Text

Description automatically generated

Real-Time Defence of IoT Networks

Toyatma Fedee – B00099349

Eoin Finney - B00093420

Ciaran Meade - B00086698

Department of Informatics, School of Informatics and Engineering,

Technological University Dublin

Supervisor: Mark Lane

Submitted in part fulfilment for the degree of

B.Sc. (Hons) Digital Forensics & Cybersecurity

School of Informatics and Engineering,

Technical University of Dublin,

Blanchardstown Campus,

Dublin, Ireland

**Project Declaration Form**

Student: Toyatma Fedee , Eoin Finney, Ciaran Meade

Supervisor: Mark Lane

Course Coordinator: Mark Cummins

Module: 3rd Year Group Project

Submission Date: 14th May 2021

We hereby certify that this material, which we now submit for assessment on the programme of study leading to the award of Degree of Honours B.Sc. in Digital Forensics and Cybersecurity in Technical University of Dublin ,Blanchardstown Campus , is entirely our own work except where if not stated and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Signed :

Toyatma Fedee Ciaran Meade Eoin Finney

Table of Contents

[Abstract](#_gjdgxs) **6**

[**1 Introduction**](#_a8ws2alf02sj) **7**

[1.2 Research Question](#_3znysh7) 7

[1.3 Project objectives](#_tyjcwt) 7

[**2 Literature Review**](#_f0u8325btd62) **8**

[2.1 Historical IoT devices](#_1t3h5sf) 8

[2.2 Modern IoT](#_4d34og8) 9

[3 IoT architecture dependency](#_2s8eyo1) **10**

[3.1 IoT Communication Models](#_3rdcrjn) 10

[3.1.1 Device-to-Device communication](#_26in1rg) 10

[3.1.2 Device-to-Cloud communication](#_35nkun2) 11

[3.1.3 Device-to-Gateway communication](#_44sinio) 11

[3.1.4 Back-End Data-sharing communication](#_z337ya) 11

[3.2 Understanding IoT Attacks](#_1y810tw) 12

[3.2.1 Lack of Security Software](#_4i7ojhp) 12

[3.2.2 Interoperability Issues](#_2xcytpi) 12

[3.2.3 Unskilled Manufacturers](#_1ci93xb) 12

[3.2.4 Physical Security Risk](#_3whwml4) 12

[3.3 OWASP IoT Top 10](#_2bn6wsx) 12

[3.3.1 IoT Attack Areas](#_qsh70q) 13

[3.3.2 IoT Attacks](#_3as4poj) 14

[3.4 IoT Hacking Steps](#_1pxezwc) 15

[4 Intrusion Detection Systems](#_2p2csry) **16**

[4.1 IDS Detection Range](#_147n2zr) 16

[4.2 IDS in Networks](#_3o7alnk) 16

[4.3 Obfuscation Techniques](#_23ckvvd) 16

[4.4 Bypassing IDS](#_ihv636) 17

[4.4.1 Insertion Attacks](#_32hioqz) 17

[4.4.2 Evasion Attacks](#_41mghml) 18

[5 Intrusion Prevention Systems](#_vx1227) **19**

[5.1 IPS Process](#_3fwokq0) 19

[5.2 IPS Classification](#_1v1yuxt) 19

[5.3 IPS Detection Methods](#_4f1mdlm) 20

[5.4 Defending Against Intrusion](#_2u6wntf) 20

[5.5 Host-Based vs Network-Based solutions](#_3tbugp1) 21

[6 Firewalls](#_nmf14n) **22**

[6.1 Software vs. Hardware Firewalls](#_37m2jsg) 22

[6.2 Firewall Types](#_1mrcu09) 22

[6.2.1 Cloud Firewalls](#_2lwamvv) 24

[6.2.2 Cloud Firewall Types](#_111kx3o) 25

[6.2.2.1 FWaaS Firewalls](#_3l18frh) 25

[6.2.2.2 Next-Generation Firewalls](#_206ipza) 25

[6.2.3 Advantages of Cloud Firewall](#_4k668n3) 25

[6.3 Firewall Architecture](#_2zbgiuw) 26

[6.3.1 Bastion Host](#_1egqt2p) 26

[6.3.2 Screened Subnet](#_2dlolyb) 26

[6.3.3 Demilitarized Zone](#_sqyw64) 26

[6.3.4 Multi-homed Firewall](#_1rvwp1q) 27

[6.4 Bypassing Firewalls](#_2r0uhxc) 27

[6.4.1 By-Passing Techniques](#_1664s55) 28

[7 Honeypots](#_3q5sasy) **28**

[7.1 Honeypot Types](#_25b2l0r) 28

[7.2 Honeypot Technologies](#_kgcv8k) 29

[8 Methodology](#_34g0dwd) **29**

[8.1 Proposed method](#_1jlao46) 30

[8.2 Available Honeypots](#_43ky6rz) 30

[8.3 Integrated Tools](#_2iq8gzs) 31

[9 Analysis](#_xvir7l) **32**

[9.1 ADB Honey Analysis](#_1x0gk37) 32

[9.1.1 Examining Captured sample](#_1baon6m) 33

[9.1.2 UFO Crypto Miner](#_3mzq4wv) 36

[9.2 Suricata CVE Investigation](#_40ew0vw) 38

[9.3 Suricata Alert signatures](#_3ep43zb) 39

[9.4 Username and Password attacks](#_4du1wux) 40

[10 Testing / IDS Script](#_279ka65) **41**

[10.1](#_meukdy) 42

[10.2](#_36ei31r) 43

[10.3](#_1ljsd9k) 43

[10.4 IoT Security Hardening](#_2koq656) 43

[11. Conclusion](#_zu0gcz) **44**

[11.1 Future Work](#_3jtnz0s) 44

[11.2 Drawbacks](#_1yyy98l) 44

[12. Code](#_2y3w247) **45**

[12.1 Initial Code testing](#_sl8pbxdln96b) 45

[12.2 Code Scripts Walkthrough](#_58sfqb1uw69i) 50

[12.2.1 block-up.sh](#_c6eac765tfnf) 50

[12.2.2 sort-blocking.sh](#_j6ugpb6sy8rf) 50

[12.2.3 ip-blocker.sh](#_2ibh4ufobgra) 50

[Table of Figures](#_1d96cc0) **52**

[Table of Tables](#_3x8tuzt) **52**

[References](#_2ce457m) **53**

# Abstract

The goal of this project is to create an Intrusion Detection System integrated with an Intrusion Prevention system, creating a real-time defence network. This tool provides a platform for users to defend their network in real-time, thus tackling the issue of a lack of small, light, real-time defence. This tool is created in Python 3.x and Bash and is run on the server that is to be protected.

Keywords : IoT, Honeypots, Intrusion Detection, Intrusion Prevention, Firewalls, Security

# 

# 1 Introduction

The expression "Internet of Things" (IoT) [1] refers to any physical device that can access the web. Since the devices are attached to a network, they can interact and exchange data with one another. Cell phones, thermostats, lamps, headphones, refrigerators, televisions, fitness tracking devices, wearables and smart home systems are all instances of IoT devices that are wired to the internet and the list keeps growing regularly as more people rely on them on a daily basis. IoTs also comprise a variety of sensors, hydraulics, and measurement tools used in such as electricity grids [2], and oil rigs. As the number of IoT devices grows, so does the number of targeted attacks against them. An in-depth level of IoT attacks, Intrusion detection, intrusion prevention, honeypot deployment, IoT architecture and automation to defend against these attacks will be discussed and implemented in this project.

Therefore, this paper proposes the idea of a custom Intrusion Detection System (IDS), one of which will detect incoming scans on the network, log them, parse out the IP address and block it using iptables. The purpose of this program being that it is far lighter compared to other IP addresses, and one that can block IP addresses that come in real-time.

## 1.2 Research Question

The main questions of this project are:

1. How would you best describe IoT? Why has it gained so much buzz in recent years?
2. What are the security concerns related to IoT? Is the present security infrastructure enough to handle the data security requirements of IoT?

## 1.3 Project Objectives

The aim of this group work is to analyse some of the existing attack patterns on IoT systems. Based on this, we used a honeypot for IoTs that can handle attacks via some of the major protocols used in IoTs. Once the honeypot is set up, we deploy it and collect data. The collected data is then analysed, and patterns are then obtained from the data. A real time defence mechanism will be created to mitigate those attacks.

# 2 Literature Review

The Internet of Things (IoT) is a rapidly growing scenario surrounding modern communication. The idea of this concept is of the sheer amount of a variety of things in our everyday lives, such as radio-frequency identification (RFID) tags, Bluetooth devices and mobile phones, through unique, individual addressing schemas, have the ability to communicate with other devices.

According to Parashar, Khan and Neha [3], in 2003 there were more things connected to the internet than people living on Earth. In 2018, there were 7 billion IoT devices connected to the internet, which rose to 26.66 billion active IoT devices in 2019 [4]. According to Juniper Research, the amount of IoT devices, sensors and actuators will be over 46 billion by the end of 2021 [5], though Security Today predict that number to be closer to 31 billion [2].

The term internet-of-things (IoT) has been around for quite a while and is credited to Kevin Ashton, though the theory of connecting devices together has been circling around since at least the 1970’s, perhaps even before. Ashton’s original definition was as follows [4]:

“Today computers—and, therefore, the Internet—are almost wholly dependent on human beings for information. Nearly all of the roughly 50 petabytes (a petabyte is 1,024 terabytes) of data available on the Internet were first captured and created by human beings—by typing, pressing a record button, taking a digital picture or scanning a bar code. Conventional diagrams of the Internet ... leave out the most numerous and important routers of all - people. The problem is, people have limited time, attention and accuracy—all of which means they are not very good at capturing data about things in the real world. And that's a big deal. We're physical, and so is our environment. You can't eat bits, burn them to stay warm or put them in your gas tank. Ideas and information are important, but things matter much more. Yet today's information technology is so dependent on data originated by people that our computers know more about ideas than things. If we had computers that knew everything there was to know about things—using data they gathered without any help from us—we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best. The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so.”

According to IBM, the Internet of Things “represents an evolution in which objects are capable of interacting with other objects” [6]. They state that hospitals would be able to “monitor and regulate pacemakers” over a long distance, hotels would be able to “adjust temperature and lighting according to a guest’s preference”, and that factories would be able to “automatically address production line issues”.

## 2.1 Historical IoT devices

In 1989 at a trade show, a man called John Romkey was challenged to put a device online and demonstrate it using existing protocols. In 1990, a year or so after the internet came mainstream, John Romkey produced a toaster. The toaster was connected to a computer, where the computer controlled the power inputs in the toaster. This was to be later known as the first IoT device.

The “Trojan Room coffee pot” [7] might well be the first IoT webcam device (or the idea of it, anyways). A camera was placed looking at a coffee pot, with wires running from the camera to a frame-grabber. A server program was made that captured images 3 times a minute, and a client program was made so that everyone could see when the pot was full.

In June 2000, LG launched the first fridge to be connected to the internet. Although a commercial failure, it helped pave the way for smart devices over the next 20 or so years. It had an LCD screen that featured a pen, notepad, video and management functionality. It also provided information such as the temperature inside the fridge, the freshness of the food inside, nutritional information and recipes. It had webcam functionality and an MP3 player, and also had an ice maker [8].

## 2.2 Modern IoT

In the case of Modern IoT, an awful lot of it relies on the 802.11 standards, more specifically Wi-Fi. Wi-Fi is the most widely used network in the world, used to link devices such as desktops, laptops, tablets, smartphones, printers and smart TVs to a wireless router to connect them to the internet.

In 1999, Apple released theiBook G3 which was the first laptop with integrated Wi-Fi directed at consumers. Other laptops followed, and the first mobile phone with Wi-Fi was introduced in 2004, which was the Nokia 9500 Communicator.

By the late 2000’s, interest in IoT was increasing massively. In May 2005, a device in the shape of a rabbit known as Nabaztag was released. It was permanently connected to the internet via Wi-Fi, and would move its ears, light up in various colours, sing and talk to convey information. It had a built-in microphone and speakers, and it could function in numerous ways, such as reading RSS feeds, play internet radio/podcasts/mp3 files, announce the time and respond to spoken commands [9]. In 2008, the Internet Protocol for Smart Objects (IPSO) alliance was created primarily to promote the use of IP networked devices for applications such as energy, health-care, industrial and consumers.

Radio frequency Identification (RFID) is wirelessly using electromagnetic fields to transfer data. It is used to automatically identify and track tags attached to objects. RFID tags store electronic information. There are three types of RFID devices: 1) those powered by electromagnetic induction from magnetic fields from the reader; 2) those that act as a passive transponder; and 3) those that have their own power source and can work at a distance from the reader.

# 3 IoT Architecture Dependency

An architecture is needed as the majority of IoT technologies are wireless and makes the framework more complex and difficult to control. This classifies physical components of a network by their configuration, functionality, processes, and data type upon execution. The architecture of IoT depends on five layers [9] which are illustrated in Figure 1.

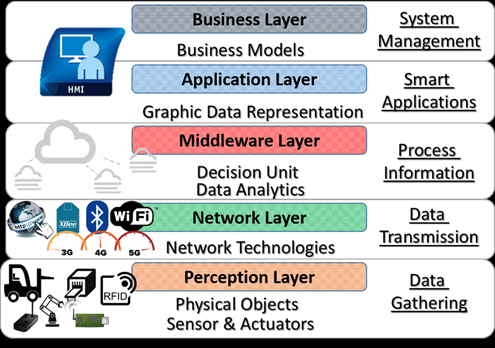


Figure 1. The 5 Layers of IoT Architecture

* Perception Layer : It is the first layer and covers IoT capable devices to gather and pass data to the next layer so that some actions can be carried out.
* Network Layer : It is responsible for secure and confidential protocol transmission and messaging between the perception layer to the middleware layer
* Middleware Layer : It consists of advanced features capabilities for device and data management.
* Application Layer : The layer responsible for processing data from the middleware to users. This is the user interface to control, manage and command IoT devices through graphical data representation.
* Business Layer : It is a system management layer for consumers that involves data analysis , graphical representation and among other things.

## 3.1 IoT Communication Models

There are numerous pathways in which IoT devices can communicate [10]. The following are some of these communication patterns as presented by Internet Architecture Board(IAB) in their paper , Smart Object Architectural Considerations for referencing purposes:

### 3.1.1 Device-to-Device Communication

It is a simple mechanism in which two devices communicate wirelessly together without interacting with other devices. Bluetooth, Wi-Fi, Z-Wave, RFID and NFC are wireless communication means for this model.

Logo, company name

Description automatically generated

Figure 2. Device to Device Model

### 3.1.2 Device-to-Cloud Communication

IoT devices interact with one another using a cloud service like an application server to exchange data and monitor communication traffic. For instance, in manufacturing environments, many sensors send data to a server. The data is processed by application servers, which then take automatic actions based on the reports.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 3. Device to Cloud Model

### 3.1.3 Device-to-Gateway Communication

A local gateway device runs an application software that acts as an interface between the device and the cloud service, offering security and other functionality such as data or protocol translation.

Timeline

Description automatically generated

Figure 4. Device to Gateway Model

### 3.1.4 Back-End Data-Sharing Communication

This model expands on the Device-to-Cloud model by allowing multiple entities to connect and manage IoT devices and sensors in a scalable context and IoT devices frequently interact with an application server in this framework.

Diagram

Description automatically generated with medium confidence

Figure 5. Back-end Data sharing Model

## 3.2 Understanding IoT Attacks

IoT targeted attacks are increasing as more devices are coming to the market, on top of the usual attacks, other key challenges [11] are present in IoT ecosystems.

### 3.2.1 Lack of Security Software

IoT devices do not come up with a firewall or virus scanner as computers would have. Difficulties to update firmware and operating system can also cause major security risk.

### 3.2.2 Interoperability Issues

Multiple devices with the same security mechanisms produce vulnerable interfaces. if an attack works with one device it will work with thousands more.

### 3.2.3 Unskilled Manufacturers

The companies are on average from a specific sector and lack the IT security experience of server/computer manufacturers and poor merchant backing for devices.

### 3.2.4 Physical Security Risk

Most device owners access their devices remotely. Owners also do not realise their devices have been infected until it is too late. Once an intruder has control of a device, it can operate for hours until the owner physically shuts it down.

## 3.3 OWASP IoT Top 10

According to OWASP [12] latest top 10 vulnerabilities include the following areas:

* Weak, Guessable, or Hard Coded Password: Backdoors in firmware or client software that permit unauthorised access to deployed systems, including use of passwords that are easily brute forced, freely available, or unchangeable.
* Insecure Network Services: Unnecessary or unprotected network services running within the device especially those subjected to the internet, that compromise data confidentiality, integrity/authenticity, or availability, or allow unauthorised remote control.
* Insecure Ecosystem Interfaces: Insecure web, backend API, cloud, or mobile interfaces in the ecosystem that enable the device or its related components to be compromised. A lack of authentication/authorisation, missing or poor encryption, and a shortage of input and output filtering are all common problems.
* Lack of Secure Update Mechanism: Failure to update the system in a safe manner. This involves a lack of firmware verification on devices, insecure distribution (data is sent unencrypted), anti-rollback protocols, and alerts of security changes caused by updates.
* Insecure or Outdated Components usage: Outdated or vulnerable software components/libraries that may enable the system to be exploited. It also involves inaccurate operating system platform modifications and the use of third-party software or hardware parts obtained from a corrupted distribution network.
* Insufficient Privacy Protection: Sensitive user data which is stored on the device or system and is used in an unsafe, incorrect, or unauthorized manner.
* Insecure Data Transfer and Storage: No encryption or access control for confidential data in the system, including at rest, in transit, and during processing.
* Lack of Device Management: Asset management, patch management, security backing, safe disposal, systems monitoring, and mitigation are all missing on devices that have been deployed in production.
* Insecure Default Settings: Devices or systems that are sold with vulnerable default settings or which lack the capacity to protect the device by blocking operators from changing configuration.
* Lack of Physical Hardening: Physical hardening steps are missing, leading future attacks to access confidential data which can be used in a future remote attack or hijacking the system.

### 3.3.1 IoT Attack Areas

The following are the most common attack surface [13] areas for IoT:

* Access Control
* Confidentiality and Integrity issues
* Cloud Computing Attacks
* Device Memory containing credentials
* Firmware Attacks
* Firmware Extraction
* Hardware(sensors)
* Insecure APIs
* Malicious Updates
* Mobile Application Threats(3rd Party)
* Network Services Attacks
* Privilege’s Escalation
* Removal of Storage Media
* Unencrypted local Data Storage
* Web Attack
* Vendor Backend APIs

### 3.3.2 IoT Attacks

IoT devices can be targeted [14] in different ways. Some most common techniques are classified below.

* *DDoS attacks:* All the services associated [15] with IoT involving devices, gateways, and application servers, can be attacked using this technique.
* *Code Hopping attacks:* The attacker intercepts the code [16], sequence, or signal sent by the transmitters while simultaneously blocking the receivers. The code could be potentially used to obtain physical access in the future. For instance, a car's opening signal that could be captured and replayed later.
* *Blue Borne attacks:* It is the use of various techniques to exploit Bluetooth vulnerabilities [17] in devices to gain illegal access.
* *Eavesdropping:* A hacker captures network traffic [14] to intercept sensitive data through a compromised link between an IoT device and a server in this type of attack. Listening to digital/analog voice communication or intercepting sniffed data are some examples for this attack.
* *Exploit kits:* They are automated programs [18] that attackers use to take advantage of known flaws in systems or applications and launch silent attacks when victims are browsing the web, with the aim of downloading and running malware.
* *Jamming ACK:* Jamming [19] signals to prevent devices communication is a specialised type of Denial of Service.
* *Backdoor:* Placing backdoor [20] on the victim's computer or IoT devices like Wi-Fi security camera to gain persistent access to the network is one common technique of attack.
* *Sybil attack:* The Sybil attack [21] is a method of corrupting a legitimate system in peer-to-peer networks by forging identities.
* *Firmware Hijacking :* Most hardware manufacturers do not sign embedded firmware [14] cryptographically. IoT firmware should be updated regularly, and the source must be verified as hackers can exploit the system and install malicious software.
* *MitM attacks:* A man-in-the-middle [14] attack happens when an intruder intercepts data from two different devices. This form of attack deceives the recipient into believing they are receiving a genuine message by secretly intercepting messages between two parties.
* *Ransomware attack:* Ransomware [14] is a form of malware that encrypts files and prevents access to them. Naturally, this form of attack can cause significant disruption to daily operations and obtaining the encryption key can be costly. The hackers will provide you with a decryption key once the victim pays them. This can prove disastrous for example if a power grid or oil rig is targeted.
* Replay attacks: A replay attack [22] is a low-level attack that simply replicates a signal that is used to control some system. IR, RFID, NFC are some examples where this attack can be maliciously used.
* Side channel attacks : These attacks [23] leverage the power of communication patterns generated by the execution of computing tasks in embedded systems and other similar systems.

## 3.4 IoT Hacking Steps

The hacking approach used on IoT platforms is identical to that used on other platforms. They are classified below.

* Data-gathering : Shodan [24], Censys [25], and Thingful [26] are excellent search engines to gather IoT device info such as IP addresses, protocols being used, open ports, device types, vendor details, and so on.
* Vulnerability scanning: . Nmap [27] and other related programs are extremely useful to discover security flaws, weak passwords, software and firmware issues, default settings, and other conflicts by scanning the network and devices.
* Launching the attack: Popular hacking tools like HackRF One [28] and RFCrack [29] exploit weaknesses with various attacks such as DDoS, rolling code, jamming, and many more.
* Gain Access: By gaining access to the IoT, hackers can do privilege escalation, install backdoor and control few more things in this stage.
* Attack Persistence: Logging out by being undetected, clearing logs, and covering tracks occurs at this stage.

Attacks not only are carried out manually; they are also fully automated by computers using specially designed software. As a result, automation and security tools must be capable of dealing with large numbers of attacks and the resulting traffic. Firewalls, intrusion detection systems, intrusion prevention systems, and honeypots are valuable tools in maintaining secure and trusted data exchange between different devices, networks, or systems. A detailed explanation about those tools will be given in the next section.

# 4 Intrusion Detection Systems

An intrusion detection system (IDS) [30] is a device or software application which tracks malicious behaviour or data breaches on a network. A security information and event management(SIEM) framework is commonly used to monitor or collect any malicious activity or breach. Some IDSs have the flexibility to adjust to intrusions as soon as they are detected and those are classified as an Intrusion prevention system(IPS).

## 4.1 IDS Detection Range

IDS come in a variety of ranges [31], from antivirus applications to monitoring systems that track complete network's traffic. IDS are generally classified as Network IDS and Host-based IDS, but there is also another subclass based on detection techniques namely Signature-based [32] and Anomaly-based [33].

* Network intrusion detection systems (NIDS): Incoming network traffic is monitored by this process.
* Host-based intrusion detection systems (HIDS): A process that keeps track of key operating file systems.
* Signature-based: This checks for specific patterns(signatures) in network traffic, such as byte sequences or known malicious set definition utilised by malware, to detect potential threats, however new attacks for which no signature exists are impossible to detect.
* Anomaly-based: This detection approach uses machine learning to build a classifier for reliable behaviour, which is then compared to new behaviour. While this method allows detection of newly discovered attacks, it is open to false positives, where certain previously unknown legitimate behaviour is mistakenly classified as malicious.

## 4.2 IDS in Networks

An IDS, when deployed at strategic points [34] inside a network to monitor traffic, can detect and compare passing traffic on subnets to a database of known attacks. Warnings may be sent to the administrator after an attack has been detected or suspicious behaviour has been detected.

## 4.3 Obfuscation Techniques

Understanding what mechanisms [30] hackers employ to hack a protected network can support to understand how IDS systems can be manipulated to bypass enforceable attacks. Below is a list of common tactics used:

* Fragmentation: Through sending fragmented packets, attackers will remain undetected by signature detection systems.
* Avoiding Defaults: A protocol's port does not always include data about the protocol which is being transmitted. The IDS may not be capable of detecting the intrusion of a trojan if it has been modified to use a different port by a hacker.
* Coordinated, low-bandwidth attacks: Using several hackers to perform a scan, or even giving separate ports or hosts to different hackers. This makes it hard for the IDS to match the captured packets and determine whether a network scan is going on.
* Address spoofing/proxying: By bouncing an attack across unsecured or incorrectly configured proxy servers, attackers may mask the source of the attack. It is very difficult to detect if the source is fake and bounced by a server.
* Pattern changing obfuscation : Pattern matching is used by IDS to detect attacks. Detection can be minimized by creating small tweaks to the attack architecture.

## 4.4 Bypassing IDS

Most intrusion detection systems are based on signatures. A hacker could potentially generate a custom packet payload that does not fit any of the signatures in the IDS's predefined database. This allows the hacker to get around the IDS and potentially compromise the remote device without raising any alarms [35]. Such attacks are explained below.

### 4.4.1 Insertion Attacks

IDSs will accept packets that End-systems denies. This leads IDSs to believe that end-systems have accepted and processed the packet when it has not. Attackers will take advantage of this condition by sending packets to end-systems that will be ignored but that IDSs will believe are legitimate.

Attackers will “insert” data into IDSs with packets containing bad checksum or TTL values and out of order, and those bad packets will be ignored by the system but accepted by the IDS. Insertion attacks vulnerabilities can be used by hackers to get around signature analysis employed by IDS. For example, attackers may send this one-character packet, the IDS and End-system will reconstruct two different strings. An attacker will send out of order packets with bad checksum and TTL value, Due to the bad checksum, IDS and End-System reassemble different streams. IDs can accept the stream while End-system may reject it. To reduce insertion attacks, IDS should be as firm as the End system while handling packets, however while this design technique is applied, evasion attacks can occur.

Diagram

Description automatically generated

*Figure 6. Insertion Attack*

### 4.4.2 Evasion Attacks

End-systems will accept packets that IDS denies. IDSs that incorrectly deny such packets ignore their content. This condition can be manipulated, by pushing critical data in the packets which IDSs will process strictly. These packets are therefore evading the detection of intrusion detection systems.

Pattern matching is hindered by evasion attacks similar to insertion attacks. Again, the attacker leads the IDS to see a separate source of data than the end-system; however, this time, the end-system sees more than the IDS, and the data that the IDS ignores is vital to detecting an attack. The example below illustrates an attacker sending out of order packets with overlapping fragments and TCP segments. IDS will drop overlapping segments, dropping the “A”, IDS sees “ATTCK”, but the End-System may accept the overlapping segments, thus seeing “ATTACK”.

Diagram

Description automatically generated

*Figure 7. Evasion Techniques*

# 5 Intrusion Prevention Systems

An intrusion prevention system [36] (IPS) is a network security device that evaluates network traffic by monitoring and responding to potential threats automatically based on a set of rules set by a network administrator. Intrusion detection devices are installed in-line and are capable of actively preventing or blocking intrusions. IPS core features will be to detect suspicious activity, carry relevant data logging, blocking suspicious activity, and finally reporting those activities. Firewalls, anti-virus software, and anti-spoofing software are all examples of intrusion prevention systems. An IPS can also be used by organisations for other reasons, like detecting issues with security policies, tracking known attacks, and discouraging people from breaking security protocols. Since cyber-attacks are indeed likely to become more complex, it is essential that mitigation techniques catch pace. When a conventional security system fails, the intrusion prevention system serves as an efficient system security solution.

## 5.1 IPS Process

An intrusion prevention system [36] detects malicious activity and known attack patterns by actively scanning forwarded network traffic. The IPS engine analyses network traffic and compares it to its internal signature database for known attack patterns. If a packet is determined to be malicious, the IPS can drop it and then block all future traffic from the attacker's IP address or port. Legitimate traffic will continue to flow without causing any potential downtime. Usually, an IPS will log data concerning detection instances, alert security admins and compile reports. Intrusion prevention systems may also undertake other complex examination and evaluation, such as monitoring irregular traffic flows or packets and responding accordingly. Detection methods can include:

* Address comparison
* HTTP string/sub-string comparison
* Generic pattern comparison
* Packet anomaly analysis
* TCP connection analysis
* TCP/UDP port comparison
* Traffic anomaly analysis

## 5.2 IPS Classification

There are four main groups of intrusion prevention [36] systems:

Network-based intrusion prevention system (NIPS): Evaluates protocol activity across the entire network in search of any potentially unsafe traffic.

Wireless intrusion prevention system (WIPS): Evaluates network protocol activity through the entire wireless network for any unsafe traffic.

Host-based intrusion prevention system (HIPS): A separate application software that monitors and analyses events taking place on a given host for malicious activity.

Network behaviour analysis (NBA): Evaluates network traffic to detect threats that trigger unusual traffic patterns. The far more prevalent threats are distributed denial of service (DDoS) attacks, malware, and violations. To detect attacks, pattern matching is used, but detection can be minimized by making small tweaks to the attack architecture.

## 

## 5.3 IPS Detection Methods

The most common monitoring [36] systems used by intrusion prevention systems are signature-based, statistical anomaly-based, and stateful protocol analysis.

* Signature-based detection: IDS analyses network packets and compares them to known attack patterns called "signatures."
* Statistical anomaly-based detection: An IDS that is focused on irregularities will analyse network traffic and compare these to anticipated patterns. The benchmark would determine what is regarded “normal” for that network, like the types of packets that pass through it and the protocols that are used. If the benchmarks are not thoughtfully configured, it may raise a false positive warning for genuine network usage.
* Stateful protocol analysis detection: This approach compares detected behaviours to old behaviour patterns of normal activity to identify protocol anomalies.

## 5.4 Defending Against Intrusion

Many IPS may systematically prevent a known attack from progressing. They implement a variety of strategic responses, such as:

* Adjusting the security setting, such as updating a firewall to provide added security against newly discovered exploits.
* Resetting a connection.
* Blocking traffic from malicious IP addresses.
* Adapting the context of the attack, such as swapping potentially malicious sections of an email, like fake links, with alerts about deleted data.
* Automatically notify system administrators of potential security breaches
* Dropping malicious packets when detected.

Now that we investigated both Intrusion detection system and intrusion detection system, a more concise features definition is illustrated below.

Table

Description automatically generated

Figure 8 Feature comparison between IDS and IPS

## 5.5 Host-Based vs Network-Based solutions

IDS/IPS can be divided into two groups [37] based on their network location. The first is a host-based IDS, and the second is a network-based IDS. There is no simple method to ascertain which type of IDS would be well on the network and all types of IDS have advantages and disadvantages. In certain scenarios, deploying all types of IDS can be useful, for example, network-based IDS can provide protection to the entire network, while host-based IDS can be deployed for sensitive data hosts. Below a comparison chart has been created to illustrate their differences based on the features required for the network.

Table

Description automatically generated

Figure 9 Host-based vs Network-based IPS/IDS feature comparison

There are many monitoring factors that contribute to a good defence system. Such factors are:

* *File system monitoring:* It checks for variations in system files to see if they have been tampered with or modified. Often, hash functions are used, which hold the previous version and compare it to the current version.
* *Connection Analysis:* Attempts to investigate all network connections and identify between authorised and authorized traffic.
* *Kernel Level Detection:* The OS kernel begins to detect changes to the system binaries and, if anomalies are found, the intrusion attempt is issued.
* *Log Files Analysis:* Log files are reviewed, and vital alerts are sent to administrators. Several modern methods analyse behaviour patterns and perform further comparisons with real events.

# 6 Firewalls

Firewalls are security mechanisms [38] that are used to prevent or limit unauthorised access to private networks connected to the Internet. Firewall policies separate a trusted network from an untrusted network by specifying the traffic permitted on the network; all other traffic attempting to connect is blocked. All data entering or leaving the network must pass through a network firewall, which inspects each incoming message and rejects those that do not meet the specified security benchmarks. A firewall, when configured, enables users to access any services they need while preventing unauthorised access, hackers, viruses, worms, or other malicious programs from gaining access to the protected network.

## 6.1 Software vs. Hardware Firewalls

Firewalls may be hardware or software based. Firewalls will restrict access to protected systems or networks by logging all traffic entering or exiting a network and subsequently manage remote access to a private network through access control certificates and logins. The most important factor in determining which solution to use is performance [39].

* Hardware firewalls: These firewalls are sold as independent solutions for enterprise use or as part of a router or other networking system. They are a vital piece of every intrusion detection system or network setup. Most such hardware firewalls include at least four network ports for connectivity to different devices. Expansive networking firewall solution is also available for wider networks.
* Software firewalls: These are either enabled on a computer or installed by the manufacturer of an operating system or a network interface. They can be configured and provide a limited degree of control over functionality and security features. A software firewall can protect a device from common control and access attempts, but it will not be able to defend it from more complex network attacks.

## 6.2 Firewall Types

Firewalls are instances of endpoint security, although it is the first line of defence in protecting private data, however, it cannot be the sole defence mechanism. A generic firewall software or hardware would filter all data passing through the network. This process can be configured based on the user requirements and the firewall's capability. There are many different firewall filtration instances [40] in place that keep malicious content out of the network and can be used simultaneously such as :

* Application-layer: These firewalls filter traffic at 3, 4, 5, 7 OSI layer. This can be a server plug-in, a hardware implementation or software filter. It defines HTTP connection rules and channels security mechanisms on top of predefined applications like FTP servers. These rules are customised for every application to improve detection and prevention of network attacks.
* Circuit-level: Once a UDP or TCP link is created, these firewall models ensure that connections and sessions are safe and operate at the session layer of the OSI layer without further packet filtration.
* Packet Filtering: It connects to a router or switch and operates as an inline security barrier. It tracks network traffic by filtering incoming packets based on the information they contain, then accepts or denies it as defined by rules set by the user, but it is prone to IP spoofing.
* Proxy Server: it inspects all messages entering and exiting a network, then mask the actual network addresses from external scans
* Next-Generation (NGFW): This operates by analysing network traffic; filtration is decided by the applications or traffic types that are allocated to the ports. Instead of relying exclusively on header content like traditional firewalls does, NGFW examines the packet's actual payload.
* Stateful Firewalls: This firewall monitors the TCP 3-way handshake to keep track of the connection status from start to finish, creating a state table. Based on this table, a dynamic firewall rule gets created to filter anticipated traffic.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Firewall Types** | **Advantages** | **Disadvantages** | **Protection level** | **Intended usage** |
| **Application-Layer** | Authenticate individuals  Difficult to spoof and make DoS attacks  Monitor and filter application data  Detailed logging solution | Throughput limitation  full up a lot of disk space by writing many logs. | Filtering at 3, 4, 5, 7 OSI layer | Enable the highest level of filtering for protocol. |
| **Packet-Filtering** | Fast and effective for filtering headers  Do not use up a lot of resources  Low cost | No payload checks.  Vulnerable to IP spoofing.  Cannot filter application layer protocols.  No user authentication | Not very secure as they do not check the packet payload. | A cost-efficient solution to protect devices within an internal network.  A means of isolating traffic internally between different departments. |
| **Circuit Level** | Resource and cost-efficient  Provide data hiding and protect against address exposure.  Check TCP handshakes. | No content filtering.  No application layer security.  Require software modifications. | Moderate protection level (higher than packet filtering, but not totally efficient as there is no content filtering) | They should not be used as a stand-alone solution.  They are often used with application-layer gateways. |
| **Proxy Server** | Protect systems by preventing contact with other networks.  Ensure user anonymity.  Unlock geolocational restrictions. | May reduce performance.  Need additional configuration to ensure overall encryption.  Not compatible with all network protocols. | Offer good network protection if configured properly. | Used for web applications to secure the server from malicious users.  Utilised by users to ensure network anonymity and for bypassing online restrictions. |
| **Next-Generation** | Integrates deep inspection, antivirus, spam filtering, and application control.  Automatic upgrades.  Network traffic monitoring from Layer 2 to Layer 7 | Costly compared to other solutions.  May require additional configuration to integrate with existing security management. | Highly secure. | Suitable for businesses that require PCI or HIPAA compliance.  For businesses that want a package deal security device. |
| **Stateful Firewalls** | Keep track of the entire session.  Inspect headers and packet payloads.  Offer more control.  Operate with fewer open ports. | Not as cost-effective as they require more resources.  No authentication support.  Vulnerable to DDoS attacks.  May slow down performance due to high resource needs. | Provide more advanced security as it inspects entire data packets while blocking firewalls that exploit protocol vulnerabilities.  Not efficient when it comes to exploiting stateless protocols. | Considered the standard network protection for cases that need a balance between packet filtering and application proxy |

Table 1. Comparison between Firewall instances

### 6.2.1 Cloud Firewalls

As the name suggests, this firewall is software-based and hosted in the cloud, which is designed to prevent or minimize unauthorised access to private networks. It is a safe virtual firewall encompassing cloud networks, applications, and infrastructure.

### 6.2.2 Cloud Firewall Types

There are two types [41] of cloud firewalls, they are defined mainly by what the user needs to secure in a specific environment. Both are software based that tracks all incoming and outgoing data packets and filters them against access policies to block and log malicious behaviour.

### 6.2.2.1 FWaaS Firewalls

They are like regular local hardware or software firewalls that are configured to protect an organization's network and its users. The only difference is that it is hosted in the cloud and implemented remotely. Firewall-as-a-service [42](FWaaS), Software-as-a-service(SaaS) or Security-as-a-service(SECaaS) are common examples of this type of firewall.

### 6.2.2.2 Next-Generation Firewalls

The rising complexity of virtualized networks and increasingly sophisticated security risks, NGFW [43] is needed. In a platform-as-a-service [44] (PaaS) or infrastructure-as-a-service (IaaS) model, it secures the organisation's servers. The goal of next-generation firewalls is to include more layers of the OSI model, improving filtering of network traffic that is dependent on the packet contents.

### 6.2.3 Advantages of Cloud Firewall

Cloud Firewalls [45] comes with few advantages, listed below.

* Scalability: Since implementation is much easier, businesses can scale up or down their security solution without the hassles of on-site deployment, servicing, and upgrades. Cloud firewalls will dynamically adapt to ensure consistency as bandwidth grows. Distributed denial-of-service (DDoS) attacks, for example, can be mitigated disregarding bandwidth limits.
* Availability: By maintaining hardware, cloud firewall vendors pay for the built-in cost of high availability. In the case of a site disruption, this involves ensuring redundant electricity, HVAC, and network resources, as well as automating backup techniques; appropriate updates may be introduced instantly as no large system downloads or patches are needed.
* Extensibility: Cloud firewalls can be accessed and installed from any location where an enterprise has a secure network connectivity. The extensibility of an on-site system is restricted by the enterprise’s available resources when aiming for a firewall solution.
* Migration Security: A cloud firewall can filter traffic from a multitude of sources, including the internet, virtual networks, users, and even a virtual data centre. It will ensure the reliability of connections between physical data centres and the cloud, which is particularly useful for companies seeking to move their solutions from on-site to cloud-based networks.
* Secure Access Parity: Cloud firewalls provide the same degree of protection as traditional on-site firewalls. Advanced access policies, connection management, and filtering between clients and the cloud are all part of this. This also applies to content that has been encrypted.
* Identity Protection: Cloud firewalls can work with access control providers to offer users great control over filtering capabilities.
* Performance Management: Cloud firewalls offer tools to manage performance, visibility, access, configuration, and logging, all of which are typically found in physical solutions.

## 6.3 Firewall Architecture

Firewalls may be implemented at various levels in the network, depending on the company's security policy. The following are the most popular implementation scenarios [46].

### 6.3.1 Bastion Host

A bastion host [47] is a system that sits in the middle of a public and private network. It is designed to be the crossing point from which all traffic transits. It is given specific functions and responsibility to execute.

Chart

Description automatically generated with medium confidence

Figure 10 Bastion Architecture

### 6.3.2 Screened Subnet

A firewall with three interfaces can be used to build a screened subnet. The public, private, and demilitarized zones(DMZ) are all connected via these interfaces. The aim of this architecture is to separate the various zones. As networks grow, the need for a zone to secure internal assets becomes more pressing. As a result, many deployments also have a Demilitarized Zone (DMZ) to distinguish internal assets from assets that link to the Internet.

### 6.3.3 Demilitarized Zone

A DMZ network [48] is a subnetwork that maintains a company’s open external services. It aims to provide an extra layer of protection to a company's local area network from the internet. When properly implemented, a DMZ network provides additional security by detecting and preventing security breaches before they enter the internal network, which contains valuable resources such as email, web servers, and DNS servers. They are mounted in the monitored subnetwork to help protect the rest of the network if they become compromised due to the increased risk of attack. Since data passing through the DMZ is not as protected as data passing through the internal network, hosts in the DMZ have securely managed access permissions to other services within the internal network. Furthermore, communications between hosts in the DMZ and the external network are limited to help extend the secure border zone. The firewall filters and handles all traffic exchanged between the DMZ and the internal network, allowing hosts in the secure network to communicate with both the internal and external networks. Typically, an extra firewall would oversee securing the DMZ from being exposed to the rest of the network by implementing stateful inspection, Packet filtering, URL filtering and Virtual Routing Forwarding(VRF).

Diagram

Description automatically generated

Figure 11. DMZ Architecture

### 6.3.4 Multi-homed Firewall

A multi-homed firewall [49] is one with several network interfaces, each of which is attached to a logically and physically independent network portion. This improves network performance and reliability.

A picture containing chart

Description automatically generated

Figure 12 Multi-homed Architecture

## 6.4 Bypassing Firewalls

Firewall identification is the first step to bypass it. Different techniques can be applied to fingerprint a firewall to gather sensitive data such as services running, open ports and version data. Tools like Nmap and hping3 are widely used in this reconnaissance process. Techniques listed below are some most commonly used by pen testers or hackers to gather data.

* *Port Scanning:* Distinct packets may be sent to the host to analyse the responses and deduce environmental information, particularly open ports.
* *Firewalking:* It is a technique that uses ICMP packets to locate a firewall and trace a network by probing the ICMP echo request with TTL values that surpass one by one. It aids in determining the number of hoops.
* *Banner Grabbing:* The various devices in a network reveal different banners, manufacturer details that can be discovered, device and firmware can also be acquired. It is also called service fingerprinting.
* *IP Address Spoof:* Attackers may use spoofed IP addresses to send packets that imitate end user systems and obtain illegal access.
* *Source Routing*: This method sends a spoofed packet through a route that replicates the legitimate traffic path.

### 6.4.1 By-Passing Techniques

Different techniques exist to circumvent firewalls.

* *Using proxies:* It is typical to use proxies to bypass blocked websites by covering the actual IP address and accessing the website via the proxy IP address.
* *ICMP tunnelling method:* ICMP tunnelling is a process of inserting arbitrary data into the payload of an echo packet before sending it to the destination host. Since firewalls do not check the payload area of ICMP packets, a TCP link is tunnelled over ping requests and replays in this case.
* *HTTP tunnelling method:* It is using a real HTTP server, incorporating data in HTTP traffic, and using services such as FTP.
* *SSH tunnelling method:* OpenSSH can be used by an attacker to encrypt traffic to evade detection by security systems.
* *External systems*: This approach involves hijacking a legitimate user's session with privileges to connect to external networks then use that session to get around the firewall..

# 7 Honeypots

Honeypots [50] are deceptive systems or servers that are placed alongside a network's development systems. By diverting the attacker’s attention away from their actual objective, additional defence detection strategies can be implemented to the network. Honeypot varies based on the business requirement and they can be a valuable point of defence in detecting attacks fast. This section delves further into what honeypots are, how they are used, and the advantages of using them. Honeypots offer plenty of security benefits to organisations that choose to implement them. They slow down the attackers and disrupt the kill chain. Typically, they are low maintenance, easy to use and improve incident response procedure evaluation.

### 7.1 Honeypot Types

There are various levels inside development and analysis honeypots [51], based on the degree of sophistication any business employs:

* Pure honeypot: This is a complete, production-like machine that operates on several servers. It is full of sensors and includes "confidential" data and user data. The data they offer are valuable, even though they can be complicated and difficult to maintain.
* High-interaction honeypot: This is similar to a pure honeypot as it runs numerous services but is less complex and holds less data. While high-interaction honeypots are not intended to replicate a full-scale production system, they do run (or seem to run) all the operations that a production system might, including a proper operating system. The deploying entity will observe attacker behaviour and tactics using this form of honeypot. High-interaction honeypots need a lot of resources and are difficult to maintain, but the results can be worth it.
* Mid-interaction honeypot: These do not use their own operating system, but they do replicate features of the application layer. They try to obstruct or delay attackers so that entities have more time to find out how to react accordingly to such an attack.
* Low-interaction honeypot: This is the most used honeypot in a production setting. Low-interaction honeypots run a few services and are mostly used as an early detection mechanism. Often security teams install several honeypots around various segments of the network because they are simple to set up and manage.

### 7.2 Honeypot Technologies

A few honeypot technologies [52] in use include the following:

* Malware honeypots: To detect malware, these employ well-known replication and attack vectors. Honeypots (such as Ghost) have been designed to look like a USB storage device. If a machine becomes infected with malware that spreads via USB, the honeypot can deceive the malware into infecting the emulated system.
* Iot Honeypots: This is typically used to capture attacks in IoT devices
* Spam honeypots: Open mail relays and open proxies are simulated with these. Spammers would first send themselves an email to test the open mail relay. If they are successful, they can send out a large amount of spam. This form of honeypot can detect and identify the test, as well as successfully blocking the vast amount of spam that follows.
* Database honeypot: Since SQL injections can sometimes go undetected by firewalls, some organisations may use a database firewall, to generate decoy databases and provide honeypot service.
* Client honeypots: Client honeypots constantly track malicious servers that target clients and monitor any odd or unexpected changes. Virtualization technology is used in each of these systems, and an isolation system is implemented to keep the host secure.
* Honeynets: A honeynet is a network made up of many honeypots. Honeynets are configured to detect an attacker's tactics and intentions while collecting both inbound and outbound traffic.

# 8 Methodology

Honeypots will be deployed across the world wide web. Data will be monitored, collected, and analysed . Based on these analyses, we will implement active intrusion detection/prevention to mitigate attacks on IoT devices. Firewall rules may also be implemented by blacklisting or whitelisting malicious IPS. Scripts to automate this process will be included further below in this chapter.

## 8.1 Proposed method

We decided to use T-Pot [53] Honeypot which consists of a dockerised version of several honeypots and analysis tools. The framework will be hosted online using services such as Digital Ocean or Google Cloud. Once data is collected for a few weeks, we will write scripts to mitigate these attacks based on malicious abusive IPs and attacks. The available honeypot and tools are briefly listed and explained below.

## 8.2 Available Honeypots

* Adbhoney: Low interaction honeypot designed for Android Debug Bridge over TCP/IP. The purpose of this project is to provide a low interaction honeypot intended to catch whatever malware is being pushed by attackers to unsuspecting victims which have port 5555 exposed.
* Ciscoasa : A low interaction honeypot for the Cisco ASA component capable of detecting CVE-2018-0101, DoS and remote code execution vulnerability.
* CitrixHoneypot : Detects and logs CVE-2019-19781 scan and exploitation attempts.
* Conpot: Conpot is a low interactive server-side Industrial Control Systems honeypot.
* Cowrie: Cowrie is a medium to high interaction SSH and Telnet honeypot designed to log brute force attacks and the shell interaction executed by the attacker. In medium interaction mode (shell) it emulates a UNIX system in Python, in high interaction mode (proxy) it functions as an SSH and telnet proxy to monitor attacker behaviour to other systems.
* Dicompot : A Digital Imaging and Communications in Medicine (DICOM) Honeypot.
* Dionaea : Dionaea objective is to trap malware exploiting vulnerabilities exposed by services offered to a network, the goal is getting a copy of the malware.
* ElasticPot : This is an elasticSearch Honeypot. It emulates GET PUT, POST, DELETE requests.
* Glutton: Generic Low Interaction Honeypot
* Heralding : It is a simple honeypot that just collects credentials. The following protocols are supported: ftp, telnet, ssh, rdp, http, https, pop3, pop3s, imap, imaps, smtp, vnc, postgresql and socks5.
* Honeypy : A low interaction honeypot with the ability to be more of a medium interaction honeypot.
* Honeysap : HoneySAP is a low-interaction research-focused honeypot specific for SAP services.
* Honeytrap : Honeytrap is an extensible and open source system for running, monitoring, and managing honeypots.
* Ipphoney: This is a honeypot simulating a printer that supports the Internet Printing Protocol and is exposed to the Internet. It uses ideas from various other honeypots, like ADBHoneypot (for output plugin support), Citrix Honeypot (for general structure), and Elasticpot.
* Mailoney : An SMTP Honeypot.
* Medpot: HL7 / FHIR honeypot.
* Rdpy: RDP Honeypot.
* Snare/Tanner: Tanner is a remote data analysis and classification service to evaluate HTTP requests and write the response then served by snare. Tanner uses multiple application vulnerability type emulation techniques when providing responses for SNARE, thus fuelling the luring technique of the latter.

## 8.3 Integrated Tools

* Suricata :Suricata is a free and open source, fast, and robust network threat detection engine. The Suricata engine is capable of real time intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM) and offline pcap processing.
* Cyberchef : A useful web app loaded with encryption, encoding, forensics, compression, and data analysis.
* Cockpit : A lightweight, web user interface for docker, operating system, real-time performance monitoring and web terminal.
* Kibana : Kibana is a free data visualization and exploration tool that can be used for log and time-series analytics, application monitoring, and operational intelligence. Histograms, line graphs, pie charts, heat maps, and built-in geospatial support are among the powerful and simple-to-use tools.
* ELK stack: Visualising all the events captured by T-Pot.
* Elasticsearch Head : A web front end for browsing and interacting with an ElasticSearch cluster.
* Spiderfoot: It is an open-source intelligence automation tool

# 9 Analysis

Once the honeypot was deployed with open ports, attacks came quickly. It is evident that bots are actively scanning cloud IP ranges. The first fifteen minutes that the honeypot was up, one can observe that port 8088 (Asterisk Management Port) was attacked the most.

Port 8088/TCP is used for software update service on Mac OS X Server v10.4 and later(radan-http), web service and iTunes Radio streams(irdmi)

Port 8088/UDP is used for EMC2 (Legato) Networker or Sun Solcitice Backup (Official) and QuickTime Streaming Server over radan-http.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 13 First 15 mins of deployment

## 9.1 ADB Honey Analysis

We left the honeypot running and collecting data for over a week. As mobile phone popularity has been ever increasing in the recent years. We decided to look first at ADB honey, designed for Android Debug Bridge over TCP/IP. There were 3068 attacks with 1123 unique Ips.

Chart

Description automatically generated

Figure 14 ADB attacks in one week

A screenshot of a video game

Description automatically generated

Figure 15 ADB attacks histogram

### 9.1.1 Examining Captured sample

A closer look at the ADB dashboard showed us, the captured samples. There were several counts on some samples captured. Those hashes were compared to VirusTotal database.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 16 ADB Captured Samples

When those collected samples was cross referenced from Virus Total, all of them resulted to be crypto miners.

Graphical user interface, application, Teams

Description automatically generated

Figure 17 Hash search on Virus Total

Graphical user interface, application

Description automatically generated

Figure 18 Hash2 search in Virus Total

According to VirusTotal behaviour analysis, the malicious sample resolves to the DNS coinhive.com or ws015.coinhive.com. We will further investigate this below.

Graphical user interface, text, application

Description automatically generated

Figure 19 Virustotal behaviour analysis on hash

Continuing the analysis of the ADB dashboard, we set out to look at the command line input by the attackers. There were few entries that looked interesting as illustrated in Figure 20.

A picture containing text, screenshot, indoor, screen

Description automatically generated

Figure 20 ADB Command Line Input

We can observe that there were 42 counts of entering the command illustrated in the red rectangle above. Further investigation based on the above stated IP was carried out using URLhaus [54] and Shodan [55]. The IP originates from Germany, is active and known to be malicious and associated with the new Mirai variant [56].

Graphical user interface, application

Description automatically generated

Figure 21 URLhaus Malicious database search

Graphical user interface, website

Description automatically generated

Figure 22 Shodan IP Search

Further investigation revealed that a bash file w.sh is downloaded to the infected system. It contains different architecture of android and has read, write, and execute permission(chmod 777). This is potentially very dangerous, as this bash file will execute in privileged mode by looping through all the android architecture when it gets downloaded. Possibility of backdoor and remote code execution may arise in this case.

Text

Description automatically generated

Figure 23 Bash file retrieved from ADB command input

### 9.1.2 UFO Crypto Miner

Additional investigation is conducted on the yellow rectangle section as previously when comparing the hashes to VirusTotal database, a crypto miner exhibited a similar command input while analysing its behaviour.

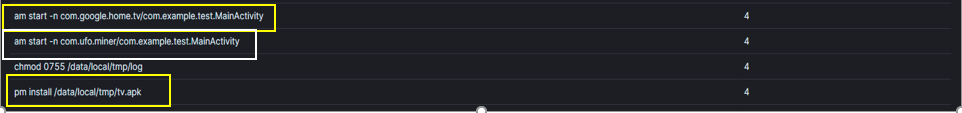


Figure 24 Android Crypto Miner command input

As per Sophos Security report [57], the volume of Android IoT devices infected by small but highly effective cryptocurrency miner applets is increasing, as hackers attempt to exploit devices that are left to do their own thing with little user supervision. Casting dongles, smart TVs, smart speakers, central home controllers, and other gadgets fall under this category. Since those devices are still dependent on the internet and kept on, an intruder might use them to install crypto miners and profit from other people's resources. As android phones have more power, they will be better for this, but the Android Debug Bridge(ADB) is hidden behind the “USB Debugging” option, which a user must activate. However, since Android IoT devices lack that layer, attackers may use known (or unknown) remote code execution vulnerabilities to send commands to them. When a device is infected with the "UFO" crypto miners, the user will not notice anything. There is no app icon or indication. However, since “UFO” makes use of 99 percent of the available resources, the system will become noticeably hot and sluggish.

Graphical user interface

Description automatically generated with medium confidence

Figure 25 AVD running process, UFO miner at 99%

The app uses the Android WebView browser to load a single JavaScript script from Coinhive on a Web page embedded within the app, named run.html, illustrated in Figure 26. According to the research, the UFO miner's creator programmed CPU throttling to 80 percent, but testing revealed that this did not work for some reason, as the mining load remained at 99 percent as illustrated in Figure 25.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 26 Coinhive Malicious Script

Running a crypto miner on an Android IoT device 24/7 involves stressing its resources and making it hotter than it should be. Therefore, causing hardware deterioration and subsequent failure If the victim does not detect the malicious activity for a prolonged period. Sophos researchers also identified a growing group of attackers that attempted to deliver Linux shell scripts using the same remote code execution exploits used by UFO Miner.

Text

Description automatically generated

Figure 27 Crypto attack group killing rival groups attack

These shell scripts inspect the device's processor architecture before downloading a bot in the form of a Linux ELF program that is adapted for that specific architecture. Since the scripts include shell commands to delete UFO Miner, as well as bots delivered by rival botnet groups, these attackers seem to be aware of the app's growth. Sophos researchers advised for a complete factory-reset. Doing so, will roll back the infection, while blocking access to “coinhive.com” on the network echelon, thus preventing any further exploitation.

## 9.2 Suricata CVE Investigation

Now that we have looked at ADB dashboard and attacks, we will further look at the Tpot dashboard to investigate separate attacks. Based on the collected 7-day data, we can observe that attackers are targeting mostly port 445(SMB), port 22(SSH), port 80(HTTP), port 443(Hypertext Transfer Protocol over SSL/TLS) and port 25(SMTP). Successfully attacks on these ports can cause serious denial of service, resulting in unavailability.

A screenshot of a computer

Description automatically generated

Figure 28 Overview of 7 Day Data collection

Diving further into our investigation, we can observe that suricata has picked up numerous old CVE dating as far as 2001. The CVE-2001-0540 had a maximum attack count of 5721 in total. This causes memory leak in Terminal servers in Windows NT and Windows 2000 allowing remote attackers to cause a denial of service (memory exhaustion) via many malformed Remote Desktop Protocol (RDP) requests to port 3389. CVE-2012-0152, with the second number of attacks looks for a Remote Desktop Protocol (RDP) service in Microsoft Windows Server 2008 R2 and R2 SP1 and Windows 7 Gold and SP1 allowing remote attackers to cause a denial of service (application hang) via a series of crafted packets, also known as "Terminal Server Denial of Service Vulnerability." CVE-2002-0013 looks for Vulnerabilities in the SNMPv1 request handling of many SNMP implementations allowing remote attackers to cause a denial of service or gain privileges via GetRequest, GetNextRequest and SetRequest messages. All these is a clear indication that attackers are trying to scan and crash old servers that have old vulnerabilities to narrow their scanning and attack scope.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 29 CVE Top 10

## 9.3 Suricata Alert signatures

We can further observe that there were 7423 attempts to install the DoublePulsar [58] Backdoor in the alert signature section, highlighted in red. Initially this backdoor was developed by the U.S. National Security Agency's (NSA) Equation Group but was leaked by hacker group The Shadow Brokers in early 2017. This backdoor runs in Kernel mode, granting a high privilege level for attackers. Once installed, it employs three commands: Ping, kill and Execute, allowing malware to be deployed on servers and systems. The area highlighted in yellow is also of great concern as suricata detected reassembly sequence gaps, mostly missing packets. This can indicate malicious TCP traffic. [59]

A screenshot of a computer

Description automatically generated

Figure 30 Alert Signature

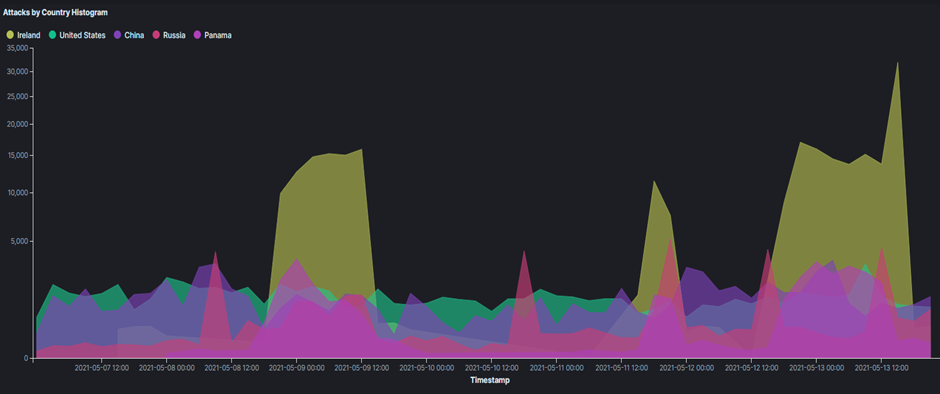
## 9.4 Username and Password attacks

Mirai Botnet and other variant IoT botnets are being used for DDoS attacks and they not only generate a colossal volume of attack traffic but also target insecure devices to compromise them by password brute forcing attack. The Autonomous system number(AS/N) shows clearly that attackers have been targeting ISPs, hosting providers, and DNS providers. We had a huge volume of attacking traffic from Ireland, United states, china and so forth, indicating that attackers are using cloud services to run botnets or attacks.

A screenshot of a computer

Description automatically generated with medium confidence

*Figure 31 ASN of Attacks*



*Figure 32 Attacks by Country Histogram*

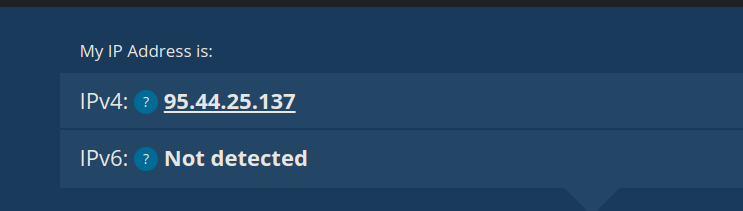
Graphical user interface

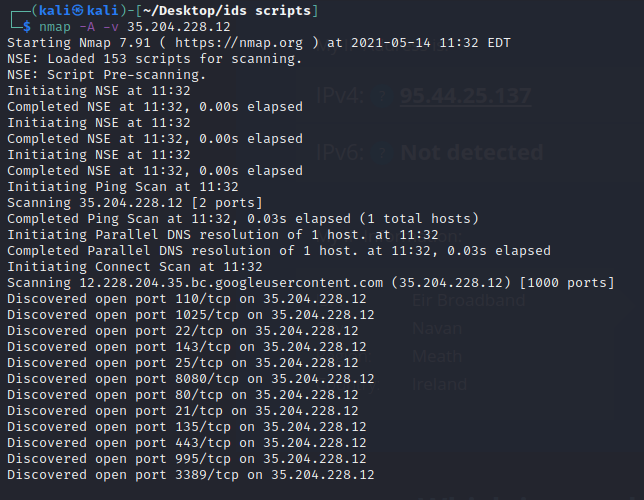
Description automatically generated

*Figure 33 Credential Brute-Force*

# 10 Testing / IDS Script

In the testing phase, we began with an isolated network, before opening to the public. In the isolated testing, we started by going through the logs and making sure there were no unwanted IP addresses. Once we were satisfied that the only IP’s connected were ours, we started to scan and attack the machine. A static ip address of 95.44.25.137 was used to attack and test the script initially.





The scan showed us all the open ports of the machine, which an attacker could enter. Once the machine was scanned we started to look at the log files to see if the attackers IP address would show up.



From the image above we can see the attackers IP address in the log files. The next step was to create scripts to block any unwanted IP address to the machine.

The first script we created was a bash script to grab every IP address that is in the log files. After every IP address was obtained, we added more code to only split and merge unique IP’s into a text file. Once every unique address was added, we made sure that our own IP address and the loopback address was excluded from that list, so we don't get locked out of the machine in future while trying to connect to it.

When the list of addresses we needed to block was created, we made a bash script to block every IP address with iptables.



Table

Description automatically generated

Figure 34 Open ports

## 10.4 IoT Security Hardening

IoT security hardening measures include:

* Firmware updates
* Block unnecessary ports
* Disable telnet
* Use encryption communication such as SSL/TLS
* Use strong passwords
* Use encryption in drivers
* User account lockout
* Periodic assessment of devices
* Secure password recovery
* Two-factor authentication
* Disable UPnP

# 11. Conclusion

As more and more people are migrating to smaller devices and Internet of Things platforms for their connectivity in daily life, attacks towards those systems are alarmingly increasing. This is due to the fact that they are mostly left unattended. Most IoT rarely gets automatic updates, and their default login is rarely changed. We have observed multiple logins attempts with default credentials to the honeypot. Also, mass scanning and numerous fragmented packets were intercepted by the IDS. Trojans, botnets, crypto miners, DDoS ,backdoors are some of the many tactics being used by cybercriminals. Using IPSec or Intra-Cloud Connect to gain remote access to IoT devices is a more secure option than using the public Internet. A cellular firewall is one way to avoid attempts to get remote access to IoT devices as well as fully block attacks. Devices are only allowed to communicate with a given subset of IP addresses while using a cellular firewall.

## 11.1 Future Work

At this point, the functionality of the real time defence is good, although there is room for improvement. The main area is a fully functioning custom IDS (based off of the code work mentioned in section 12) that continuously listens on either the main network interface or all network interfaces. This IDS would also have a working logging feature that uses the timestamp module within the work above for a naming system along with the source/attacker IP address, and that it would search the already created log names to see if there were a match with the IP addresses.

Following these fixes, the bash script would also be updated to search through the time stamped files and take out the source Ip’s (similar to what its current functionality is) and add them to a previously defined firewall rule to drop that IP address. It would also be easily modified to work with any firewall solution, as long as the firewall has command line interface functionality.

A modification of the scripts to work in tandem with fail2ban would also be an option, so that if an IP address tried to login more than 5 times per minute, it would automatically add that IP address to the IP addresses to block, where the second script would then ban the IP address.

## 11.2 Drawbacks

Initially we used Digital Ocean to host the Tpot. We encountered several issues as per the documentation Tpot needs over 8gb RAM to perform. We encountered runtime out of memory issues. Kibana and Logstash were not able to forward the logs to Elastic search. So, we could not manually extract the numerous fields collected in the logs file to examine properly. Also using Digital Ocean tended to go offline several times periodically which resulted in rebuild and redeploy, instead of rebooting the honeypot. After much troubleshooting, we moved the honeypot to google cloud platform, which resolved the memory and log passing issue. Secondly the initial code implemented did work out, so a new scripting methodology had to be implemented. Further details has been provided in the code section below.

A screenshot of a computer

Description automatically generated with medium confidence

*Figure 35 Digital Ocean Issues*

# 12. Code

## 12.1 Initial Code testing

The detector script was to act a detect incoming scans, it'll take the results of each scan and add it to a folder (either a pcap, or a json file so that it's easy to search) with a particular naming format, using the timestamp as the name.

The detector script is below:

#!/usr/bin/env python3

#Written by Eoin Finney

import dpkt, socket, sys, time, pcap, optparse

def flagz(tcp):

#return flags

lst=list()

if tcp.flags & dpkt.tcp.TH\_SYN != 0:

lst.append("SYN")

if tcp.flags & dpkt.tcp.TH\_RST != 0:

lst.append("RST")

if tcp.flags & dpkt.tcp.TH\_FIN != 0:

lst.append("FIN")

if tcp.flags & dpkt.tcp.TH\_ACK != 0:

lst.append("ACK")

print(lst)

'''

try:

a=open(sys.argv[1])

pcaps=dpkt.pcap.Reader(a)

except:

return

'''

syn=dpkt.tcp.TH\_SYN

rst=dpkt.tcp.TH\_RST

fin=dpkt.tcp.TH\_FIN

ack=dpkt.tcp.TH\_ACK

tcp=dpkt.tcp.TCP

udp=dpkt.udp.UDP

class port\_scan\_entry(object):

def \_\_init\_\_(self):

self.src\_ip=0

self.dst\_ip=0

self.timestamp=0

self.logged=False

self.ports=[]

self.ordered\_flag=0

class timeout\_entry\_list(list):

def \_\_init\_\_(self, maxsize, live\_time):

self.maxsize=maxsize

self.live\_time=live\_time

def append\_entry\_item(self, item):

if len(self) < self.maxsize:

super(timeout\_entry\_list, self).append((time.time(), item))

else:

b=self.collect()

if b:

super(timeout\_entry\_list,self).append((time.time(), item))

else:

raise ValueError

def collect(self):

a=time.time()

old\_time=[]

for item in self:

if(a-item[0] > self.live\_time):

old\_time.append(item)

for item in old\_time:

self.remove(item)

return len(old\_time)

class logger():

type\_of\_scan={

syn: "syn",

ack: "ack",

rst: "rst",

fin: "fin"

}

def \_\_init\_\_(self, logfile="/var/log/scanned"):

#logging

try:

self.scanned=open(logfile, 'a')

except:

raise ValueError

def steps(self,packet):

# if it doesn't have packet or an ip ignore

if not hasattr(packet, "ip"):

return

ip=packet.ip

if type(ip.data) not in (tcp, udp):

return

def logthescan(self, scan):

src\_ip, dst\_ip=socket.inet\_ntoa(struct.pack("A",scan.src)), socket.inet\_ntoa(struct.pack("A", scan.dst))

def log(self):

pc=pcap.pcap()

decode={pcap.DLT\_LOOP:dpkt.loopback.Loopback,

pcap.DLT\_EN10MB:dpkt.ethernet.Ethernet}[pc.datalink()]

print("listening on %s: %s" % (pc.name, pc.filter))

for timestamp, packet in pc:

self.process(decode(packet))

timestamp\_array = []

def timestamp\_stuff(pc):

for timestamp in pc:

timestamp\_array.append(timestamp)

ip=eth.data

if not ip:

continue

tcp=ip.data

if type(tcp) != dpkt.tcp.TCP:

continue

flag=flagz(tcp)

# source and dest

src\_ip=socket.inet\_ntoa(ip.src)

dst\_ip=socket.inet\_ntoa(ip.dst)

def run(self):

self.log()

def main():

o=optparse.OptionParser()

o.add\_option("-f", dest="logfile", help="Where to save logs to", default="/var/log/scanned")

arguments, options=o.parse\_args()

s=logger(options.logfile)

s.run()

if \_\_name\_\_=='\_\_main\_\_':

try:

while True:

try:

main()

except TypeError:

time.sleep(1)

print("Running...")

continue

except KeyboardInterrupt:

print("Stop

ping")

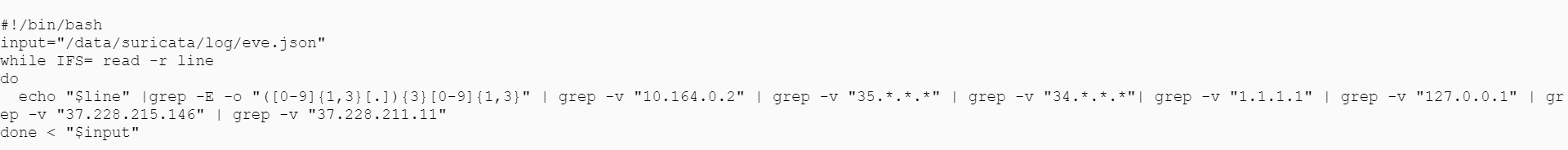
The above script didn't worked as intended, so we took another approach to filter and ban the malicious IPs

## 12.2 Code Scripts Walkthrough

Three scripts has been written namely block-ip.sh , sort-blockings.sh and ip-blocker.sh

First the block-ip.sh will list all the ips listed in the log path "/data/suricata/log/eve.json" from suricata IDS. Secondly duplicate ips will be dropped and a list of unique ips will be populated by running the sort-blocking.sh script. Once this is done . The last phase is to actually block the abusive ips by running ip-blocker.sh

### 12.2.1 block-up.sh



### 12.2.2 sort-blocking.sh



### 12.2.3 ip-blocker.sh



# Table of Figures

[Figure 1. The 5 Layers of IoT Architecture 7](#_17dp8vu)

[Figure 2. Device to Device Model 8](#_lnxbz9)

[Figure 3. Device to Cloud Model 8](#_1ksv4uv)

[Figure 4. Device to Gateway Model 8](#_2jxsxqh)

[Figure 5. Back-end Data sharing Model 9](#_3j2qqm3)

[*Figure 6. Insertion Attack* 14](#_1hmsyys)

[*Figure 7. Evasion Techniques* 14](#_2grqrue)

[Figure 8 Feature comparison between IDS and IPS 16](#_19c6y18)

[Figure 9 Host-based vs Network-based IPS/IDS feature comparison 17](#_28h4qwu)

[Figure 10 Bastion Architecture 21](#_3ygebqi)

[Figure 11. DMZ Architecture 22](#_3cqmetx)

[Figure 12 Multi-homed Architecture 23](#_4bvk7pj)

[Figure 13 First 15 mins of deployment 27](#_3hv69ve)

[Figure 14 ADB attacks in one week 27](#_4h042r0)

[Figure 15 ADB attacks histogram 28](#_2w5ecyt)

Figure 16 ADB Captured Samples 28

[Figure 17 Hash search on VirusTotal 29](#_2afmg28)

[Figure 18 Hash2 search in VirusTotal 29](#_pkwqa1)

[Figure 19 Virustotal behaviour analysis on hash 29](#_39kk8xu)

[Figure 20 ADB Command Line In](#_1opuj5n)

[30](#_1opuj5n)

[Figure 21 URLhaus Malicious database search 30](#_48pi1tg)

[Figure 22 Shodan IP Search 30](#_2nusc19)

[Figure 23 Bash file retrieved from ADB command input 31](#_1302m92)

[Figure 24 Android Crypto Miner command input 31](#_2250f4o)

[Figure 25 AVD running process, UFO miner at 99% 32](#_haapch)

[Figure 26 Coinhive Malicious Script 32](#_319y80a)

[Figure 27 Crypto attack group killing rival groups attack 32](#_1gf8i83)

[Figure 28 Overview of 7 Day Data collection 33](#_2fk6b3p)

[Figure 29 CVE Top 10 34](#_upglbi)

[Figure 30 Alert Signature 34](#_1tuee74)

[Figure 31 ASN of Attacks 35](#_2szc72q)

[Figure 32 Attacks by Country Histogram 35](#_184mhaj)

[Figure 33 Credential Brute-Force 35](#_3s49zyc)

[Figure 34 Open ports 36](#_45jfvxd)

[Figure 35 Digital Ocean Issues 37](#_4iylrwe)

# Table of Tables

[Table 1. Comparison between Firewall instances 20](#_46r0co2)

# References

|  |  |
| --- | --- |
| [1] | “Internet of things,” [Online]. Available: https://en.wikipedia.org/wiki/Internet\_of\_things.. |
| [2] | “Infrastructure\_applications of IoT,” [Online]. Available: https://en.wikipedia.org/wiki/Internet\_of\_things#Infrastructure\_applications. |
| [3] | R. Parashar, A. Khan and Neha, “A Survey: The Internet of Things,” *International Journal of Technical Research and Applications,* vol. 4, no. 3, pp. 251-257, 2016. |
| [4] | Security Today, “The IoT Rundown for 2020L Stats, Risks, and Solutions,” 13 January 2020. [Online]. Available: https://securitytoday.com/Articles/2020/01/13/The-IoT-Rundown-for-2020.aspx?Page=2. [Accessed May 2021]. |
| [5] | Juniper Research, “"Internet of Things" Connected Devices to Triple by 2021,” 13 December 2016. [Online]. Available: https://www.juniperresearch.com/press/internet-of-things-connected-devices-triple-2021. [Accessed May 2021]. |
| [6] | IBM, “Internet of Things,” [Online]. Available: https://www.topionetworks.com/topics/ibm-internet-of-things-5384c9e9fcefd2045300cf9d. [Accessed April 2021]. |
| [7] | Q. Stafford-Fraser, “The Trojan Room Coffee Pot,” May 1995. [Online]. Available: https://www.cl.cam.ac.uk/coffee/qsf/coffee.html. [Accessed April 2021]. |
| [8] | J. Kissell, “Getting to Know the Nabaztag Internet Rabbit,” 27 April 2008. [Online]. Available: https://tidbits.com/2008/04/27/getting-to-know-the-nabaztag-internet-rabbit/. [Accessed May 2021]. |
| [9] | “5 Layers of IoT architecture,” [Online]. Available: https://www.geeksforgeeks.org/5-layer-architecture-of-internet-of-things/. |
| [10] | “Architectural Considerations in Smart Object Networking,” [Online]. Available: https://www.rfc-editor.org/rfc/rfc7452.txt. |
| [11] | “Challenges in IoT Ecosystem,” [Online]. Available: https://iotsense.io/blog/iot-and-its-current-challenges/. |
| [12] | “OWASP IoT TOP10,” [Online]. Available: https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf. |
| [13] | “IoT attack surfaces list,” [Online]. Available: https://lexcia-com.appspot.com/iot-attack-surface-areas/. |
| [14] | “A Survey on various IoT Attacks,” [Online]. Available: http://ijercse.com/specissue/april-2018/27.pdf. |
| [15] | “Security attacks on IoT,” [Online]. Available: https://www.onetech.ai/en/blog/10-types-of-cyber-security-attacks-in-the-iot. |
| [16] | “Code Hopping,” [Online]. Available: https://hackaday.com/2016/03/06/rf-hacking-how-to-bypass-rolling-codes/. |
| [17] | “BlueBorne,” [Online]. Available: https://en.wikipedia.org/wiki/BlueBorne\_(security\_vulnerability). |
| [18] | “Exploit kits,” [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/exploits-malware. |
| [19] | “Jamming ACK attacks,” [Online]. Available: https://www.uncg.edu/cmp/faculty/j\_deng/papers/jack\_globecom08.pdf. |
| [20] | “Backdoor Exploit in IoT,” [Online]. Available: https://www.techradar.com/news/dangerous-backdoor-exploit-found-on-popular-iot-devices. |
| [21] | “Sybil attack in IoT,” [Online]. Available: http://ijcsi.org/papers/IJCSI-15-6-36-41.pdf. |
| [22] | “Replay attacks,” [Online]. Available: https://resources.infosecinstitute.com/topic/iot-radio-communication-attack/. |
| [23] | “Side Channel Attacks on IoT Applications,” [Online]. Available: https://research-information.bris.ac.uk/ws/portalfiles/portal/210500367/Final\_Copy\_2019\_03\_19\_Yan\_Y\_PhD.pdf. |
| [24] | “Shodan,” [Online]. Available: https://www.shodan.io/. |
| [25] | “Censys,” [Online]. Available: https://censys.io/. |
| [26] | “Thingful search engine,” [Online]. Available: https://www.thingful.net/. |
| [27] | “Nmap,” [Online]. Available: https://nmap.org/. |
| [28] | “HackRF one,” [Online]. Available: https://greatscottgadgets.com/hackrf/one/. |
| [29] | “RFCrack,” [Online]. Available: https://www.kitploit.com/2018/01/rfcrack-software-defined-radio-attack.html. |
| [30] | “Intrusion detection system,” [Online]. Available: https://en.wikipedia.org/wiki/Intrusion\_detection\_system. |
| [31] | “IDS Taxonomy,” [Online]. Available: https://neuro.bstu.by/ai/To-dom/My\_research/Paper-0-again/For-research/D-mining/Anomaly-D/Intrusion-detection/taxonomy.pdf. |
| [32] | “Signature based IDS,” [Online]. Available: https://www.omnisecu.com/security/infrastructure-and-email-security/types-of-intrusion-detection-systems.php. |
| [33] | “Anomaly based IDS,” [Online]. Available: https://www.omnisecu.com/security/infrastructure-and-email-security/types-of-intrusion-detection-systems.php. |
| [34] | “IDS best practice,” [Online]. Available: https://cybersecurity.att.com/resource-center/videos/ids-best-practices. |
| [35] | “Eluding Network IDS,” [Online]. Available: https://insecure.org/stf/secnet\_ids/secnet\_ids.html. |
| [36] | “Intrusion Prevention System,” [Online]. Available: https://www.geeksforgeeks.org/intrusion-prevention-system-ips/. |
| [37] | “HIDS vs NIDS comparison,” [Online]. Available: https://ebrary.net/26721/computer\_science/use\_ids. |
| [38] | “Firewall -,” [Online]. Available: https://en.wikipedia.org/wiki/Firewall\_(computing). |
| [39] | “Hardware vs Software Firewalls,” [Online]. Available: https://operavps.com/hardware-or-software-firewall/. |
| [40] | “Types of firewall,” [Online]. Available: https://phoenixnap.com/blog/types-of-firewalls. |
| [41] | “Cloud Firewall types,” [Online]. Available: https://www.cloudflare.com/learning/cloud/what-is-a-cloud-firewall/. |
| [42] | “Firewall as a service,” [Online]. Available: https://www.fortinet.com/resources/cyberglossary/firewall-as-a-service-fwaas. |
| [43] | “Next Generation Firewalls,” [Online]. Available: https://www.vmware.com/topics/glossary/content/next-generation-firewall. |
| [44] | “Platform as a service,” [Online]. Available: https://www.vmware.com/topics/glossary/content/next-generation-firewall. |
| [45] | “Cloud Firewall advantages,” [Online]. Available: https://pentestmag.com/advantages-and-disadvantages-of-cloud-security-main-benefits-current-risks-and-the-way-to-avoid-them/. |
| [46] | “Firewall Architecture,” [Online]. Available: https://ebrary.net/26717/computer\_science/firewall\_deployment\_architecture. |
| [47] | “Bastion Host,” [Online]. Available: https://en.wikipedia.org/wiki/Bastion\_host. |
| [48] | “DMZ,” [Online]. Available: https://en.wikipedia.org/wiki/DMZ\_(computing). |
| [49] | “Multi Homing,” [Online]. Available: https://en.wikipedia.org/wiki/Multihoming. |
| [50] | “Honeypot Practical Guide,” [Online]. Available: https://www.cs.wustl.edu/~jain/cse571-09/ftp/honey.pdf. |
| [51] | “Honeypot types,” [Online]. Available: https://ipwithease.com/understanding-types-and-benefits-of-honeypot-in-network-security/. |
| [52] | “Honeypots Technologies,” [Online]. Available: https://securitytrails.com/blog/top-20-honeypots. |
| [53] | “Tpot Honeypot,” [Online]. Available: https://github.com/telekom-security/tpotce. |
| [54] | “URL haus -Malicious URL sharing,” [Online]. Available: https://urlhaus.abuse.ch/. |
| [55] | “Shodan-search engine for internet connected devices,” [Online]. Available: https://www.shodan.io/. |
| [56] | “Mirai Variant,” [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/new-mirai-variant-found-spreading-like-wildfire. |
| [57] | “Sophos Security Report 2019,” [Online]. Available: https://news.sophos.com/en-us/2019/02/26/automated-android-attacks-deliver-ufo-cryptominer-trojan/. |
| [58] | “DoublePulsar Backdoor,” [Online]. Available: https://en.wikipedia.org/wiki/DoublePulsar. |
| [59] | “Reassembly sequence gap,” [Online]. Available: http://www.icsi.berkeley.edu/pubs/networking/robustTCPstream05.pdf. |
| [60] | R. Parashar, A. Khan and Neha, “A Survey: The Internet of Things,” International Journal of Technical Research and Applications, Faridabad, 2016. |
| [61] | Security Today, “The IoT Rundown for 2020: Stats, Risks and Solutions,” 13 January 2020. [Online]. Available: https://securitytoday.com/Articles/2020/01/13/The-IoT-Rundown-for-2020.aspx?Page=2. [Accessed May 2021]. |
| [62] | Juiper Research, “"Internet of Things" Connected Devices to Triple by 2021,” 13 December 2016. [Online]. Available: https://www.juniperresearch.com/press/internet-of-things-connected-devices-triple-2021. [Accessed May 2021]. |
| [63] | K. Ashton, “That 'Internet of Things' Thing,” RFID Journal, 22 June 2009. [Online]. Available: https://www.rfidjournal.com/that-internet-of-things-thing. [Accessed May 2021]. |
| [64] | IBM, “Internet of Things”. |