01.Introduction

1.1Why IOT Devices?

Increasingly, IoT devices are using AI and machine learning to bring intelligence and autonomy to systems and processes, such as autonomous driving, industrial smart manufacturing, medical equipment, and home automation. Many of these devices are small, power- and cost-constrained microcontroller-based systems. Network bandwidth and consumer expectations around data privacy and user experience continue to demand more on-device processing, where data is processed on the IoT endpoint, rather than using cloud-based approaches. (arm)

1.2 What are the IOT Devices?

Internet of things (IoT) devices are nonstandard computing hardware -- such as sensors, actuators or appliances -- that connect wirelessly to a network and can transmit data.

IoT extends internet connectivity beyond typical computing devices -- such as desktops, laptops, smartphones and tablets -- to any range of traditionally dumb or non-internet-enabled physical devices and everyday objects. Embedded with technology, these devices can communicate and interact over the internet, and can be remotely monitored and controlled.

IoT devices have both industrial and consumer uses and are typically integrated into other tools such as mobile devices, industrial equipment and medical devices. Over a broad range, they can also be used in smart cities. They're then used to send data or interact with other IoT devices over a network.

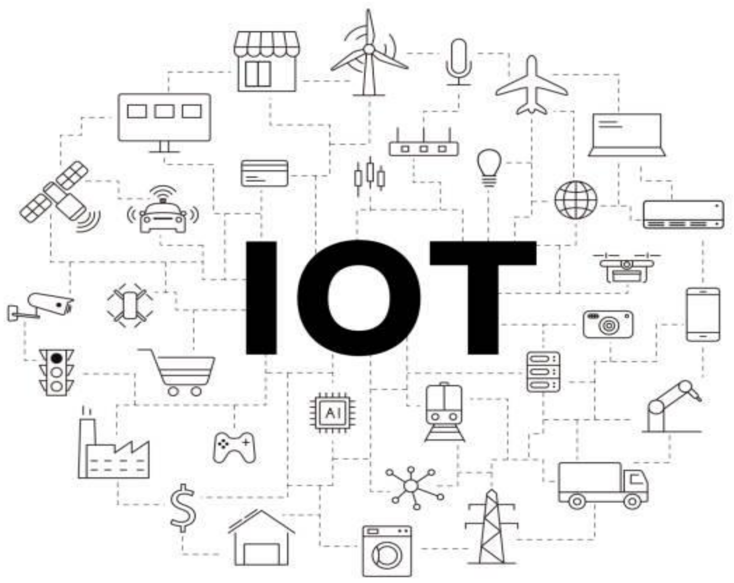
IoT and IoT devices aid in making daily activities faster, easier or more convenient for consumers while also providing real-time data for industrial or enterprise use cases.

(S.Gillis)

1.3 Why IOT Devices so Important?

Over the past few years, IoT has become one of the most important technologies of the 21st century. Now that we can connect everyday objects—kitchen appliances, cars, thermostats, baby monitors—to the internet via embedded devices, seamless communication is possible between people, processes, and things.

By means of low-cost computing, the cloud, big data, analytics, and mobile technologies, physical things can share and collect data with minimal human intervention. In this hyperconnected world, digital systems can record, monitor, and adjust each interaction between connected things. The physical world meets the digital world—and they cooperate.



1.4 what are the benefits from IOT devices?

Manufacturing

Manufacturers can gain a competitive advantage by using production-line monitoring to enable proactive maintenance on equipment when sensors detect an impending failure. Sensors can actually measure when production output is compromised. With the help of sensor alerts, manufacturers can quickly check equipment for accuracy or remove it from production until it is repaired. This allows companies to reduce operating costs, get better uptime, and improve asset performance management.

Automotive

The automotive industry stands to realize significant advantages from the use of IoT applications. In addition to the benefits of applying IoT to production lines, sensors can detect impending equipment failure in vehicles already on the road and can alert the driver with details and recommendations. Thanks to aggregated information gathered by IoT-based applications, automotive manufacturers and suppliers can learn more about how to keep cars running and car owners informed.

Retail

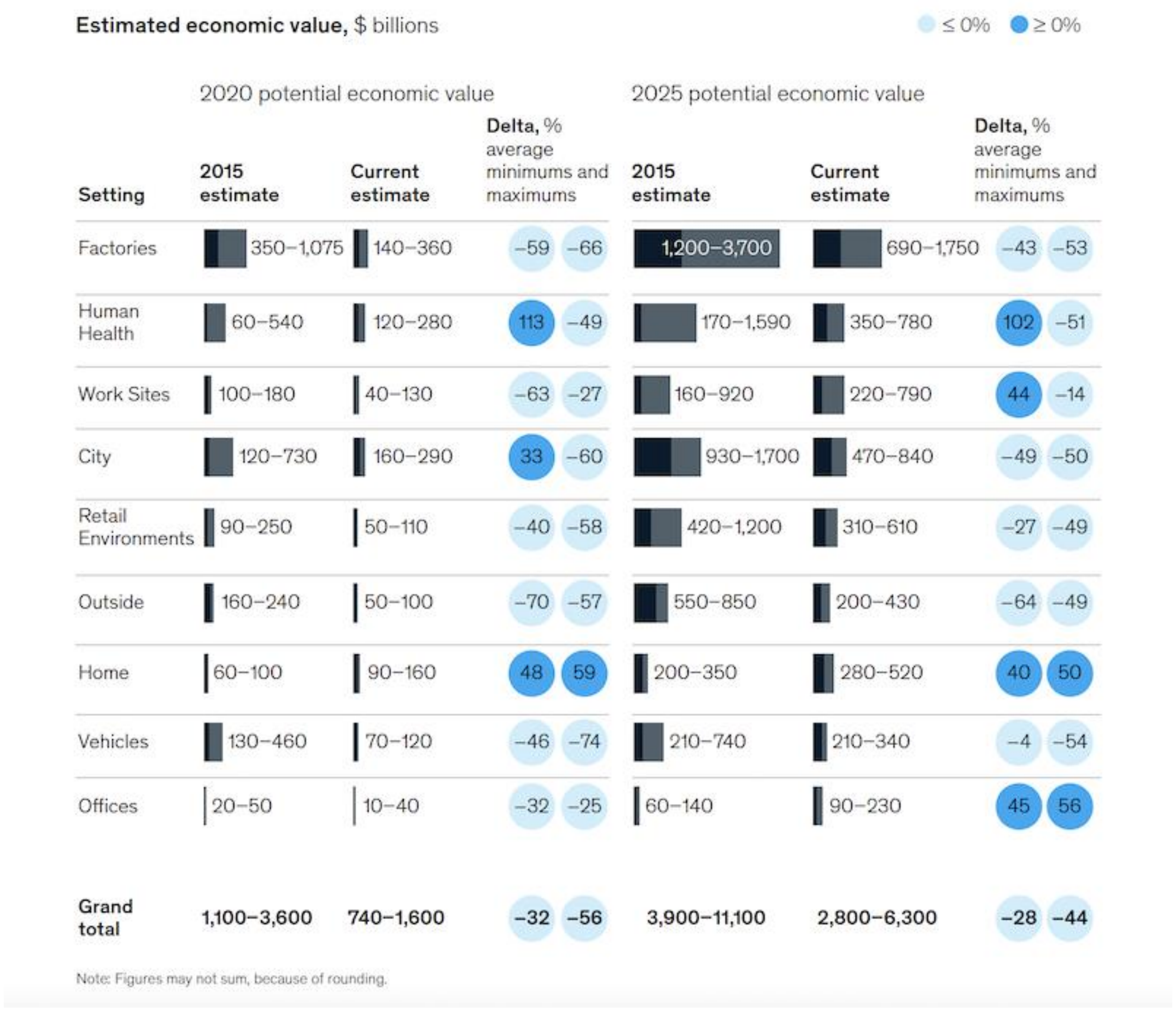
IoT applications allow retail companies to manage inventory, improve customer experience, optimize supply chain, and reduce operational costs. For example, smart shelves fitted with weight sensors can collect RFID-based information and send the data to the IoT platform to automatically monitor inventory and trigger alerts if items are running low. Beacons can push targeted offers and promotions to customers to provide an engaging experience.

Healthcare

IoT asset monitoring provides multiple benefits to the healthcare industry. Doctors, nurses, and orderlies often need to know the exact location of patient-assistance assets such as wheelchairs. When a hospital's wheelchairs are equipped with IoT sensors, they can be tracked from the IoT asset-monitoring application so that anyone looking for one can quickly find the nearest available wheelchair. Many hospital assets can be tracked this way to ensure proper usage as well as financial accounting for the physical assets in each department.

General Safety Across All Industries

In addition to tracking physical assets, IoT can be used to improve worker safety. Employees in hazardous environments such as mines, oil and gas fields, and chemical and power plants, for example, need to know about the occurrence of a hazardous event that might affect them. When they are connected to IoT sensor–based applications, they can be notified of accidents or rescued from them as swiftly as possible. IoT applications are also used for wearables that can monitor human health and environmental conditions. Not only do these types of applications help people better understand their own health, they also permit physicians to monitor patients remotely.



Estimated economic value, $ billions

● ≤ 0% ● ≥ 0%

	2020 potential economic value				2025 potential economic value			
Setting	**2015 estimate**	**Current estimate**	**Delta,** % average minimums	maximums	**2015 estimate**	**Current estimate**	**Delta,** % average minimums	maximums
Factories	350–1,075	140–360	–59	–66	1,200–3,700	690–1,750	–43	–53
Human Health	60–540	120–280	113	–49	170–1,590	350–780	102	–51
Work Sites	100–180	40–130	–63	–27	160–920	220–790	44	–14
City	120–730	160–290	33	–60	930–1,700	470–840	–49	–50
Retail Environments	90–250	50–110	–40	–58	420–1,200	310–610	–27	–49
Outside	160–240	50–100	–70	–57	550–850	200–430	–64	–49
Home	60–100	90–160	48	59	200–350	280–520	40	50
Vehicles	130–460	70–120	–46	–74	210–740	210–340	–4	–54
Offices	20–50	10–40	–32	–25	60–140	90–230	45	56
Grand total	**1,100–3,600**	**740–1,600**	**–32**	**–56**	**3,900–11,100**	**2,800–6,300**	**–28**	**–44**

Note: Figures may not sum, because of rounding.

1Economic-value estimates by settings for IoT devices – McKinsey.

02. How they work

2.1 IOT

Many IoT devices gather information via their sensors and then use software to analyse it and determine what decisions to make based on the data.

These devices usually connect to a central server to get more information. They also compare and transmit data to public websites and services to collect data, as well as connect to a messaging server that can email, text or call. IoT devices can also connect to other IoT devices via the same wireless network to instruct them on what to do.

Although IoT devices can make tasks easier, more convenient or expand the capabilities of the task, these devices can also pose a security and safety threat if they are hacked or compromised.

2.2 IOT Device Technologies

IOT devices utilize a variety of technologies to enable communication, sensing and interaction with the environment. Some of the key technologies used in IoT devices include:

- **Low-energy wireless protocols:**

  Bluetooth
  ZigBee
  NFC (Near Field Communication)
  LTE (Long-Term Evolution)
  NB-IoT (Narrowband Internet of Things)

- Machine learning and artificial intelligence (AI) for data processing and analysis.
- Sensors for environmental monitoring and data collection (ex: temperature, humidity, pressure, motion)
- Internet connectivity protocols (ex: HTTP, CoAP, MQTT) for communication with the cloud and other devices.
- Power management technologies to minimize power consumption and extend battery life.
- Secure communication protocols (ex: SSL/TLS, DTLS) for data encryption and authentication.

Additionally, some IoT devices may employ other technologies, such as:

- GPS (Global Positioning System) for locating tracking.
- RFID (Radio-Frequency Identification) for asset tracking and inventory management.
- Cellular connectivity (ex: 2G, 3G, 4G, 5G) for wide-area networking.
- Wi-Fi and other wireless local area network (WLAN) technologies for local connectivity.
- Analog-to-Digital converters (ADCs) and digital-to-analog converters (DACs) for signal processing and conversion.

These are the technologies use for IoT devices. In the future AI automation solutions will be coming to IoT devices. So, we have to update with new technologies and patterns.

When we study with this IoT devices sometimes they must be compromised by attackers. Our sensitive data like video footages, pictures, credentials, details of employees can be compromised.

03. vulnerabilities and attacks can be exploited to IoT devices

The Internet of Things (IoT) is increasingly permeating modern society, from end-users to enterprises and industrial usage. The rapid growth in connected IoT devices creates many possibilities, but it also introduces significant cybersecurity risks.

A vulnerable device can risk IoT security by giving cyber criminals access to connected networks, enabling them to steal critical corporate data and user credentials. Organizations therefore must understand how to secure IoT devices and recognize the top IoT vulnerabilities they face. (FORTINET)

3.1. Cyber Attacks

- **Botnet Attacks**:

  What is a Botnet?

  Professionals, who make system security arrangements, are well-aware of the term ‘botnet’. Often used for the chain of hijacked computers/systems, the term ‘botnet’ should be well understood if a restorative and robust system is instructed as their wrong usage can lead to tremendous chaos. (Lee)

  Botnet attacks use a command-and-control (C&C) model to allow one or more hackers to drive the actions of those devices (often called ‘zombie bots’) from a remote location. The more devices that have been infected with the attacker’s malware, the stronger the attack is likely to be.

Any device capable of accessing the internet could be used as a zombie bot in a botnet attack that puts enterprises in jeopardy. This is especially true if the device doesn't receive regular antivirus software updates.

All of the five main classes of Internet of Things (IoT) applications can present security risks, including consumer, commercial, industrial, smart city infrastructure, or military arenas. In each of those fields, the market is flooded with IoT devices, many of which are lacking in security.

How does Botnet Attack work?

Botnet attacks can be carried out by a single person or team. Either way, a force of zombie bots is controlled by the bot herder, which is the individual or group driving the attack. The bot herder can build their own botnet from scratch or rent it from other bad actors (sometimes dubbed "malware-as-a-service," or Maas).

Once infected, zombie bots are anonymously controlled via a centralized client-server model or decentralized peer-to-peer (P2P) model.

Botnet attacks model:

1) Centralized model
2) Decentralized model

Centralized model

A centralized botnet attack is executed by a single server functioning as the bot herder. There may be a hierarchy of proxy or sub-herding servers set up under it, but the commands originate from the bot herder server.

The centralized approach is a bit outdated. As you can imagine, identifying and shutting down one centralized server is much easier than locating and stopping an attack when the commands are deployed by multiple zombie bots.

Decentralized model (P2P)

In a decentralized botnet attack, the responsibility for giving instructions is embedded across all the bots in the botnet. If the attacker can communicate with any one of them, the malware can still be propagated through the other hijacked devices.

As you can imagine, the P2P framework makes it much harder to identify the person or people in control. Because of this, the decentralized model is much more widely used.

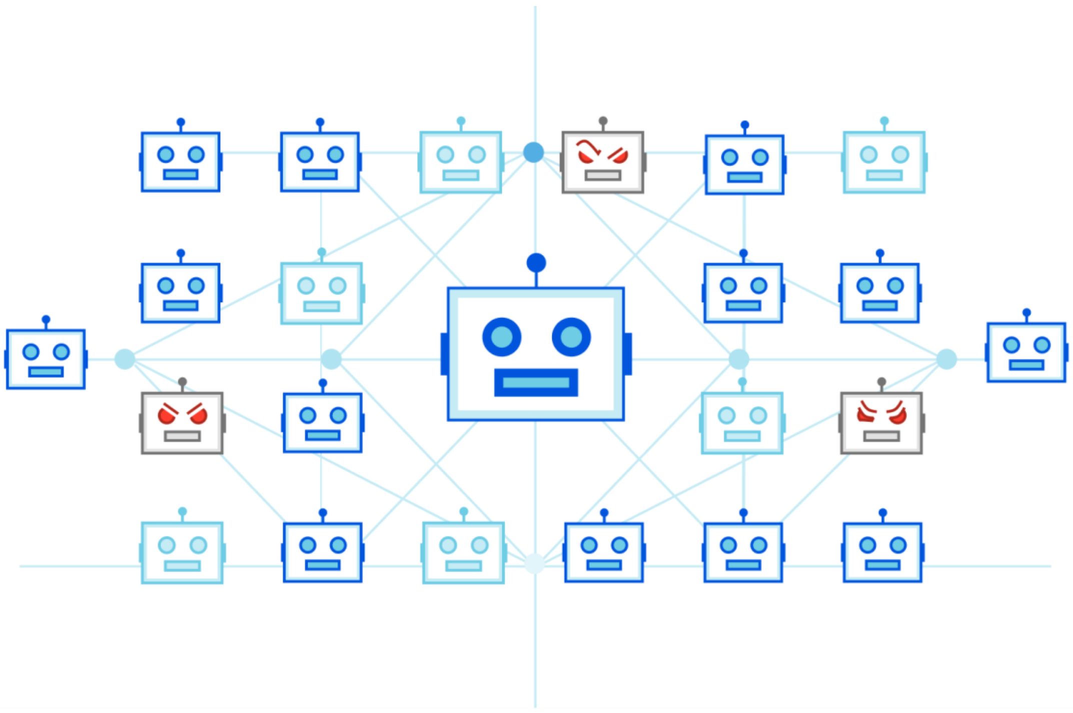


Figure 2 Botnet Attack.

Exploitations to Botnet

Pentesters and hackers are using vulnerabilities to enter into devices. Vulnerabilities are assigned values according to their risk. It is called the **CVE** number. **CVE** stands for common vulnerabilities exposure. Many hackers find the exploitation related to the cve number, create the relevant codes and necessary tools from it and launch attack using them.

1. **CVE-2021-28372**: A critical vulnerability in the ThroughTek Kalay platform's SDK allows remote compromise and control of millions of IoT devices, including cameras, smart baby monitors, and DVRs. The vulnerability was discovered in late 2020 and patched in version 3.1.10. Attackers can register a device on the network with a victim's UID, allowing them to obtain authentication materials and access the device.

2. **CVE-2022-29555 and CVE-2022-29556**: Two vulnerabilities in Mender's IoT-manager microservice and device connect, respectively,

allow Server-Side Request Forgery (SSRF) and cross-tenant actions via internal API endpoints. The vulnerabilities were patched in Mender 3.2.2.

3. **CVE-based classification of vulnerable IoT systems**: A research paper proposing a real-world-based classification of IoT systems using the Common Vulnerabilities and Exposures (CVE) database. The authors employ a Support Vector Machine (SVM) algorithm to classify new vulnerability records.

These are the examples for vulnerabilities to identify how attackers can enter to the IoT devices. Botnet attacks, DDOS, remote code shell executions, user credential using Brute force attacks are the major and they are help to gain access to hackers to IoT devices.

- **Cve-2021-28372 exploit:**

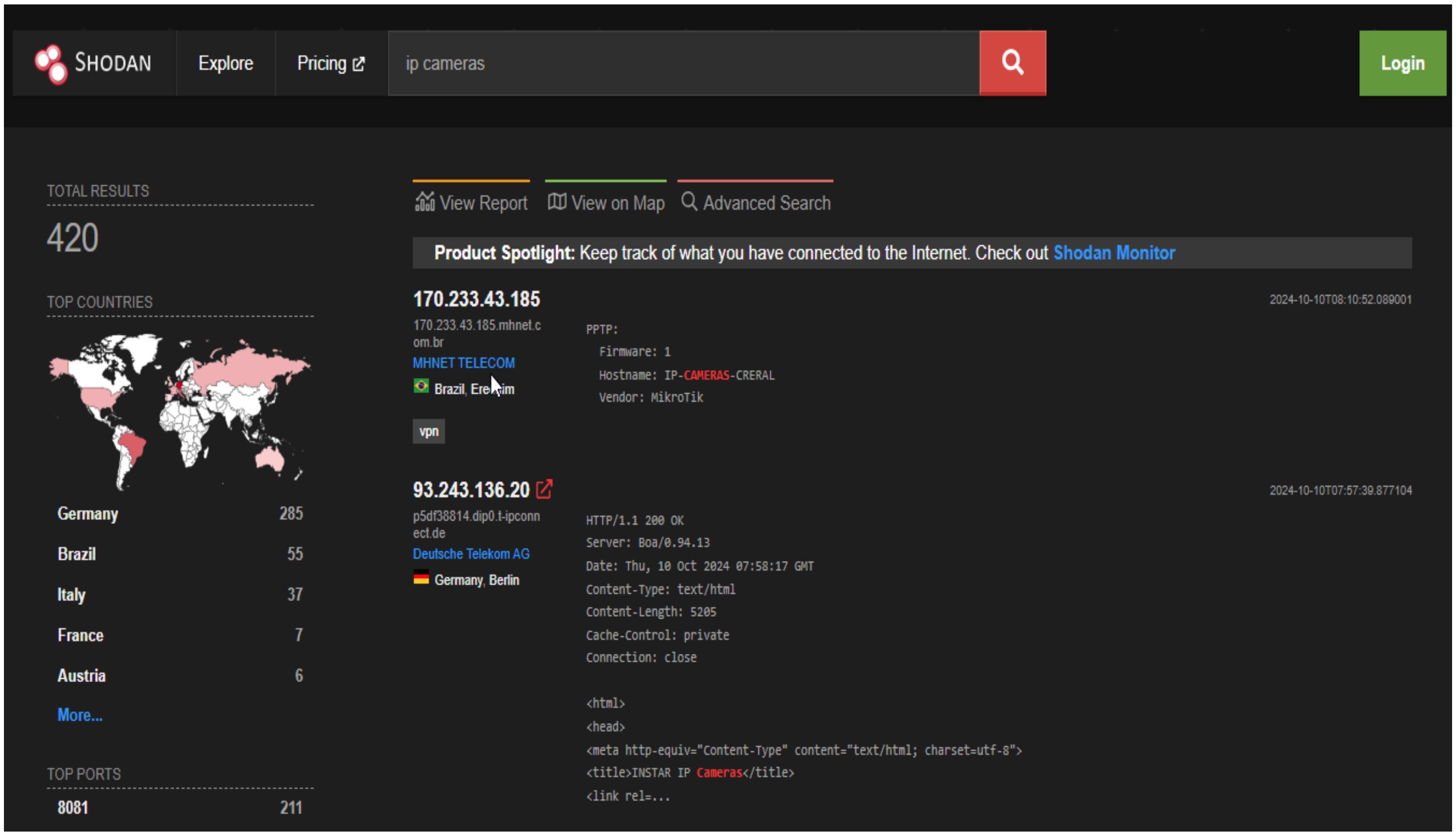
  CVE-2021-28372 is a remote code execution (RCE) vulnerability in the ThroughTek Kalay P2P software development kit (SDK) used by many IP cameras and surveillance systems. The vulnerability allows an attacker to impersonate an arbitrary ThroughTek device, hijack a victim's connection, and force them to supply credentials needed to access the victim's device.

  Exploit details:

  1. Vulnerability Type: Remote Code Execution (RCE).
  2. Affected Component: ThroughTek Kalay P2P SDK
  3. Attack Vector: Network-based attack
  4. CVSS Score: 9.6

These are the exploitation examples for Botnet. And also, we can find more vulnerabilities on IoT devices and secure them from Botnet devices. But before that we have to identify what are the ports open to attackers can gain access via its connected devices. among the hackers most popular website use for that called **“Shawdan.io”** to identify open ports and find vulnerabilities. If we need, we can study about them deeply with tool named **“Nmap”.**

Figure 3 General Details



TOTAL RESULTS
420

TOP COUNTRIES

Germany	285
Brazil	55
Italy	37
France	7
Austria	6
More...	

TOP PORTS

8081	211

Product Spotlight: Keep track of what you have connected to the Internet. Check out Shodan Monitor

170.233.43.185 2024-10-10T08:10:52.089001
170.233.43.185.mhnet.com.br
MHNET TELECOM
Brazil, Erechim
vpn

```
PPTP:
  Firmware: 1
  Hostname: IP-CAMERAS-CRERAL
  Vendor: MikroTik
```

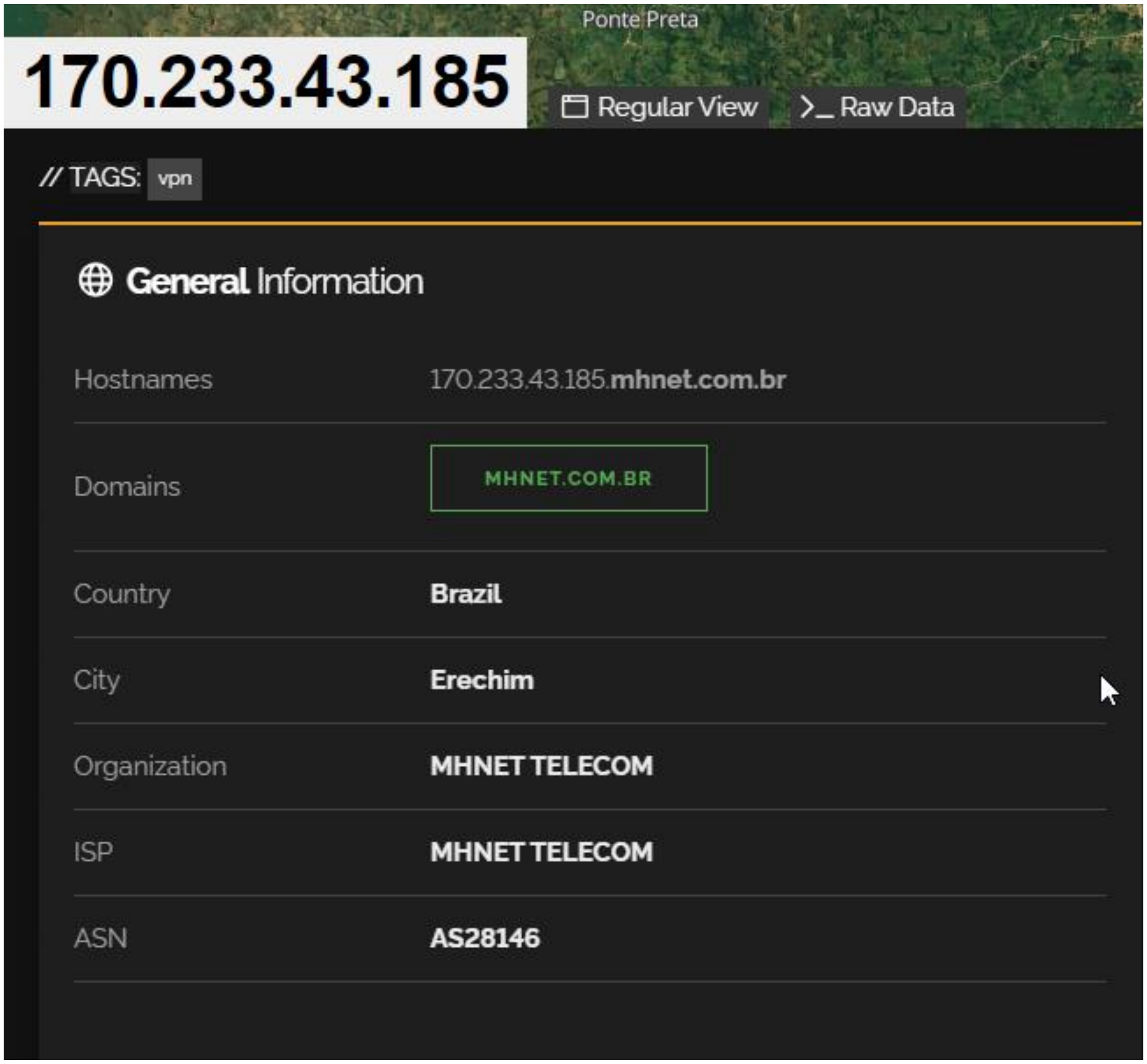
93.243.136.20 2024-10-10T07:57:39.877104
p5df38814.dip0.t-ipconnect.de
Deutsche Telekom AG
Germany, Berlin

```
HTTP/1.1 200 OK
Server: Boa/0.94.13
Date: Thu, 10 Oct 2024 07:58:17 GMT
Content-Type: text/html
Content-Length: 5205
Cache-Control: private
Connection: close

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>INSTAR IP Cameras</title>
<link rel=...
```

we can look details about **170.233.43.185 IP Camera:**

- General details:



170.233.43.185

General Information

Hostnames	170.233.43.185.mhnet.com.br
Domains	MHNET.COM.BR
Country	Brazil
City	Erechim
Organization	MHNET TELECOM
ISP	MHNET TELECOM
ASN	AS28146

Figure 4 General Details

Here all general details are available in one site called Sawdon.io and its give us all the general details what we need. Attackers can get details from it and manage their attacks behind user's devices without any catching.

But general details not enough to attack IoT devices, so attackers scanning the IoT devices network using '**Nmap'** tool and find it's open ports to enter the network by bypassing the frameworks not patched. Scanning that networks attackers can find open ports and vulnerabilities in IoT devices.

What is Nmap?

Nmap is stand for network mapper. It is the main tool attackers and pentesters are used to find vulnerabilities in network. So, it is very helpful to attackers enter the IoT devices via network.

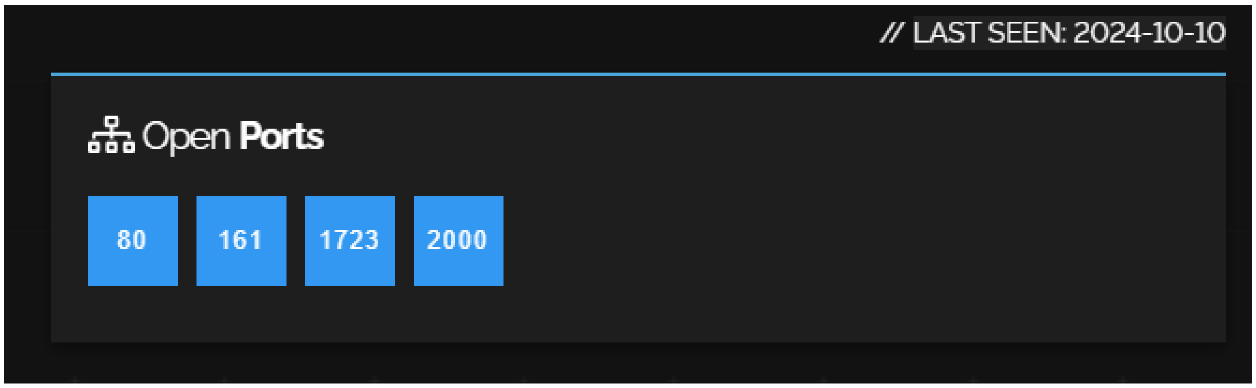
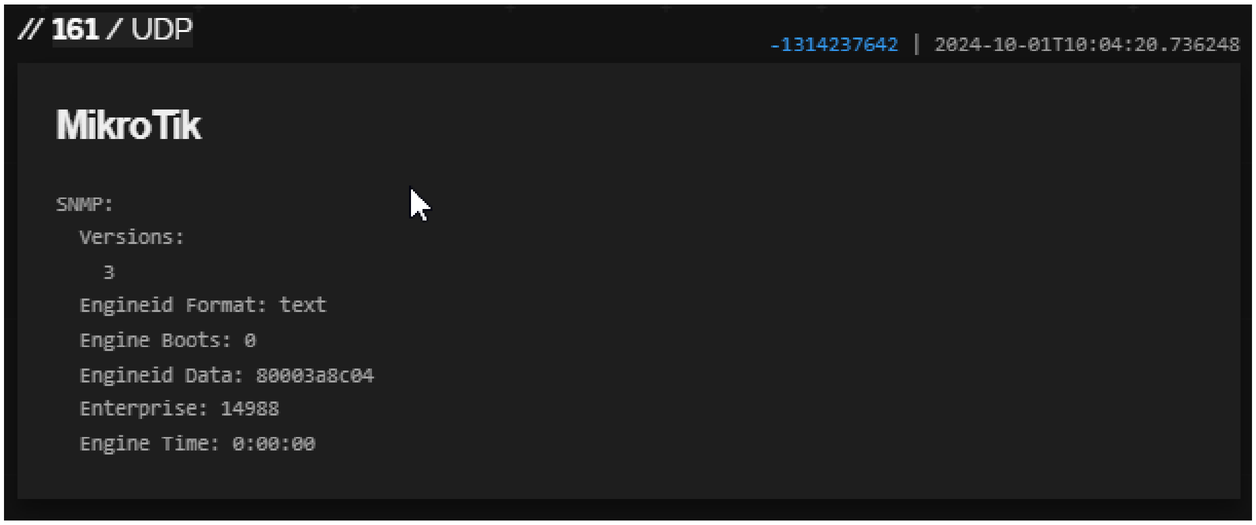


Figure 5 Open Ports found by Nmap tool

- **161 Port (SNMP: Simple Network Management Protocol)** is open on that IP Camera. Its mean we can study deeply about that port and what are the frameworks and software running on this port. After we study about that software versions, patches, what is the latest version and we can compare the version running on this port in that camera. If that IP camera didn't unpatched to latest version attackers can exploit their attacks through bugs and patches.



```
SNMP:
  Versions:
    3
  Engineid Format: text
  Engine Boots: 0
  Engineid Data: 80003a8c04
  Enterprise: 14988
  Engine Time: 0:00:00
```

Figure 6:161 Port Details

- **1723 Port (PPTP: Point-Point tunneling Protocol)**

  It is used to establish a TCP connection between two peers, which then enables the creation of a GRE (General Routing Encapsulation) tunnel for secure data transmission.

4. How secure IoT devices from Botnet attacks.

4.1 Secure Procedures.

According to the "CIA" triad we must protect credentials and data from attackers. Our sensitive data can be leak by not securing our network connection or falling down. So, we necessary increase our security in IoT devices connected to network too.

We can protect our IoT devices by writing algorithms, automating and improve them.

- Detect anomalies.
- Identify their patterns.
- Analyze the current Algorithm.
- Improve the current algorithm, if they have any bugs.
- Test run.

Detect anomalies

Detect anomalies is always humanized. But we can't find anomalies always monitoring IoT devices or network. As that reason we have to find anomalies and its solutions do automatically. In cybersecurity field, security operation center (SOC) analysts find events and monitor every step in IoT devices and networks, but in the house or small area we can't do it. We need automotive anomalies and solutions for them.

Identify their Patterns

We have to identify the risks and attacks patterns to prevent or mitigate them. In that case our network always has too up-to-date. Not only network, but also software that running on that IoT devices must be up to date. We can new algorithms from analyzing patterns of the attacks like how to attacked, what are the vulnerabilities were open to outdoor from network, how attacker bypass that are the examples.

Analyze the current Algorithm

After we analyzing attacks pattern we can get current Algorithm current running in networking system.

Why we analyze the Algorithm in networking system? Because while we connected each other to a hub, we must protect the hub before all the others. So, we need to analyze the current Algorithm to find bug and fixes them.

Improve the Current Algorithm

Earlier we analyzed the current algorithm running in our network system and in a hub. After we found the bugs, vulnerabilities, or any false in running Algorithms we have to fix them or improve them. Otherwise, we can write a new Algorithm and install it to a hub correctly.

Test run

After we identifying attacks patterns, analyzing current Algorithms and improving Algorithms we can test how it performs against the attacks again. We can running again and again find errors and fix them and run again as long as we can. Writing algorithm and their flags will be described step by step.

5.Algorithm and Description

Algorithm source code written by '**Python**' language.

1 Import and Setup

```
import re
import os
import time
import logging
from collections import defaultdict
```

- **re**: Used for matching IP addresses against malicious patterns (regex matching).
- **os**: Used to execute system commands, like blocking malicious IPs using a firewall.
- **time**: Used for time-based calculations, such as tracking how often a device is making requests.
- **logging**: Helps log activity (like suspicious behavior, blacklisting, etc.) for tracking.

- **defaultdict**: A dictionary that automatically creates an empty list for new IP addresses to track requests made by each IP.

2. Global Constans and Data structures

```
RATE_LIMIT = 100  # Maximum allowed requests per minute from an IP
REQUEST_TIME_WINDOW = 60  # Time window in seconds
BLACKLIST_TIMEOUT = 3600  # Duration to block a malicious IP (in seconds)

blacklist = {}  # Stores blacklisted IPs and the time they were added
ip_requests = defaultdict(list)  # Stores requests for each IP
```

- **RATE_LIMIT**: This defines the number of requests any single IP is allowed to make in a given period (100 requests per minute).

- **REQUEST_TIME_WINDOW**: The time window during which requests from the same IP are monitored (60 seconds).

- **BLACKLIST_TIMEOUT**: The time period after which an IP is unblocked (3600 seconds = 1 hour).

- **blacklist**: This dictionary stores the IP addresses that have been blacklisted, along with the time they were added.

- **ip_requests**: A dictionary that keeps track of all requests made by each IP address. The **defaultdict(list)** automatically creates an empty list for any new IP.

3. Detecting suspicious IP

```
def detect_suspicious_ip(ip_address):
    """Check if the IP address matches any known malicious patterns."""
    known_malicious_patterns = [
        "192.168.1.*",  # Example local network pattern
        "10.0.0.*",     # Example internal pattern
    ]
    for pattern in known_malicious_patterns:
        if re.match(pattern.replace('*', '.*'), ip_address):
            return True
    return False
```

- This function checks if the IP address belongs to a known **malicious IP pattern**. It compares the IP to patterns in the known_malicious_patterns list (e.g., IP ranges like "192.168.1.*")

- **How it works**: It replaces the * in patterns with .* (a regex wildcard) and checks if the IP matches.

4.Monitoring Network Traffic and Rate Limiting

```
def monitor_network_traffic(ip_address):
    """Monitor requests and apply rate limiting for each IP."""
    current_time = time.time()

    # Clean old requests that are outside of the time window
    ip_requests[ip_address] = [req_time for req_time in ip_requests[ip_address]
if current_time - req_time < REQUEST_TIME_WINDOW]

    # Log the new request time
    ip_requests[ip_address].append(current_time)
    # Check if the IP exceeds the rate limit
    if len(ip_requests[ip_address]) > RATE_LIMIT:
        logging.warning(f"Rate limit exceeded by IP: {ip_address}")
        add_to_blacklist(ip_address)
```

- Purpose: This function monitors the number of requests made by each IP within the last **REQUEST_TIME_WINDOW** (60 seconds).

- How it works:

  1. **Remove old requests**: It removes requests from the IP's history that are older than 60 seconds.

  2. **Log new request**: It records the current request time.

  3. **Check the rate limit**: If the number of requests exceeds RATE_LIMIT (100), the IP is flagged for suspicious behavior and added to the blacklist.

5.Blacklisting Suspicious Ips

```
def add_to_blacklist(ip_address):
    """Blacklist the IP address and block it using firewall."""
    blacklist[ip_address] = time.time()
    logging.info(f"IP {ip_address} added to blacklist.")
    block_ip(ip_address)
```

- Purpose: This function adds an IP to the blacklist and blocks it.
- How it works

  - The IP is added to the **blacklist** dictionary with the current timestamp.
  - It logs the event and calls the **block_ip()** function to block the IP.

6.Blocking Ips using Firewall

```
def block_ip(ip_address):
    """Block the IP using a firewall rule (e.g., iptables for Linux)."""
    os.system(f"iptables -A INPUT -s {ip_address} -j DROP")
    logging.info(f"Blocked traffic from {ip_address}")
```

- **Purpose**: This function blocks network traffic from the blacklisted IP using the Linux iptables firewall command.
- **How it works**: The command iptables -A INPUT -s {ip_address} -j DROP blocks all traffic from the specified IP address.

7.Unblocking Ips After Timeout

```
def unblock_ip(ip_address):
    """Unblock an IP after the blacklist timeout expires."""
    os.system(f"iptables -D INPUT -s {ip_address} -j DROP")
    logging.info(f"Unblocked IP {ip_address}")
```

- Purpose: This function unblocks the IP after it has been in the blacklist for more than **BLACKLIST_TIMEOUT** (1 hour).
- **How it works**: It uses the iptables -D command to remove the block on the IP address.

8.Cleaning the Blacklist Periodically

```
def clean_blacklist():
    """Remove IPs from the blacklist after the timeout period has passed."""
    current_time = time.time()
    for ip_address, blacklisted_time in list(blacklist.items()):
        if current_time - blacklisted_time > BLACKLIST_TIMEOUT:
            unblock_ip(ip_address)
            del blacklist[ip_address]
```

- **Purpose**: This function checks if any blacklisted IPs should be unblocked after BLACKLIST_TIMEOUT (1 hour).
- **How it works**: It checks each IP in the blacklist and unblocks it if enough time has passed.

9.Main Function: Monitoring and Protecting the Hub

```
def scan_iot_traffic(ip_address):
    """Monitor, detect, and block suspicious traffic."""
    # Check for known malicious IP patterns
    if detect_suspicious_ip(ip_address):
        logging.warning(f"Suspicious IP detected: {ip_address}")
        add_to_blacklist(ip_address)

    # Monitor the network traffic for the IP
    monitor_network_traffic(ip_address)

    # Clean up any old blacklist entries
    clean_blacklist()
```

- Purpose: This function ties everything together. It:
    1. Checks if the IP matches any known malicious patterns.
    2. Monitors the traffic and applies rate limits.
    3. Cleans up old blacklisted IPs that should no longer be blocked.

10.Simulating IoT Traffic

```
def simulate_iot_network():
    """Simulate the IoT traffic and apply defense logic."""
    sample_ips = ["192.168.1.10", "10.0.0.5", "8.8.8.8"]  # Example IPs

    for i in range(200):  # Simulate 200 network events
        ip_address = sample_ips[i % len(sample_ips)]
        scan_iot_traffic(ip_address)
        time.sleep(0.5)  # Simulating traffic at intervals
```

- **Purpose**: This function simulates network traffic by making 200 requests from three sample IPs.
- **How it works**: It calls the scan_iot_traffic() function for each simulated IP and waits 0.5 seconds between requests.

11.Auto-Update Firmware

```
def auto_update_firmware():
    """Check for firmware updates and apply them automatically."""
    logging.info("Checking for firmware updates...")
    logging.info("Firmware update applied successfully.")
```

- Purpose: This function is a placeholder for automatically checking and applying firmware updates to patch any vulnerabilities.

12.Securing Communication with Encryption

```
def secure_communication():
    """Ensure communication between IoT devices and hub is encrypted."""
    logging.info("Securing communication using SSL/TLS...")
```

- Purpose: This placeholder function ensures that communication between the IoT devices and the hub is encrypted using secure protocols like SSL/TLS.

13.Main Protection Function

```python
def protect_hub():
    """Main function that coordinates the defense strategy."""
    logging.info("Starting hub protection...")

    # Start traffic monitoring and protection
    simulate_iot_network()

    # Apply firmware updates periodically (you can adjust this time as needed)
    auto_update_firmware()

    # Secure communication (you may want to use SSL certificates here)
    secure_communication()

if __name__ == "__main__":
    protect_hub()
```

- Purpose: This is the main function that orchestrates everything**.**

  1. It starts monitoring traffic.
  2. Applies automatic firmware updates.
  3. Ensures communication is encrypted.