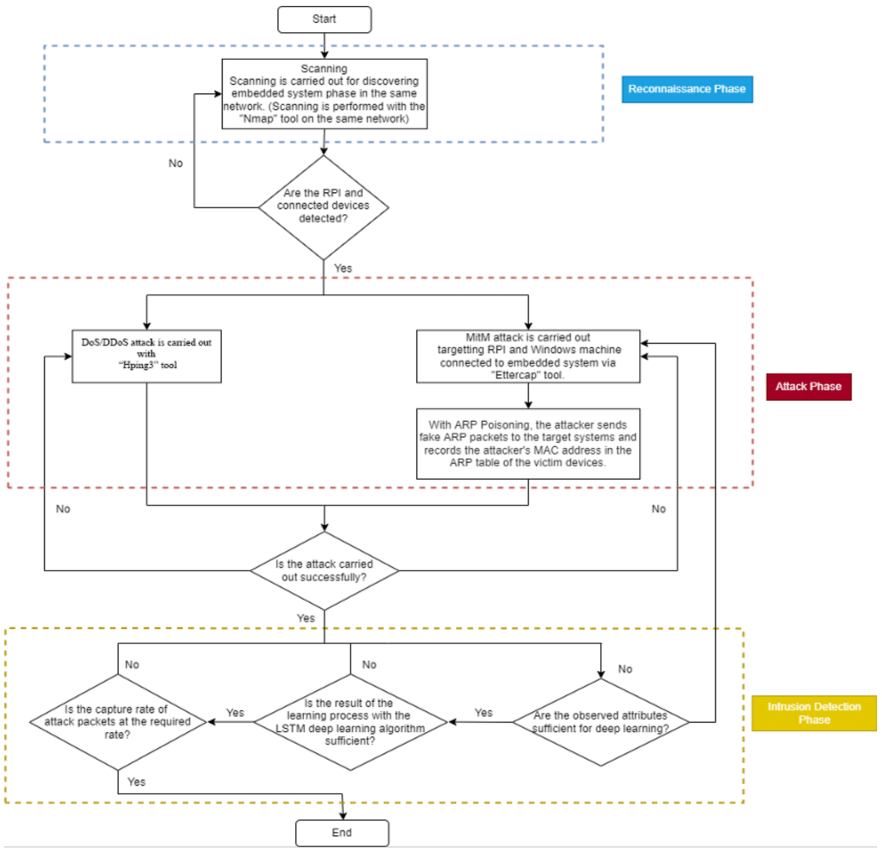

Start
Scanning
Scanning is carried out for discovering embedded system phase in the same network. (Scanning is performed with the "Nmap" tool on the same network)
Reconnaissance Phase
Are the RPI and connected devices detected?
No
Yes
DoS/DDoS attack is carried out with "Hping3" tool
MitM attack is carried out targetting RPI and Windows machine connected to embedded system via "Ettercap" tool.
With ARP Poisoning, the attacker sends fake ARP packets to the target systems and records the attacker's MAC address in the ARP table of the victim devices.
Attack Phase
No
No
Is the attack carried out successfully?
Yes
No
No
No
Is the capture rate of attack packets at the required rate?
Yes
Is the result of the learning process with the LSTM deep learning algorithm sufficient?
Yes
Are the observed attributes sufficient for deep learning?
Intrusion Detection Phase
Yes
End

**Figure 1: Flowchart of Tests Carried Out**

Since its development, Raspberry Pi has been used in large-scale projects such as innovative home solutions, robotic systems, and the Internet of Things. Its low cost, small size, and easy accessibility have increased the use of Raspberry Pi in embedded system applications. With this increase in its use in embedded system applications, the number of attackers who want to exploit the vulnerabilities of this single-board computer system has increased, especially in mission-critical air defense systems, medical devices, and industrial automation systems. Due to its widespread use in critical systems, it is vital to conduct cyber security analyses on embedded systems safely.

The analysis steps performed on the testbed developed in the study are depicted in Figure 1. A network scan has been carried out in the first stage of tests to determine whether there is any embedded system on the network. In this context, devices on the network have been scanned using the n-map tool.

Embedded systems/systems have been determined according to the parameters (brand, model, version number) used at the end of the scan. In the second stage, to test whether this detection is true or false positive, the embedded system detected in the field has been tested by listening to the network traffic with the man-in-the-middle attack. It has been seen that the same brand, model, and version number have been found on the Wireshark open-source network traffic analyzer. In the next stage, the Hping3 tool was used to distract the network administrators and carry out the DDoS attack, described as a cloaking attack before the actual final attack. The packets sent through the tool have prevented the network from serving legitimate users. While the network administrators have been dealing with this attack, the process killing attack, the final target attack, has been started. Process-killing attack gains importance because of process management, which is the purpose of embedded systems in many areas, such as smart factory, smart city, and critical infrastructures. While carrying out this attack, first of all, the transactions that have been managed and sent via the embedded system have been tracked. The most critical/ones of these processes have been identified and canceled. When DDoS and Process Kill attacks are performed, it is effortless to detect by the system administrator because their effects on the system can be detected when these attacks are carried out. However, the man-in-the-middle attack, called a passive attack, is complicated to detect because it does not leave any effect on the network. In our study, the focus of the attack detection is the detection of a man-in-the-middle attack, described as a passive attack with machine learning and artificial intelligence algorithms. Thanks to the algorithms used in the study, our attack could be detected with a success rate above the desired threshold value.

In the following parts of the study, respectively, In the second part, similar studies are presented. In the third part, the experimental environment in which the attack and analysis are carried out is explained, and the attack and the results of the attack are stated. In the fourth stage, machine learning and artificial intelligence are introduced for attack detection. Subsequently, studies on applications for detecting the man-in-the-middle attack are given. In the fifth section, the discussion section, general values have been discussed, and the study has been completed with the last stage, the conclusion section.

## 2 Literature Review

Embedded systems are widely used in various applications, and their security has become a critical concern due to increasing threats and vulnerabilities [1, 2]. Although significant research has been conducted on general-purpose computer security, embedded system security is an emerging area of study [3, 4]. Artificial intelligence (AI) can potentially improve security in embedded systems, and several researchers have explored AI-supported detection systems for addressing security challenges in embedded devices. Security-related research has become more important as the 2020-2021 global pandemic makes these systems even more critical. However, cyber threats to these systems started long before this popularity. Cyber attacks against embedded systems date back to 1982 [5]. Since most of these devices come with a non-changeable Randomly Generated Password (RCP) for their GUI, when the algorithm behind these RCPs is broken, attackers can access all the data they need to finish the attack [6].

Moreover, this type of attack is only one of the numerous attacks [7]. As these embedded devices become increasingly a part of our lives [8], the harm they can cause becomes even more critical. Tabrizi [9] proposed an AI-based intrusion detection system (IDS) algorithm for embedded systems, considering historical attacks, system security features, and factors validating these features. Özcanhan [10] presented a secure prototype implementation for embedded systems with security vulnerabilities, while Sungur [11] developed a hardware-independent Industrial IoT approach with speed, flexibility, and cost advantages. Arış, Oktuğ, and Yalçın [12] examined and classified Denial of Service Attacks targeting the Internet of Things, a domain that encompasses numerous embedded devices. In their study, Ni et al. [13] discussed the challenges in embedded system design, and Parameswaran and Wolf [14] emphasized the integration of security as a crucial component of embedded system design. Chaudhary et al. [15] provided background information on embedded systems, including their definition, history, architecture, usage areas, and programming languages. Gonen [16] conducted a security and vulnerability analysis on Siemens S-7 1200 (Firmware 2.2) PLC, examining the effects of targeted attacks on hardware and systems. Chen et al. [17] discussed network security and network attacks in embedded systems, suggesting that security protocols should be developed to strengthen embedded system networks. Jeremiah [18] explored the use, features, and applications of Raspberry Pi devices in cybersecurity. Mehra et al. [19] designed home security systems using Raspberry Pi devices to control camera and door lock systems remotely. Martin et al. [20] analyzed malware and its infection mechanisms for devices connected to the internet, demonstrating the risks of misconfigured Pi-based IoT devices. Lastly, Abomhara et al. [21] analyzed intruders and attacks encountered by IoT devices and services, classifying various threat types.

## 3 Phases of Attack

This section discusses the phases of the attack on the embedded system, accompanied by a scenario. In this context, first of all, the embedded systems on the network have been scanned, and then the man-in-the-middle attack has been carried out to confirm the accuracy of the detected system. After obtaining the confirmation of the embedded system and essential information about the characteristics of the system, a denial of service attack as a cloaking attack has been carried out to distract the security analysts/experts before ending the critical function of the system, which is the ultimate target of attack analysis.

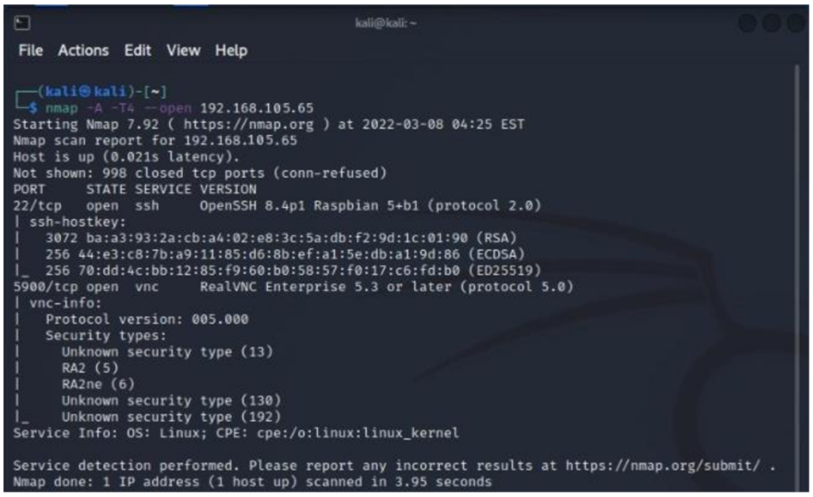
### 3.1 Attack Analysis

In the first phase of the attack analysis, the Nmap network scanning tool was used to detect embedded systems on the network. As a result of the scan, the IP address and information of the services that can be accessed on the target embedded system have been determined, as shown in Raspberry Pi Figure 2. This way, the system to be targeted in other attacks could be determined.

### 3.2 Man-in-the-Middle Attack

In the second phase of the attacks on the embedded system, the Man-in-the-Middle Attack (MitM) has been implemented. The man-in-the-middle attack is a type of attack where the malicious attacker enters between two communicating systems, listens to the communication, imitates both parties and reaches the information in between. MitM attack aims to damage confidentiality by eavesdropping on communication, integrity by intercepting communication and changing messages, and accessibility by destroying or modifying messages in a way that causes the termination of one of the parties [21]. This attack generally uses ARP (address resolution protocol) vulnerabilities. ARP is the protocol that helps map the IP address to the device's MAC address on the local network. [22].

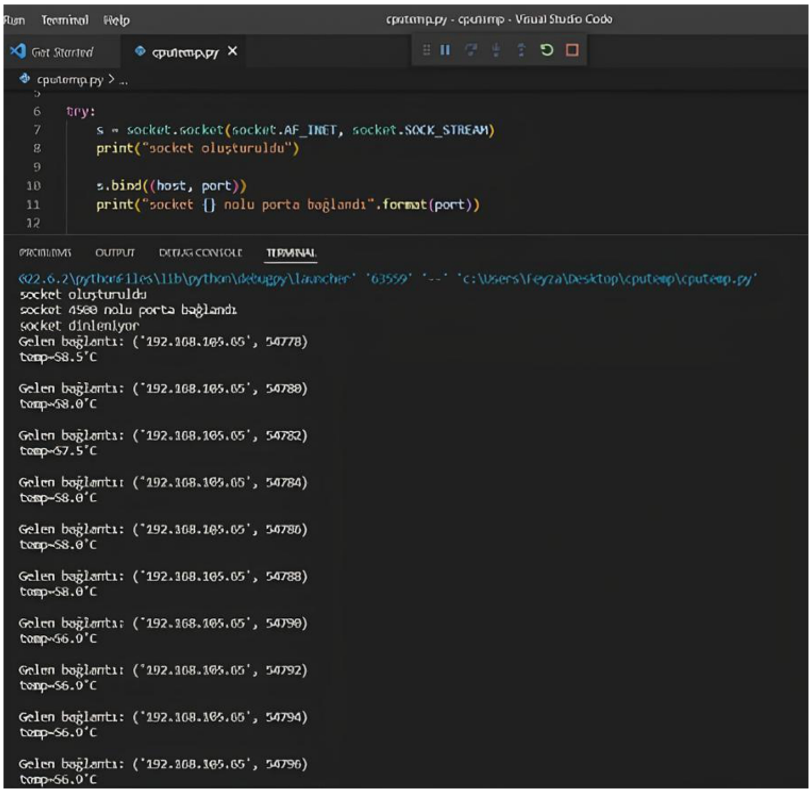
To perform the man-in-the-middle attack, the target systems were first scanned on the attacker operating system (Kali) to detect the IP addresses of Raspberry Pi and Windows systems. Subsequently, the IP addresses of the two victim computers have been defined as target systems on etthercap to perform ARP poisoning. At the last stage, ARP poisoning was performed, and the Kali System defined the arp values of both systems. This way, communication between Windows and the embedded system



```
┌──(kali㉿kali)-[~]
└─$ nmap -A -T4 --open 192.168.105.65
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-08 04:25 EST
Nmap scan report for 192.168.105.65
Host is up (0.021s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.4p1 Raspbian 5+b1 (protocol 2.0)
| ssh-hostkey:
|   3072 ba:a3:93:2a:cb:a4:02:e8:3c:5a:db:f2:9d:1c:01:90 (RSA)
|   256 44:e3:c8:7b:a9:11:85:d6:8b:ef:a1:5e:db:a1:9d:86 (ECDSA)
|_  256 70:dd:4c:bb:12:85:f9:60:b0:58:57:f0:17:c6:fd:b0 (ED25519)
5900/tcp open  vnc     RealVNC Enterprise 5.3 or later (protocol 5.0)
| vnc-info:
|   Protocol version: 005.000
|   Security types:
|     Unknown security type (13)
|     RA2 (5)
|     RA2ne (6)
|     Unknown security type (130)
|_    Unknown security type (192)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.95 seconds
```

**Figure 2: Results of Raspberry Pi vulnerability detection**



```
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print("socket oluşturuldu")

    s.bind((host, port))
    print("socket {} nolu porta bağlandı".format(port))

```

```
622.6.2\pythonFiles\lib\python\debugpy\launcher' '63559' '--' 'c:\Users\Feyza\Desktop\cputemp\cputemp.py'
socket oluşturuldu
socket 4500 nolu porta bağlandı
socket dinleniyor
Gelen bağlantı: ('192.168.105.65', 54778)
temp=58.5'C

Gelen bağlantı: ('192.168.105.65', 54780)
temp=58.0'C

Gelen bağlantı: ('192.168.105.65', 54782)
temp=57.5'C

Gelen bağlantı: ('192.168.105.65', 54784)
temp=58.0'C

Gelen bağlantı: ('192.168.105.65', 54786)
temp=58.0'C

Gelen bağlantı: ('192.168.105.65', 54788)
temp=58.0'C

Gelen bağlantı: ('192.168.105.65', 54790)
temp=56.0'C

Gelen bağlantı: ('192.168.105.65', 54792)
temp=56.0'C

Gelen bağlantı: ('192.168.105.65', 54794)
temp=56.0'C

Gelen bağlantı: ('192.168.105.65', 54796)
temp=56.0'C
```

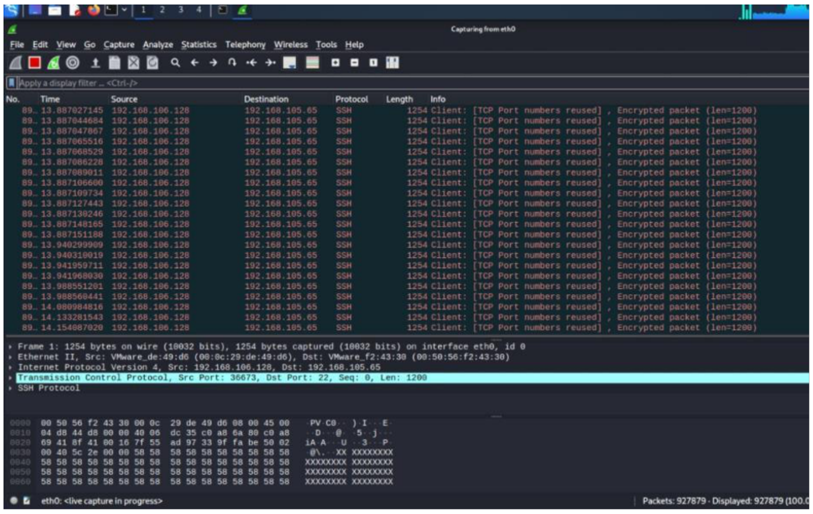
**Figure 3: Captured Data**

has continued throughout Kali. During the man-in- the-middle attack, the arp table was poisoned with the Ettercap tool. The data sent from the Raspberry Pi to the Windows machine has been captured as depicted in Figure. In this way, in addition to the legally sent commands on the embedded system, involuntary operations can be performed from the attacker's machine.

This attack has also been successfully completed, and the vulnerabilities of the Raspberry Pi and how it can be exploited are explained step by step.

## 3.3 Denial of Service Attack

In the third stage of the attack analysis, a Denial of Service (DoS) attack was carried out on the embedded system whose IP address was detected. A DoS attack is known as an attack to block or disrupt a running service. The primary purpose of DoS attacks is to put more load than the network, computer systems, and hardware resources such as bandwidth, memory, and disk space can handle, rendering the system unusable [23]. The hping3 tool has been used for the DoS attack. The state of the


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

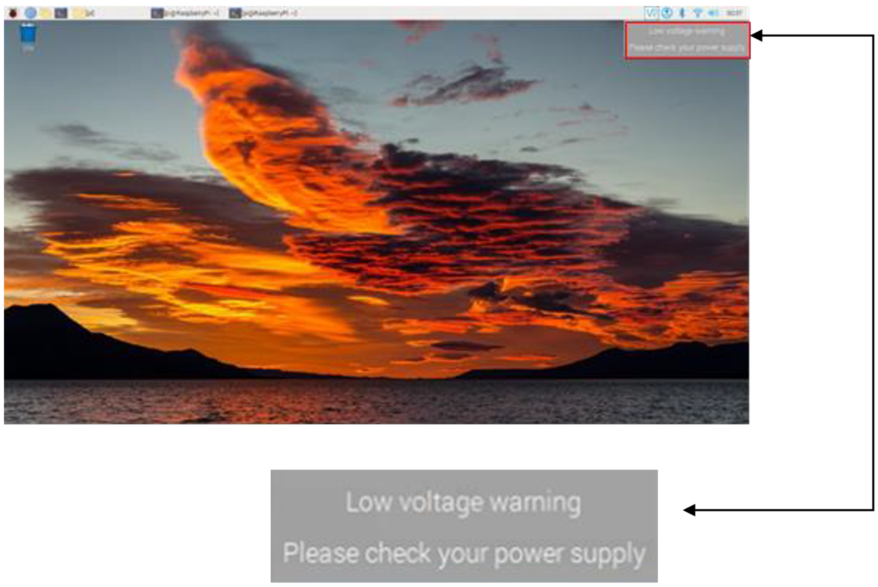
```
No.   Time          Source              Destination     Protocol  Length  Info
89... 13.887027145  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887044684  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887047867  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887065516  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887068529  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887086228  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887089011  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887106600  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887109734  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887127443  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887130246  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887148165  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.887151188  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.940299909  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.940310019  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.941959711  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.941968030  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.988551201  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 13.988560441  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 14.080984816  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 14.133281543  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)
89... 14.154087020  192.168.106.128     192.168.105.65  SSH       1254 Client: [TCP Port numbers reused] , Encrypted packet (len=1200)

› Frame 1: 1254 bytes on wire (10032 bits), 1254 bytes captured (10032 bits) on interface eth0, id 0
› Ethernet II, Src: VMware_de:49:d6 (00:0c:29:de:49:d6), Dst: VMware_f2:43:30 (00:50:56:f2:43:30)
› Internet Protocol Version 4, Src: 192.168.106.128, Dst: 192.168.105.65
› Transmission Control Protocol, Src Port: 36673, Dst Port: 22, Seq: 0, Len: 1200
› SSH Protocol

0000  00 50 56 f2 43 30 00 0c  29 de 49 d6 08 00 45 00   ·PV·C0·· )·I···E·
0010  04 d8 44 d8 00 00 40 06  dc 35 c0 a8 6a 80 c0 a8   ··D···@· ·5··j···
0020  69 41 8f 41 00 16 7f 55  ad 97 33 9f fa be 50 02   iA·A···U ··3···P·
0030  00 40 5c 2e 00 00 58 58  58 58 58 58 58 58 58 58   ·@\.··XX XXXXXXXX
0040  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0050  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX
0060  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX XXXXXXXX

eth0: <live capture in progress>                         Packets: 927879 · Displayed: 927879 (100.0
```

**Figure 4: Raspberry Pi Packet Traffic During a Denial of Service Attack**



Low voltage warning

Please check your power supply

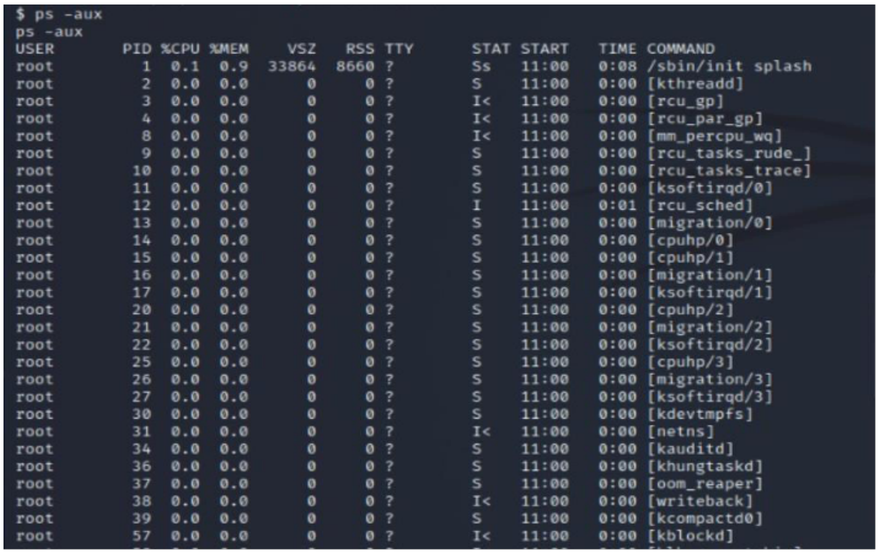
**Figure 5: Attack Alert**

Raspberry Pi before the attack was taken to determine the effect of the attack on the system. It has been observed that the CPU percentage of the target system was around 2.0% before the attack. After this detection, a DoS and DDoS attack was launched on the 22nd port of the target system with the hping3 test tool. During the Denial of Service attack, it was observed that Raspberry Pi's CPU utilization had been up to 90%. The packets flowing through the open-source packet analyzer Wireshark have been captured and analyzed to analyze the packets that occurred on the system during the Denial of Service attack. A sample screen of the analyzed packages is shown in Figure 4.

The Denial of Service attack has been successfully completed. After the attack, it was seen that the CPU usage of the Raspberry Pi increased, and the system gave a warning. A screenshot of the warning is shown in Figure 5.

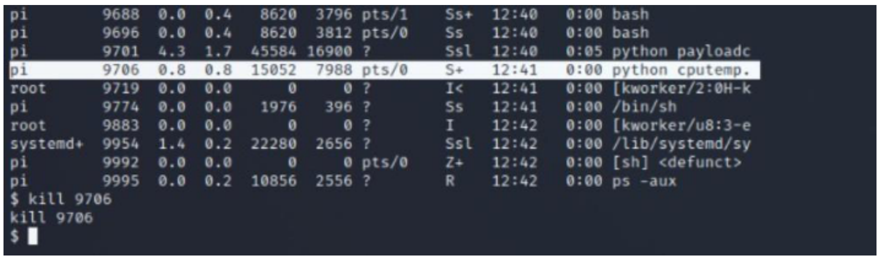
### 3.4 Process Killing Attack

In the Process Kill Attack, a Reverse TCP connection is based on the logic of opening a port on the attacker's side and transferring data by connecting the other machine to that port. This link sends an executable file that appears legitimate to the victim's machine. If the victim executes the file, the session is established, allowing the attacker to take control of the system remotely. In a process kill attack, the victim must allow the connection with their consent. This way, the connection is allowed even if the system has a firewall. After the victim is captured, the attacker sends the exploit packets to the victim. The victim system listens for exploit packets and allows connections that create the backdoor. Security systems, such as firewalls, provide strict protection to incoming packets but do not apply many protection measures against traffic within the network.



```
$ ps -aux
ps -aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.1  0.9  33864  8660 ?        Ss   11:00   0:08 /sbin/init splash
root         2  0.0  0.0      0     0 ?        S    11:00   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   11:00   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   11:00   0:00 [rcu_par_gp]
root         8  0.0  0.0      0     0 ?        I<   11:00   0:00 [mm_percpu_wq]
root         9  0.0  0.0      0     0 ?        S    11:00   0:00 [rcu_tasks_rude_]
root        10  0.0  0.0      0     0 ?        S    11:00   0:00 [rcu_tasks_trace]
root        11  0.0  0.0      0     0 ?        S    11:00   0:00 [ksoftirqd/0]
root        12  0.0  0.0      0     0 ?        I    11:00   0:01 [rcu_sched]
root        13  0.0  0.0      0     0 ?        S    11:00   0:00 [migration/0]
root        14  0.0  0.0      0     0 ?        S    11:00   0:00 [cpuhp/0]
root        15  0.0  0.0      0     0 ?        S    11:00   0:00 [cpuhp/1]
root        16  0.0  0.0      0     0 ?        S    11:00   0:00 [migration/1]
root        17  0.0  0.0      0     0 ?        S    11:00   0:00 [ksoftirqd/1]
root        20  0.0  0.0      0     0 ?        S    11:00   0:00 [cpuhp/2]
root        21  0.0  0.0      0     0 ?        S    11:00   0:00 [migration/2]
root        22  0.0  0.0      0     0 ?        S    11:00   0:00 [ksoftirqd/2]
root        25  0.0  0.0      0     0 ?        S    11:00   0:00 [cpuhp/3]
root        26  0.0  0.0      0     0 ?        S    11:00   0:00 [migration/3]
root        27  0.0  0.0      0     0 ?        S    11:00   0:00 [ksoftirqd/3]
root        30  0.0  0.0      0     0 ?        S    11:00   0:00 [kdevtmpfs]
root        31  0.0  0.0      0     0 ?        I<   11:00   0:00 [netns]
root        34  0.0  0.0      0     0 ?        S    11:00   0:00 [kauditd]
root        36  0.0  0.0      0     0 ?        S    11:00   0:00 [khungtaskd]
root        37  0.0  0.0      0     0 ?        S    11:00   0:00 [oom_reaper]
root        38  0.0  0.0      0     0 ?        I<   11:00   0:00 [writeback]
root        39  0.0  0.0      0     0 ?        S    11:00   0:00 [kcompactd0]
root        57  0.0  0.0      0     0 ?        I<   11:00   0:00 [kblockd]
```

**Figure 6: Listing Files Running on Raspberry Pi with Ps –Aux Command**



```
pi         9688  0.0  0.4   8620  3796 pts/1    Ss+  12:40   0:00 bash
pi         9696  0.0  0.4   8620  3812 pts/0    Ss   12:40   0:00 bash
pi         9701  4.3  1.7  45584 16900 ?        Ssl  12:40   0:05 python payloadc
pi         9706  0.8  0.8  15052  7988 pts/0    S+   12:41   0:00 python cputemp.
root       9719  0.0  0.0      0     0 ?        I<   12:41   0:00 [kworker/2:0H-k
pi         9774  0.0  0.0   1976   396 ?        Ss   12:41   0:00 /bin/sh
root       9883  0.0  0.0      0     0 ?        I    12:42   0:00 [kworker/u8:3-e
systemd+   9954  1.4  0.2  22280  2656 ?        Ssl  12:42   0:00 /lib/systemd/sy
pi         9992  0.0  0.0      0     0 pts/0    Z+   12:42   0:00 [sh] <defunct>
pi         9995  0.0  0.2  10856  2556 ?        R    12:42   0:00 ps -aux
$ kill 9706
kill 9706
$ █
```

**Figure 7: Killing the cputemp.py file running on Raspberry Pi with the Kill Command**

As the first step in the attack, the socket programming codes specified in the previous attack have been run to send the CPU temperature data from the Raspberry Pi to the Windows machine. The Metasploit tool is launched for the next stage of the attack. The payload starts listening from the port specified in the Metasploit tool. The session becomes active when the python payload we have created is run. The python/meterpreter/reverse_tcp payload is used on systems that support the python platform. Success in this attack largely depends on social engineering techniques. With the python command, the payload platform is in Python. With the meterpreter command, the payload is Meterpreter Shell opener, and the reverse_tcp command refers to the connection type. After the payload settings are made on Metasploit, a payload file to be run on Raspberry Pi is created to trigger the Meterpreter shell. The msfvenom tool has been used to generate the payload. Msfvenom is a tool for extracting files and shellcodes from an input payload. The created payload should be transferred to the target machine, Raspberry Pi, and run. Social engineering methods have been used to send the file to the target machine in the experimental environment. The payload has been copied to a USB memory stick, and this memory has been left lying around. A curious user has plugged it into the Raspberry Pi and ran it. However, different social engineering methods can be tried, such as creating a malicious link to run the malware on the target system.

After all these stages, the exploit tool has been run. At this point, the attacker had almost complete control over the target system. It was seen that the computer had been a Raspberry Pi. By switching to the command line of the system seized with the shell command, the desired operations can be performed from here. When the whoami command has been run, the ’pi’ response is output; when the pwd command is run, the working directory ’/home/pi’ is output. The files and folders in the system seized with the ls -l command are listed. All the files running on the target system with the ps –aux command have been listed in Figure 6.

The running code has been terminated by entering the process number with the kill command, and the data flow has been stopped as depicted in Figure 7.

When the process has been killed with the kill command on the attacking machine, The process has been terminated on the Raspberry Pi terminal, with the message ’terminated’. Thanks to this attack, it has been determined that social engineering methods can easily terminate a process running on Raspberry Pi, and this can cause great disasters in the industry sector. The Diagram of the attack can be seen in figure 8.

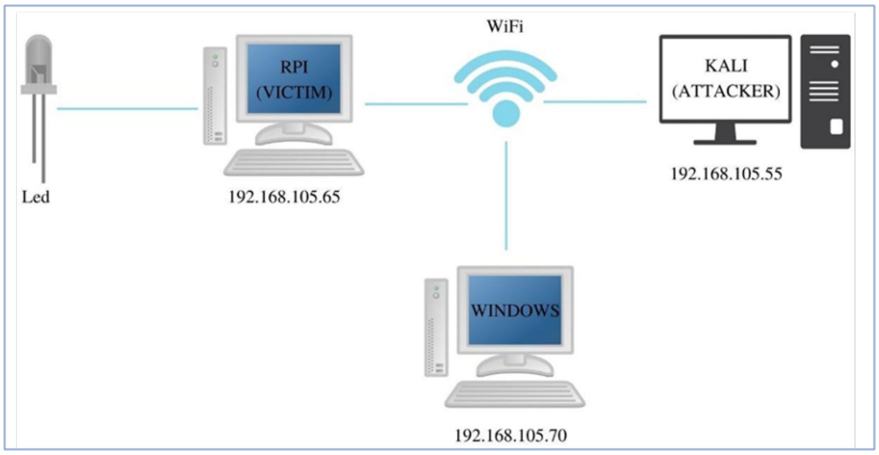


**Figure 8: Diagram of the Process Kill Attack**

## 4 Data Analysis

In this section, a comparative analysis of artificial intelligence (AI) algorithms used in the context of attacker detection is presented. The focus of the analysis is to evaluate the performance of AI algorithms other than the commonly used Long Short-Term Memory (LSTM) algorithm. The proposed AI-based intruder detection model consists of four stages.

The first stage involves preprocessing and organizing the collected network data into a suitable dataset for training AI algorithms. This dataset is then divided into 85% training data and 15% validation data and analyzed using various AI algorithms, including Neural Network (NN)-ReLU, Logistic Regression, Random Forest, Gradient Boosting, and Naive Bayes.

In the third stage, graphical tools such as Scatter Plot and Violin Plot are employed to gain a deeper understanding of the insights obtained from the AI algorithms. The final stage involves evaluating the performance of the AI algorithms and selecting the algorithm with the highest accuracy, F1 score, recall, and computational efficiency. In this stage, a confusion matrix is also created to evaluate the performance of the model by comparing the predicted and actual values of the target attribute (as shown in Figure 9).

The results of the comparative analysis indicate that the Random Forest AI algorithm was selected as the best performing algorithm, exhibiting the highest accuracy, F1 score, recall, and computational efficiency among the algorithms tested. This highlights the importance of considering multiple AI algorithms when designing an AI-based intruder detection model and the potential for improved performance through algorithm selection and optimization.

### 4.1 Long Short Term Memory (LSTM) Algorithm

An LSTM block consists of a cell and gates. There are three gates, including the entrance gate. These are listed as forget gate, input gate, and output gate. The cell remembers its value at random time intervals, and gates regulate the information flow associated with the cell. In short, it consists of repetitively connected subnets known as memory blocks. An LSTM block is shown, which includes gates, input signal xt, output yt, activation functions, and peephole connections. The output of the block connects back to the blog input and all ports.

Above the LSTM block are three doors, including the entrance door. These are listed as forget gate, input gate, and output gate.

Based on the information flow in figure 9, the mathematical model of LSTM is summarized as follows:

The update of the block input component, which combines the current input $x^t$ and the output $y^{t-1}$ of the LSTM unit in the last iteration, is allocated to the block input.

$$zt = g(W_z x_t + R_z y_{t-1} + b_z) \tag{1}$$

$W_z$ and $R_z$ components represent the weights associated with $x^t$ and $y^{t-1}$, respectively, and the bz component represents the bias weight vector. As a next step, we update the input gate by combining the current input xt, the output yt-1 of the LSTM unit, and the cell value ct-1 in the last iteration.

$$i^t = \sigma(W_i x^t + R_i y^{t-1} + p_i c^t - 1 + b_i) \tag{2}$$

$W_i$, $R_i$, and $P_i$ components represent the weights associated with $x^t$ , $y^{t-1}$, and $c^{t-1}$ components, respectively, and $b_i$ represents the bias vector associated with this component.

In previous steps, cell states in the network determine what information should be retained in $c^t$. It includes the selection of the values $z^{(t)}$ that can potentially be added to cell states and the activation values of the gates $i^t$. The forget gate unit determines

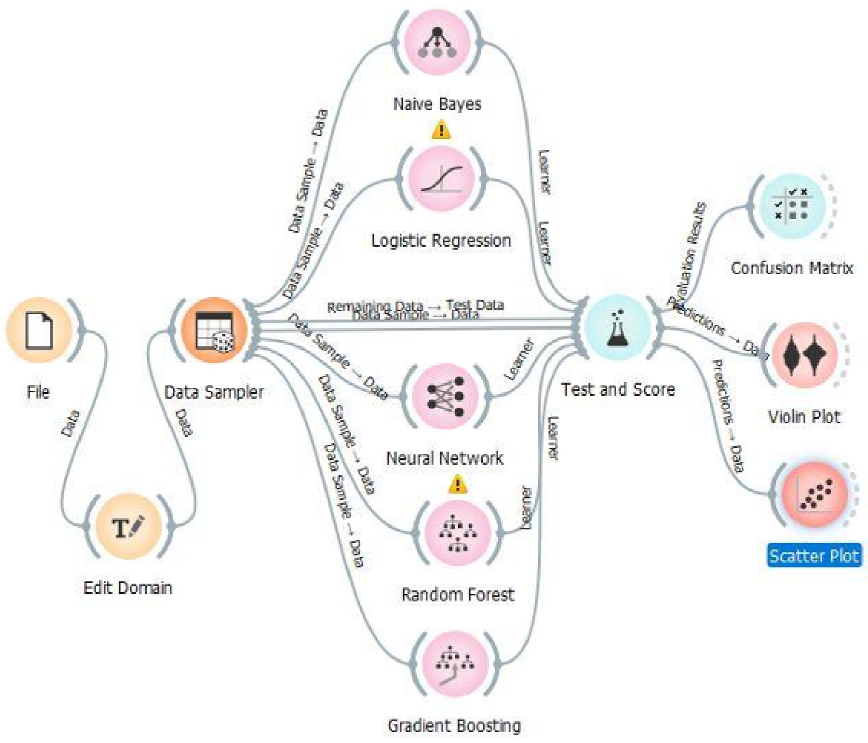


**Figure 9: Intrusion Detection with Artificial Intelligence Algorithms**

what information should be extracted from the previous cell states $c^{t-1}$. Therefore, at time step t, the forget gate activation values $f^t$ , the current input $x^t$, the outputs $y^{t-1}$, and the state of memory cells t-1 in the previous time step $c^{t-1}$, the peephole connections and the forget gate bias terms bf are calculated.

$$f_t = \sigma(W_f x^t + R_f y^{t^{-1}} + p_f c^t - 1 + b_f) \tag{3}$$

Here $W_f$, $R_f$, and $p_f$ denote the weights associated with $x^t$, $y^{t-1}$, and $c^{t-1}$, respectively, and bf the bias weight vector. Calculates the cell value by combining the block input $z^t$, the gate $i^t$, and forget gate $f^t$ together with the previous cell value.

$$c^t = f^t \cdot c^t - 1 + i^t \cdot z^t \tag{4}$$

Output gate calculates the output gate combining the current input xt, the output of the specified LSTM unit is yt- 1, and the cell value in the last iteration is t-1.

$$o^t = \sigma(W_o x^t + R_o y_{t-1} + p_o c^t + b_o) \tag{5}$$

Wo, Ro, and po are the weights associated with xt. The yt-1 and ct-1, respectively, and bo denotes the bias weight vector. Finally, the output of the block combining the current cell is calculated. Its value ct correlates with the current output gate value as follows:

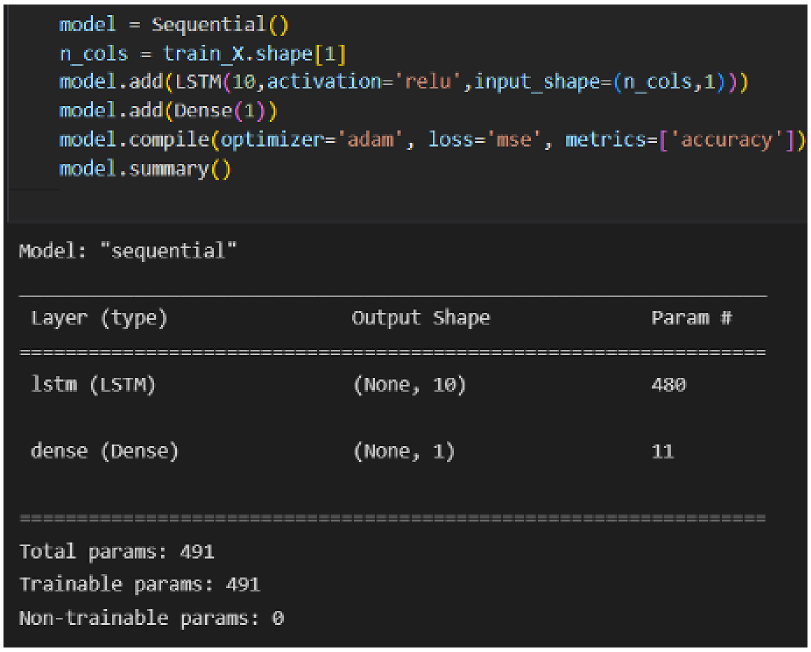
$$o^t = \sigma(W_o x^t + R_o y_{t-1} + p_o c^t + b_o) \tag{6}$$

The $\sigma$, g, and h points represent nonlinear functions. $\sigma(x) = 1/1+e^{1-x}$ is used as gate function, while g(x) = h(x) = tanh(x) is usually used as block input and output function.

## 4.2 Building and Training the Artificial Intelligence Model

This article separated data into 15% Tests and 85% training. We used four hundred ninety-one parameters, four hundred eighty in the LSTM layer and 11 in the dence layer ( Figure 10).

As seen in figure 11, we trained our model with 50 Epochs and 64 batches. In the first epoch, accuracy was seen as 62,32%, With a loss of 1152,39. The 50th epoch loss value was 0,083, and accuracy was 98,91%. In total loss was 0,00776, and the accuracy was 99%.



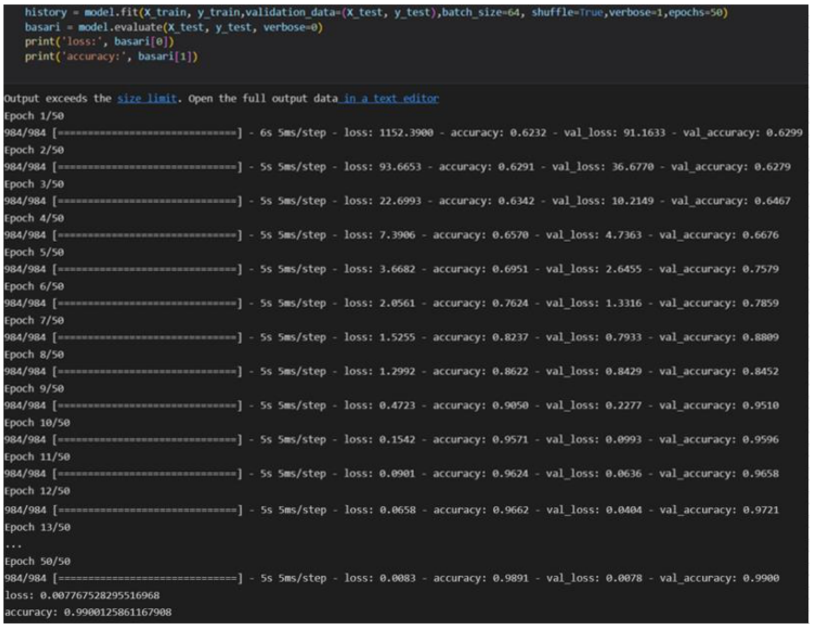
```
model = Sequential()
n_cols = train_X.shape[1]
model.add(LSTM(10,activation='relu',input_shape=(n_cols,1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10)                480

 dense (Dense)               (None, 1)                 11

=================================================================
Total params: 491
Trainable params: 491
Non-trainable params: 0
```

**Figure 10: Creating the Training Model, Adding the Necessary Information, and Compiling the Model**



```
history = model.fit(X_train, y_train,validation_data=(X_test, y_test),batch_size=64, shuffle=True,verbose=1,epochs=50)
basari = model.evaluate(X_test, y_test, verbose=0)
print('loss:', basari[0])
print('accuracy:', basari[1])
```

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/50
984/984 [==============================] - 6s 5ms/step - loss: 1152.3900 - accuracy: 0.6232 - val_loss: 91.1633 - val_accuracy: 0.6299
Epoch 2/50
984/984 [==============================] - 5s 5ms/step - loss: 93.6653 - accuracy: 0.6291 - val_loss: 36.6770 - val_accuracy: 0.6279
Epoch 3/50
984/984 [==============================] - 5s 5ms/step - loss: 22.6993 - accuracy: 0.6342 - val_loss: 10.2149 - val_accuracy: 0.6467
Epoch 4/50
984/984 [==============================] - 5s 5ms/step - loss: 7.3906 - accuracy: 0.6570 - val_loss: 4.7363 - val_accuracy: 0.6676
Epoch 5/50
984/984 [==============================] - 5s 5ms/step - loss: 3.6682 - accuracy: 0.6951 - val_loss: 2.6455 - val_accuracy: 0.7579
Epoch 6/50
984/984 [==============================] - 5s 5ms/step - loss: 2.0561 - accuracy: 0.7624 - val_loss: 1.3316 - val_accuracy: 0.7859
Epoch 7/50
984/984 [==============================] - 5s 5ms/step - loss: 1.5255 - accuracy: 0.8237 - val_loss: 0.7933 - val_accuracy: 0.8809
Epoch 8/50
984/984 [==============================] - 5s 5ms/step - loss: 1.2992 - accuracy: 0.8622 - val_loss: 0.8429 - val_accuracy: 0.8452
Epoch 9/50
984/984 [==============================] - 5s 5ms/step - loss: 0.4723 - accuracy: 0.9050 - val_loss: 0.2277 - val_accuracy: 0.9510
Epoch 10/50
984/984 [==============================] - 5s 5ms/step - loss: 0.1542 - accuracy: 0.9571 - val_loss: 0.0993 - val_accuracy: 0.9596
Epoch 11/50
984/984 [==============================] - 5s 5ms/step - loss: 0.0901 - accuracy: 0.9624 - val_loss: 0.0636 - val_accuracy: 0.9658
Epoch 12/50
984/984 [==============================] - 5s 5ms/step - loss: 0.0658 - accuracy: 0.9662 - val_loss: 0.0404 - val_accuracy: 0.9721
Epoch 13/50
...
Epoch 50/50
984/984 [==============================] - 5s 5ms/step - loss: 0.0083 - accuracy: 0.9891 - val_loss: 0.0078 - val_accuracy: 0.9900
loss: 0.007767528295516968
accuracy: 0.9900125861167908
```

**Figure 11: Training the Model and the Results**

## 5 Discussion

The attack and detection analyzes of the study were carried out on the testbed created using real systems. Attack analyzes were carried out within the scope of a designed attack scenario. In the scenario, first of all, the embedded system/systems on the network were identified, and then the verification of the detected system/systems was carried out. At this stage, critical sensitive information about the embedded system was also obtained. Critical processes controlled by the embedded system/systems were identified, and a DoS attack was carried out primarily for cloaking in order to prevent the operation of these processes. In this way, the attention of system security experts was distracted and the critical process(s) managed by the embedded system, which is the ultimate target of the attack phase, was prevented. In the attack detection analysis phase of the study, the focus was on the detection of the MitM attack, which is a passive attack type, through artificial intelligence. Although active attacks have an impact on the system and are easier to detect by security analysts, passive attacks are much more difficult to detect and affect the system. For this reason, passive attacks are directed towards detection analyzes. The analysis results showed that the MitM attack was successfully detected through artificial intelligence.

## 6 Conclusions

Quality control gained importance in the 20th century with the concepts such as efficiency and productivity that came to the fore with the industrial revolution. Embedded systems have been used to integrate computer systems into production with the developing technologies. The security aspect is of paramount importance for embedded systems that are ubiquitous and widely used in a variety of complex applications in industries, automobiles, consumer devices, home automation, business infrastructure and military hardware. Embedded systems, which are designed to work uninterruptedly and continuously with a real-time response mechanism, are an important target for attackers because the necessary importance is not given to the cyber security. Therefore, the primary purpose of this study is to determine the effects of the threat surface on the system, which will be created by the vulnerabilities of these systems in our world, which is increasingly automated with the use of embedded devices, and to keep the system active by detecting the attacks as soon as possible. As a result of the attacks made in the experimental environment, it has been seen that the exploitation of these vulnerable devices can be perceived as a real threat and the security of embedded devices can be violated.

The attack and detection stages of the study have been carried out on the testbed, which has been created using completely real systems. In the attack analysis part of the study carried out according to scenario. In the first phase of the scenario, embedded systems on the network have been detected primarily. To verify the detected embedded systems by performing a man-in-the-middle attack. In the phase, information about the system also has been obtained for further attacks. In the third phase, a cloaking attack, a denial-of-service attack, has been carried out to distract security analysts before disrupting critical functions of the system. At the last stage of the attack scenario, critical processes performed by the embedded system have been stopped. In the analysis of intrusion detection through artificial intelligence, the study focuses on the man-in-the-middle attack, a passive attack whose detection and effect on the system are hardly noticed by security experts. Intrusion detection results have been detected with a rate of 99% of the passive attack through artificial intelligence.

## Authors' Contributions

FA designed the system, FA and BA made primary data collection, and designed attack system SG, MAB, and HHS made analysis of data SG, ENY and MAB setup machine learning systems and test the collected data.

## Competing Interests

The authors declare that they have no competing interests.

## References

[1] M. Jiménez, R. Palomera, and I. Couvertier. *Introduction to Embedded Systems*. Springer, New York, NY, 2014. .

[2] D. N. Serpanos and A. G. Voyiatzis. Security challenges in embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(1s):1–10, 2013. .

[3] A. Farmahini-Farahani, S. Vakili, S. M. Fakhraie, S. Safari, and C. Lucas. Parallel scalable hardware implementation of asynchronous discrete particle swarm optimization. *Engineering Applications of Artificial Intelligence*, 23(2):177–187, 2010. .

[4] P. K. Muhuri, A. K. Shukla, and A. Abraham. Industry 4.0: A bibliometric analysis and detailed overview. *Engineering applications of artificial intelligence*, 78:218–235, 2019. .

[5] M. Keefe. Timeline: Critical infrastructure attacks increase steadily in past decade. *Computerworld*, 5, 2012.

[6] L. Apa and C. M. Penagos. Compromising industrial facilities from 40 miles away. *IOActive Technical White Paper*, 2013.

[7] D. Papp, Z. Ma, and L. Buttyan. Embedded systems security: Threats, vulnerabilities, and attack taxonomy. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, pages 145–152, 2015. .

[8] L. C. De Silva, C. Morikawa, and I. M. Petra. State of the art of smart homes. *Engineering Applications of Artificial Intelligence*, 25(7):1313–1321, 2012. .

[9] F. Molazem Tabrizi. *Security analysis and intrusion detection for embedded systems: a case study of smart meters*. PhD thesis, Electrical and Computer Engineering, University of British Columbia, 2017.

[10] M. H. Özcanhan. *Security and reliability in embedded systems*. PhD thesis, Computer Engineering and Computer Science and Control, Dokuz Eylul University, 2011.

[11] G. Sungur and B. Barış. Labview and embedded system based new iot solution for industrial applications. *Academic Platform Journal of Engineering and Smart Systems*, 10(2):106–114, 2022. .

[12] A. Arış, S. F. Oktuğ, and S. B. Ö. Yalçın. Internet-of-things security: Denial of service attacks. In *2015 23nd Signal Processing and Communications Applications Conference (SIU)*, pages 903–906, 2015. .

[13] S. Ni, Y. Zhuang, J. Gu, and Y. Huo. A formal model and risk assessment method for security-critical real-time embedded systems. *Computers & Security*, 58:199–215, 2016. .

[14] S. Parameswaran and T. Wolf. Embedded systems security—an overview. *Design Automation for Embedded Systems*, 12: 173–183, 2008.

[15] P. Chaudhary, B. B. Gupta, and A. Singh. Securing heterogeneous embedded devices against xss attack in intelligent iot system. *Computers & Security*, 118:102710, 2022. .

[16] S. Gonen. *Providing information security in micro type smart grid systems controlled by PLC*. PhD thesis, Graduate School of Natural and Applied Sciences, Gazi University, 2018.

[17] D. D. Chen, M. Woo, D. Brumley, and M. Egele. Towards automated dynamic analysis for linux-based embedded firmware. In *NDSS*, volume 1, pages 1.1–8.1, 2016.

[18] J. Jeremiah. Intrusion detection system to enhance network security using raspberry pi honeypot in kali linux. In *2019 International Conference on Cybersecurity (ICoCSec)*, 2019.

[19] M. Mehra, V. Sahai, P. Chowdhury, and E. Dsouza. Home security system using iot and aws cloud services. In *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)*, pages 1–6. IEEE, 2019. .

[20] R. Martin, K. Keller, and H. Mathews. Development of resistance to raspberry bushy dwarf virus in 'meeker' red raspberry. In *X International Symposium on Small Fruit Virus Diseases 656*, pages 165–169, 2003. .

[21] N. B. Al Barghuthi, M. Saleh, S. Alsuwaidi, and S. Alhammadi. Evaluation of portable penetration testing on smart cities applications using raspberry pi iii. In *2017 Fourth HCT Information Technology Trends (ITT)*, pages 67–72. IEEE, 2017. .

[22] M. Abomhara and G. M. Køien. Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security and Mobility*, pages 65–88, 2015. .

[23] J. Mirkovic and P. Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.