Analysis on IOT Botnet and DDOS attack

Md. Walid Bin Nur- 13101112

Supervisor: Sadia Hamid Kazi

Department of Computer Science and Engineering

BRAC University

DECLARATION

I hereby declare that this report is my own work and effort and it has not been submitted anywhere for any award. All the contents provided here is totally based on my own labor dedicated for the completion of this thesis. Where other sources of information have been used, they have been acknowledged and the sources of information have been provided in the reference section.

Signature of Supervisor:

Signature of Authors:

Department of Computer Science and Engineering
BRAC University

ACKNOWLEDGEMENT

First of all, I would like to express myheartfelt gratitude to almighty Allah. Secondly, I am sincerely grateful to my advisor, Sadia Hamid Kazi, for her patience and her insight whenever I felt stuck. Her guidance helped me in all parts of the progress.

Finally, I would like to express my sincere gratefulness to my beloved parents, brothers and sisters for their love and care. I am grateful to all of my friends who helped me directly or indirectly to complete my thesis.

Table of Contents

Chapter 5: Results and Conclusion

List of figures

ABSTRACT

IoT has the potential to affect our ways of life. It is the next step of Internet where all the physical objects around us will be connected to each other. According to Gartner, by 2020 there will be over 26 billion connected devices. However, the security of such a big network of interconnected devices is of paramount importance. According to a report from Russian-based Kaspersky Labs, botnets - not spam, viruses or worm- currently pose the biggest threat to the Internet.However, few works that have been done on this issue in the recentpast are not successful on themselves alone. In this paper, I present an in depth understanding of the problem and propose a mechanism to counter this issue. My proposal is based on Transductive Confidence Machines, which was previously proposed as a mechanism to provide confidence measures on classification decisions. It proposes to make use of this algorithm with the help of honeypot to collect attack data and uses these data to make the system more proficient.

Keywords: IoT, DDoS attack, Honeypot, TCM-kNN, Network communication simulation.

Chapter 1- Introduction:

A bot is an intelligent software application used to perform coordinated activities over the Internet. Whilst bots can be used to perform innocent, repetitive operations such as searching through quantities of data for web indexing, the majority of bots are created for malicious purposes. Bots work together to create a botnet of many thousands of infected hosts under the remote control of a human operator. This botnet serves as a single platform from which to launch massive co-ordinated attacks such as spam, distributed denial-of-service (DDoS), phishing and click-fraud. Distributed denial-of-service attacks pose an immense threat to the Internet, and many defense mechanisms have been proposed to combat the problem. Attackers constantly modify their tools to bypass these security systems, and researchers in turn modify their approaches to handle new attacks. The DDoS field is quickly becoming more and more complex, and has reached the point where it is difficult to see the forest for the trees. On one hand, this hinders an understanding of the distributed denial-of-service phenomenon. Multitudes of known attacks create the impression that the problem space is vast, and hard to explore and address. On the other hand, existing defense systems deploy various strategies to counter the problem, and it is difficult to assess their effectiveness and cost, and to compare them to each other.Anti-virus software and Intrusion Detection Systems use signature definitions of known malware to identify malicious code entering a system. Every signature definition is unique to a specific malware variant, requiring a malware sample to first be acquired, reverse engineered, then the binary analyzed for tell-tale code patterns that can be used to create that signature definition. Malware authors use a variety of mutation techniques to make minor changes to code whilst maintaining the attack profile, thus rendering the current signature definition useless in recognizing the new modified malware. Signature-based detection has little effect on a botnet. Whilst it can recognize a botnet executable binary and disinfect a compromised machine, botnets typically comprise of thousands of bots, whereby removing a single bot has little impact on the overall botnet.The signature-based methodlooks for a specific signature to match, signaling an intrusion. They can detect many or all known attack patterns, but they are of little use for as yet unknown attack methods. Most popular intrusion detection systems fall into this category. On the other hand anomaly detection focuses on detecting unusual activity patterns in the observed data. The approach I propose falls into the later category. Following this introduction, first, I describe the types of DDOS attacks, followed by different types of DDOS prevention algorithms, a brief introduction to honeypot and finally the proposed methodology.

Chapter 2- Literature review:

Muhammad Umar Farooq and Muhammad Waseem in their article, *A Critical Analysis of Security Concerns of IoT* discussed about how we will need a well-defined security infrastructure that can limit the possible threats related to scalability, availability and security of IoT. However, most IoT devices have weak security structure which leaves them wide open for attackers to use as botnet. IoT botnet is more severe than traditional botnet. Unlike traditional botnet, an IoT botnet doesn't necessarily need to consist of only computers but any other devices with the ability to transmit data over a network like toaster, or smart TV, surveillance camera. In late December 2013, a researcher at Proof point (a California-based enterprise security company) noticed that hundreds of thousands of malicious emails logged through a security gateway had originated from a botnet that included not only computers, but also other devices -- including smart TVs, a refrigerator and other household appliances. We've already seen CCTV cameras turned into botnet armies to launch DDoS attacks against banks and other targets. Unlike a desktop computer or laptop, it can be harder to know when your connected toaster has been enlisted in a bot army.How does it possible for DDoS pose threats to IoT network? This question could be considered from IoT's inherited features from the Internet as a network of connected devices and also from its distinct differences from the Internet. First, since the Internet is not designed to police intermediate traffic. Its end-to-end design paradigm make the intermediate network simple and optimized to ensure the fastest packet forwarding service while leave the complexity of packet processing to the hosts on the two ends of the communication. When proper detecting and preventive mechanism are missing on the receiver, the system becomes vulnerable to malicious packets streamed from the sender. In the IoT network, end devices are usually not equipped with high computational resources for implementing complex security algorithm and usually limited in power supply, which makes them not intelligent enough to detect and avoid network attack. Second, the service available on one IoT network component is limited, which means only certain number of requests could be served at one time. When malicious packets taking a large portion of the total requests, chances that legitimate requests being temporarily blocked becomes larger. Third, for the reason that IoT devices are connected via end network with relatively low bandwidth capacity compared with the intermediate network, it becomes easier to flood a target end network by dumping huge amount of packets from the faster intermediate network [14]. Compared with the public Internet, IoT network is exclusively designed to be opened for many types of devices.

Its loose control over the connected simple devices increases the risk of including malicious devices into the network. Moreover, for IoT, the work flow is highly dependent on the communication between the chained devices over the network. Single point failure would lead to cascade effect over an area of end network. For example, once DDoS attack brings down the serving device on an IoT network, the other IoT devices whose functions rely on the this blocked device will be also blocked from serving their client devices, which causes impairment of a local network [15].

Moreover, in contrast to the internet which majorly connects personal computers and mobile communication device, large numbers of heterogeneous devices are included in the IoT network. To seamlessly report the physical world to the digital world, those distributed heterogeneous devices need to communicate with each other at real time, which requires solution to ensure efficient and reliable end-to-end communication over the IoT network [16].

Chapter 3: DDoS

3.1 Types of DDoS attacks

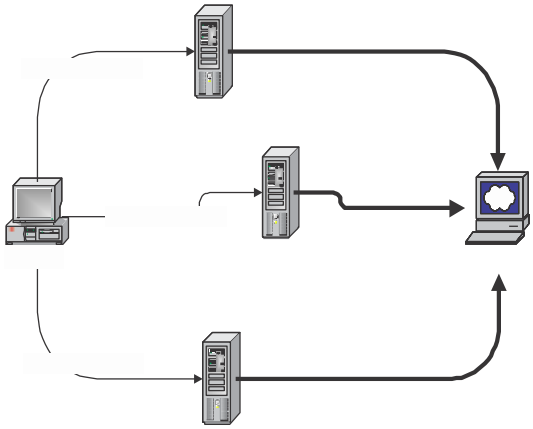


Figure1.OverviewofanDDoSAttackScenario.

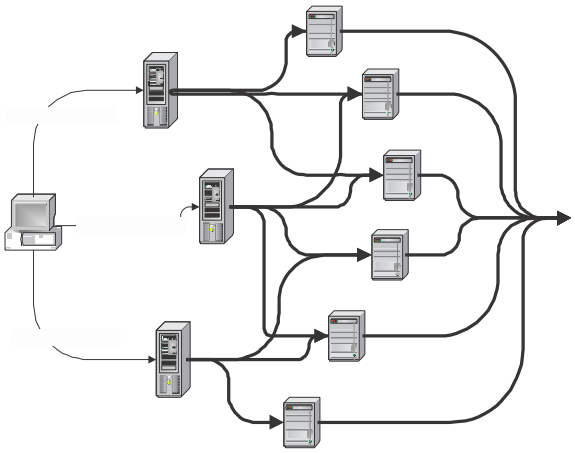


Figure2.DDoSAttackusingreflectors

This section provides an overview of major attack categories, as well as a breakdown of specific attack types within each.

3.1.1 Attacks Targeting Network Resources:

Attacks that target network resources attempt to consume all of a victim's network bandwidth by using a large volume of illegitimate traffic to saturate the company's Internet pipe. These attacks, called network floods, are simple yet effective. In a typical flooding attack, the offense is distributed among an army of thousands of volunteered or compromised computers—a botnet—that simply sends a huge amount of traffic to the targeted site overwhelms its network. In small numbers, requests of this manner may seem legitimate; in large numbers, they can be significantly harmful. A legitimate user trying to access a victim's site under a flooding attack will find the attacked site incredibly slow or unresponsive.

3.1.2 UDP Flood:

User Datagram Protocol (UDP) is a connectionless protocol that uses datagrams embedded in IP packets for communication without needing to create a session between two devices (in other words, it requires no handshake process). A UDP Flood attack does not exploit a specific vulnerability. Instead, it simply abuses normal behavior at a high enough level to cause congestion for a targeted network. It consists of sending a large number of UDP datagrams from potentially spoofed IP addresses to random ports on a target server; the server receiving this traffic is unable to process every request, and consumes all of its bandwidth attempting to send ICMP "destination unreachable" packet replies to confirm that no application was listening on the targeted ports. As a volumetric attack, a UDP flood is measured in Mbps (bandwidth) and PPS (packets per second).

3.1.3 ICMP Flood:

Internet Control Message Protocol (ICMP) is another connectionless protocol used for IP operations, diagnostics, and errors. Just as with a UDP flood, an ICMP flood (or Ping Flood) is a non-vulnerability based attack; that is, it does not rely on any specific vulnerability to achieve denial-of-service. An ICMP Flood can involve any type of ICMP message, such as a ping request (echo request and echo reply). Once enough ICMP traffic is sent to a target server, the server becomes overwhelmed from attempting to process every request, resulting in a denial-of-service condition. Like a UDP Flood, an ICMP Flood is also a volumetric attack, measured in Mbps (bandwidth) and PPS (packets per second).

3.1.4 IGMP Flood:

Internet Group Management Protocol (IGMP) is another connectionless protocol. It is used by IP hosts (computers and routers) to report or leave multicast group memberships for adjacent routers. An IGMP Flood is non-vulnerability based, as IGMP is designed to allow multicast. Such floods involve a large number of IGMP message reports being sent to a network or router, significantly slowing and eventually preventing legitimate traffic from being transmitted across the target network.

3.1.5 Amplification Attacks:

An Amplification attack takes advantage of a disparity between a request and a reply in technical communication. For instance, the attacker could use a router as an amplifier, taking advantage of the router's broadcast IP address feature to send messages to multiple IP addresses in which the source IP (return address) is spoofed to the target IP. Famous examples of amplification attacks include Smurf Attacks (ICMP amplification) and Fraggle Attacks (UDP amplification). Another example of a type of amplification attack is DNS amplification, in which an attacker, having previously compromised a recursive DNS name server to cache a large file, sends a query directly or via a botnet to this recursive DNS server, which in turn opens a request asking for the large cached file. The return message (significantly amplified in size from the original request) is then sent to the victim's (spoofed) IP address, causing a denial-of-service condition.

3.1.6 Connection-Oriented Attacks:

A connection-oriented attack is one in which the attacker must first establish a connection prior to launching a DDoS attack. The outcome of this attack usually affects the server or application resources. TCP- or HTTP- based attacks are examples of connection-oriented DDoS attacks.

3.1.7 Connectionless Attacks:

A connectionless attack, on the other hand, does not require the attacker to open a complete connection to the victim, and therefore is much easier to launch. The outcome of a connectionless attack affects network resources, causing denial of service before the malicious packets can even reach the server. UDP floods and ICMP floods are examples of connectionless DDoS attacks.

3.1.8 Reflective Attacks:

An attack is reflective when the attacker makes use of a potentially legitimate third party to send his or her attack traffic, ultimately concealing his or her own identity.

3.1.9 Attacks Targeting Server Resources:

Attacks that target server resources attempt to exhaust a server's processing capabilities or memory, potentially causing a denial-of- service condition. The idea is that an attacker can take advantage of an existing vulnerability on the target server (or a weakness in a communication protocol) to cause the target server to become so busy handling illegitimate requests that it no longer has the resources to handle legitimate ones. "Server" most commonly refers to a Website or Web application server, but these types of DDoS attacks can also target stateful devices, such as firewalls and intrusion prevention systems.

3.1.10 TCP/IP Weaknesses:

These types of attacks abuse the TCP/ IP protocol by exploiting some of its design weaknesses. They typically misuse the six control bits (or flags) of the TCP/IP protocol—SYN, ACK, RST, PSH, FIN and URG—in order to disrupt the normal mechanisms of TCP traffic. Unlike UDP and other connectionless protocols, TCP/IP is connection-based—requiring the packet sender to establish a full connection with his or her intended recipient prior to sending any packets. TCP/IP relies on a three-way handshake mechanism (SYN, SYN-ACK, ACK) where every request creates a half-open connection (SYN), a request for a reply (SYN-ACK), and then an acknowledgement of the reply (ACK). Attacks attempting to abuse the TCP/IP protocol will often involve sending TCP packets in the wrong order, causing the target server to run out of computing resources as it attempts to understand such abnormal traffic.

3.1.11 TCP SYN Flood:

In the TCP handshake mechanism, there must be an agreement between each party for a connection to be established. If the TCP client does not exist or is a non-requesting client with a spoofed IP, such an agreement is not possible. In a TCP SYN, or simple SYN flood attack, the attacking clients lead the server to believe that they are asking for legitimate connections through a series of TCP requests with TCP flags set to SYN coming from spoofed IP addresses. To handle each of these SYN requests, the target server opens threads and allocates corresponding buffers to prepare for a connection. It then tries to send a SYN-ACK reply back to the requesting clients to acknowledge their connection requests, but because the clients' IP addresses are spoofed or the clients are unable to respond, an acknowledgement (ACK packet) is never sent back to the server.

The server is still forced to maintain its open threads and buffers for each one of the original connection requests, attempting to resend its SYN-ACK request acknowledgement packets multiple times before resorting to a request timeout. Because server resources are limited and a SYN flood often involves a massive number of connection requests, a server is unable to time out its open requests before new requests arrive—causing a denial-of-service condition.

3.1.12 TCP RST Attack:

The TCP RST flag is intended to notify a server that it should immediately reset its corresponding TCP connection. In a TCP RST attack, the attacker interferes with an active TCP connection between two entities by guessing the current sequence number and spoofing a TCP RST packet to use the client's source IP (which is then sent to the server). Typically a botnet is used to send thousands of such packets to the server with different sequence numbers, making it fairly easy to guess the correct one. Once this occurs, the server acknowledges the RST packet sent by the attacker, terminating its connection to the client located at the spoofed IP address.

3.1.13 TCP PSH+ACK Flood:

When a TCP sender sends a packet with its PUSH flag set to 1, the result is that the TCP data is immediately sent or "pushed" to the TCP receiver. This action actually forces the receiving server to empty its TCP stack buffer and to send an acknowledgement when this action is complete. An attacker, usually using a botnet, can therefore flood a target server with many such requests. This overwhelms the TCP stack buffer on the target server, causing it to be unable to process the requests or even acknowledge them—resulting in a denial-of-service condition.

3.1.14 Low and Slow Attacks:

Unlike floods, "low and slow" attacks do not require a large amount of traffic. They target specific design flaws or vulnerabilities on a target server with a relatively small amount of malicious traffic, eventually causing it to crash. "Low and slow" attacks mostly target application resources (and sometimes server resources). By nature, they are very difficult to detect because they involve connections and data transfer appearing to occur at a normal rate.

3.1.15 Sockstress:

Sockstress is an attack tool that exploits vulnerabilities in the TCP stack—allowing an attacker to create a denial-of- service condition for a target server. In the normal TCP three-way handshake, a client sends a SYN packet to the server, the server responds with a SYN-ACK packet, and the client responds to the SYN-ACK with an ACK, establishing a connection. Attackers using Sockstress establish a normal TCP connection with the target server but send a "window size 0" packet to the server inside the last ACK, instructing it to set the size of the TCP window to 0 bytes. The TCP Window is a buffer that stores the received data before it uploads it up to the application layer. The Window size field indicates how much more room is in the buffer in each point of time. Window size set to zero means that there is no more space whatsoever and that the other side should stop sending more data until further notice. In this case, the server will continually send window size probe packets to the client to see when it can accept new information. But because the attacker does not change the window size, the connection is kept open indefinitely. By opening many connections of this nature to a server, the attacker consumes all of the space in the server's TCP connection table (as well as other tables), preventing legitimate users from establishing a connection. Alternately, the attacker may open many connections with a very small (around 4-byte) window size, forcing the server to break up information into a massive number of tiny 4-byte chunks. Many connections of this type will consume a server's available memory, also causing a denial of service.

3.1.16 SSL-Based Attacks Secure Socket Layer (SSL):

A method of encryption used by various other network communication protocols—as it grows in prevalence, attackers began targeting it. Conceptually, SSL runs above TCP/IP, providing security to users communicating over other protocols by encrypting communications and authenticating communicating parties. SSL-based DoS attacks take many forms: targeting the SSL handshake mechanism, sending garbage data to the SSL server or abusing certain functions related to the SSL encryption key negotiation process. SSL-based attacks could also simply mean that the DoS attack is launched over SSL-encrypted traffic, which makes it extremely difficult to identify. Such attacks are often considered "asymmetric" because it takes significantly more server resources to deal with an SSL-based attack than it does to launch one.

3.1.17 Encrypted-based HTTP Attacks (HTTPS floods):

Many online businesses increasingly use SSL/TLS (Transport Layer Security) in applications to encrypt traffic and secure end-to-end data transit. DoS attacks on encrypted traffic are on the rise, and mitigating them is not as obvious as might be expected. Most DoS mitigation technologies do not actually inspect SSL traffic, as it requires decrypting the encrypted traffic. HTTPS Floods—floods of encrypted HTTP traffic (see explanation below)—are now frequently participating in multi-vulnerability attack campaigns. Compounding the impact of "normal" HTTP Floods, encrypted HTTP attacks add several other challenges, such as the burden of encryption and decryption mechanisms.

3.1.18 THC-SSL-DoS:

Hacking group The Hacker's Choice (THC) developed this tool as a proof of concept to encourage vendors to patch SSL vulnerabilities. As with other "low and slow" attacks, THC-SSL-DoSrequires only a small number of packets to cause denial of service for even a fairly large server. It works by initiating a regular SSL handshake, and then immediately requesting for the renegotiation of the encryption key. The tool constantly repeats this renegotiation request until all server resources have been exhausted. Attackers love to launch attacks that use SSL because each SSL session handshake consumes 15 times more resources from the server side than from the client side. In fact, a single standard home PC can take down an entire SSL-based web server, while several computers can take down a complete farm of large, secured online services.

3.1.19 Attacks Targeting Application Resources:

Recent years have brought a rise in DoS attacks targeting applications. They target not only the well-known Hypertext Transfer Protocol (HTTP), but also HTTPS, DNS, SMTP, FTP, VOIP and other application protocols that possess exploitable weaknesses allowing for DoS attacks. Much like attacks targeting network resources, attacks targeting application resources come in a variety of flavors, including floods and "low and slow" attacks. Low and slow approaches are particularly prominent, mostly targeting weaknesses in the HTTP protocol—which, as the most widely used application protocol on the Internet, is an attractive target for attackers.

3.1.20 HTTP Flood:

The most common DDoS attack targeting application resources. It consists of what seem to be legitimate, session-basedsets of HTTP GET or POST requests sent to a victim's Web server, making it hard to detect. HTTP flood attacks are typically launched simultaneously from multiple computers (volunteered machines or bots). These bots continually and repeatedly request to download the target site's pages (HTTP GET flood), exhausting application resources and resulting in a denial-of-service condition. Modern DDoS attack tools, such as High Orbit Ion Cannon (HOIC), offer an easy-to-use means of performing multi-threaded HTTP flood attacks.

3.1.21 DNS Flood:

It is easy to launch yet difficult to detect. Based on the same idea as other flooding attacks, a DNS flood targets the DNS application protocol by sending a high volume of DNS requests. Domain Name System (DNS) is the protocol used to resolve domain names into IP addresses; its underlying protocol is UDP, taking advantage of fast request and response times without the overhead of having to establish connections (as TCP requires). In a DNS flood, the attacker sends multiple DNS requests to the victim's DNS server directly or via a botnet. The DNS server, overwhelmed and unable to process all of its incoming requests, eventually crashes.

3.1.22 Low and Slow Attacks:

The characteristics of the "low and slow" attacks in this section relate particularly to application resources (whereas the previous "low and slow" attacks targeted server resources). These "low and slow" attacks target specific application vulnerabilities, allowing an attacker to stealthily cause denial of service. Not volumetric in nature, such attacks can often be launched with only a single machine. Additionally, because these attacks occur on the application layer, a TCP handshake is already established, successfully making the malicious traffic look like normal traffic traveling over a legitimate connection.

3.1.23 Slow HTTP GET Request:

The idea behind a slow HTTP GET request is to dominate all or most of an application's resources through the use of many open connections, preventing it from providing service to users wishing to open legitimate connections. In this attack, the attacker generates and sends incomplete HTTP GET requests to the server, which opens a separate thread for each of these connection requests and waits for the rest of the data to be sent. The attacker continues to send HTTP header data at set, but slow, intervals to make sure the connection stays open and does not timeout. Because the rest of the required data arrives so slowly, the server perpetually waits, exhausting the limited space in its connection table and thereby causing a denial-of-service condition.

3.1.24 Slow HTTP POST Request:

To carry out a slow HTTP POST request attack, the attacker detects forms on the target website and sends HTTP POST requests to the Web server through these forms. The POST requests, rather than being sent normally, are sent byte by byte. As with a slow HTTP GET request, the attacker ensures that his or her malicious connection remains open by regularly sending each new byte of POST information slowly at regular intervals. The server, aware of the content length of the HTTP POST request, has no choice but to wait for the full POST request to be received (this behavior mimics legitimate users with slow Internet connection). The attacker repeats this behavior many times in parallel, never closes an open connection, and after several hundred open connections, the target server is unable to handle new requests—achieving a denial-of-service condition.

3.1.25 Regular Expression DoS Attacks:

A special case of "low and slow" attacks is RegExDoS (or ReDoS) attacks. In this scenario, the attacker sends a specially crafted message (sometimes called evil RegExes) that leverages a weakness in a library deployed in the server, in this case, a regular expression software library. This causes the server to consume large amounts of resources while trying to compute a regular expression over the user-provided input, or to execute a complex and resource-hungry regular expression processing dictated by the attacker.

3.1.26 Hash Collisions DoS Attacks:

This kind of attack targets common security vulnerabilities in Web application frameworks. In short, most application servers create hash tables to index POST session parameters. Sometimes application servers must manage hash collisions when similar hash values are returned. Collision resolutions are resource intensive, as they require an additional amount of CPU to process the requests. In a Hash Collision DoS attack scenario, the attacker sends a specially crafted POST message with a multitude of parameters. The parameters are built in a way that causes hash collisions on the server side, slowing down the response processing dramatically. Hash Collisions DoS attacks are very effective and could be launched from a single attacker computer, slowly exhausting the application server's resources.

3.2 DISTRIBUTED DENIAL OF SERVICE ALGORITHMS

The following are the various types of DDOS Algorithms available.

1) ASA: It is hypothesized that mimicking natures principles, and is not its epiphenomena, leads to better algorithms. The so called adaptive sampling framework is used for analyzing the 3-SAT problem, which led to one neural method and five mixed methods, that mix elements of EC and of NC in different ways. These methods have been tested against the best currently know incomplete 3-SAT algorithm. The basic principle behind these algorithms is to sample isolated features from the problem and from candidate solutions, which are adapted iteratively. Evolutionary and neural computations are the two main subfields of this class. This class of algorithms has been dubbed adaptive sampling, and it is described by the adaptive sampling framework.

2) CA: Cluster algorithm or clustering is the task of grouping a set of objects in such a way that objects in the same group called cluster are more similar in some sense or another to each other than to those in other groups clusters. The appropriate clustering algorithm and parameter settings including values such as the distance function to use, a density threshold or the number of expected clusters depend on the individual data set and intended use of the results. Clustering algorithms can be categorized based on their cluster model, there is no objectively "correct" clustering algorithm, but as it was noted, "clustering is in the eye of the beholder".

The most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one cluster model over another. It should be noted that an algorithm that is designed for one kind of models has no chance on a data set that contains a radically different kind of models. Example: k-means cannot find non-convex clusters [2].

3) TCM-KNN: Transductive confidence machines for K-Nearest Neighbors algorithm to fulfill DDOS attacks detection task towards ensuring the Qos of wed server. The method is good at detecting network anomalies with high detection rate, high confidence and low false positives than traditional methods, because it combines "strangeness" with "p-values" measures to evaluate the network traffic compared to the conventional ad-hoc thresholds based detection and particular definition based detection. The input feature spaces of TCM-KNN to effectively detect DDOS attack against web server. Finally, we introduce genetic algorithm (GA) based instance selection method to boost the real time detection performance of TCM-KNN and thus make it be an effective and lightweight mechanism for DDOS detection for web servers[3].

4) CA: Congestion algorithms to detect upsurges in traffic that can give rise to DOS but this approach may apply only simplistic signatures and also requires state information to be held on the nodes which is not a feasible solution in sensors because of limited memory [4].

5) DSA: In computer science, streaming algorithms are algorithms for processing data streams in which. The input is presented as a sequence of items and can be examined in only a few passes typically just one. These algorithms have limited memory available to them much less than the input size and also limited processing time per item. These constraints may mean that an algorithm produces an approximate answer based on a summary or "sketch" of the data stream in memory. Streaming algorithms have several applications in networking such as monitoring network links for elephant flows, counting the number of distinct flows, estimating the distribution of flow sizes [5].

6) MLA: A branch of artificial intelligence is about the construction and study of systems that can learn from data. For example, a machine learning system could be trained on email messages to learn to distinguish between spam and non-spam messages. After learning, it can then be used to classify new email messages into spam and non-spam folders. The core of machine learning deals with representation and generalization. Representation of data instances and functions evaluated on these instances are part of all machine learning systems. Generalization is the property that the system will perform well on unseen data instances; the conditions under which this can be guaranteed are a key object of study in the subfield of computational learning theory.

There is a wide variety of machine learning tasks and successful applications. Optical character recognition, in which printed characters are recognized automatically based on previous examples, is a classic example of machine learning.

7) RA: As mentioned above, the shortest paths are calculated using suitable algorithms on the graph representations of the networks. Let the network be represented by graph G (V, E) and let the number of nodes be 'N'. For all the algorithms discussed below, the costs associated with the links are assumed to be positive. A node has zero cost itself. Further, all the links are assumed to be symmetric, i.e. if di,j = cost of link from node i to node j, then d i,j = d j,i. The graph is assumed to be complete. If there exists no edge between two nodes, then a link of infinite cost is assumed. The algorithms given below find costs of the paths from all nodes to a particular node. The problem is equivalent to finding the cost of paths from a source to all destinations.

8) RSA: A node will initiate a distributed lookup according to the specific p2p routing substrate algorithm. A query message or object key lookup takes O (log N) application layer hops from source to destination. Each node has a routing table with O (log N) entries where each node entry maps a node identifier to an IP address and port number. Using routing table, each intermediate node along the routing path will forward the message to the best node in its routing table among all the candidate nodes stored as routing table entries. Here the best node in the routing table is specific to the particular routing algorithm [6]. 9) TTA: To trace back the source of the DDOS attacks in the internet is extremely hard. It is one of the extraordinary challenge to trackback the DDOS attacks, that attackers generate huge amount of requests to victims through compromised computers zombies in order to denying normal services or degrading the quality of services. IP trace back means the capability of identifying the actual source of any packet across the internet; with the help of IP trace back schemes identify the zombies from which the DDOS attack packets entered the internet. 10) NCA: Due to increase in number of users on internet, many people want to attack other system resources. Competitors also want to make their web site more popular than others. So they want to attack the service of other's web site. They keep on logon to a particular web site more times, and then service provided by the web server performance keeps degraded. To avoid that one, this application maintains a status table. In that it keeps the IP addresses of current users and their status. If the particular IP address has been signed on for a first time, it makes the status as genuine user. For 2, 3, 4 it marks as Normal user. For the fifth time it makes the particular IP address status as Attacker. In the time calculations we are only consider 5 times

Chapter 4: Proposed Methodology

4.1 Approach:

I approach a modified TCM-KNN algorithm alongside honeypot for an IoT end network to recognize botnet and prevent DDoS attack. The design of the defense system is guided by how to collect attack data and intelligently detect and avoid DDoS attacks. First, I will introduce the principles about TCM, later the classical TCM-KNN algorithm and finally modified TCM-KNN algorithm alongside honeypot to prevent DDoS attack.

4.2 Honeypot:

The exact definition of a honeypot is contentious, however most definitions are some form of the following:

A honeypot is an "an information system resource whose value lies in unauthorized or illicit use of that resources"(from the www.securityfocus.com forum)

A more practical, but more limiting, definition is given by pcmag.com:

"A server that is configured to detect an intruder by mirroring a real production system. It appears as an ordinary server doing work, but all the data and transactions are phony. Located either in or outside the firewall, the honeypot is used to learn about an intruder's techniques as well as determine vulnerabilities in the real system"

In practice, honeypots are computers which masquerade as unprotected. The honeypot records all actions and interactions with users. Since honeypots don't provide any legitimate services, all activity is unauthorized.

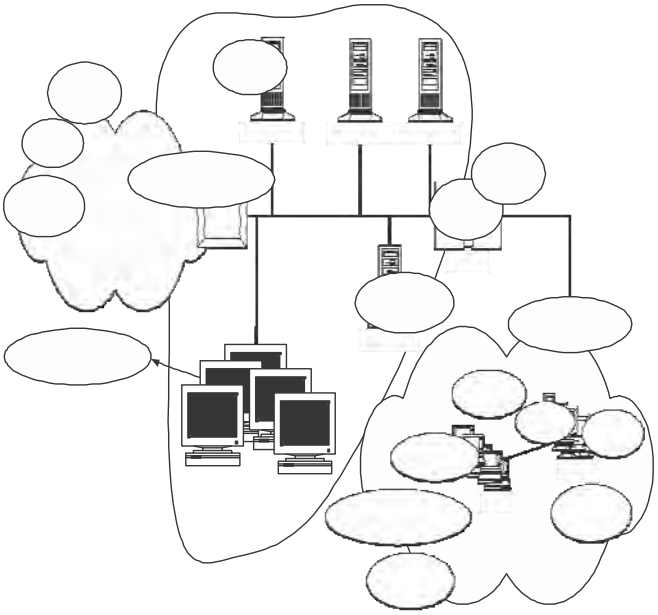


Figure3(a):Views in the organization

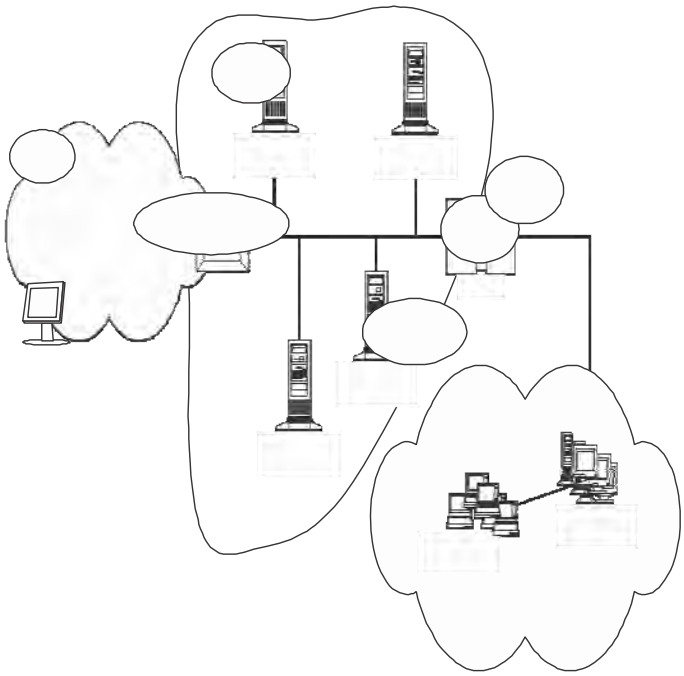


Figure3(b): Views of the attacker

Figure3. Two Different Views of the Honeypot

4.2.1 Types of Honeypots

Primarily, there are two categories of honeypots, high-interaction and low-interaction. These categories are defined based on the services, or interaction level, provided by the honeypot to potential hackers. **High-interaction honeypots** let the hacker interact with the system as they would any regular operating system, with the goal of capturing the maximum amount of information on the attacker's techniques. Any command or application an end-user would expect to be installed is available and generally, there is little to no restriction placed on what the hacker can do once he/she comprises the system. On the contrary, **low-interaction honeypots** present the hacker emulated services with a limited subset of the functionality they would expect from a server, with the intent of detecting sources of unauthorized activity For example, the HTTP service on a low-interaction honeypot would only support the commands needed to identify that a known exploit is being attempted. Some authors classify a third category, medium-interaction honeypots, as providing expanded interaction from low-interaction honeypots but less than high-interaction systems. A **medium-interaction honeypot** might more fully implement the HTTP protocol to emulate a well-known vendor's implementation, such as Apache.

However, there are no implementations of a medium-interaction honeypots and for the purposes of this paper, the definition of low-interaction honeypots captures the functionality of medium-interaction honeypots in that they only provide partial implementation of services and do not allow typical, full interaction with the system as high-interaction honeypots.

4.2.2 General Honeypot Advantages and Disadvantages

Honeypots provide several advantages over other security solutions, including network intrusion detection systems:

- Fewer false positives since no legitimate traffic uses honeypot
- Collect smaller, higher-value, datasets since they only log illegitimate activity
- Work in encrypted environments
- Do not require known attack signatures, unlike IDS

Honeypots are not perfect, though:

- Can be used by attacker to attack other systems
- Only monitor interactions made directly with the honeypot - the honeypot cannot detect attacks against other systems
- Can potentially be detected by the attacker

Tradition security solutions, such as intrusion detection systems, may not be enough in light of more complicated attacks. Honeypots provide a mechanism for detecting novel attack vectors, even in encrypted environments. Advances such as virtualization have made honeypots even more effective. Honeypots have drawbacks, though, so it is important to understand how honeypots operate in order to maximize their effectiveness [12].

4.3 TCM

A simple interpretation of transduction [8] is that unknown estimates for individual points of interest can be made directly from the training data, as opposed to using induction to infer a general rule for them. In our case, the estimates we are interested in learning are the likelihood that a point belongs to a given cluster in the current clustering model. In that sense, our “training” examples are the points already clustered.

Transduction has been previously used to offer confidence measures for the decision of labeling a point as belonging to a set of pre-defined classes (see [6, 7, 1]). TCM [1] introduced the computation of the confidence using Algorithmic Randomness Theory [9]. The confidence measure used in TCM is based upon universal tests for randomness, or their approximation. A MartinLof randomness deficiency test [9] based on such tests is a universal version of the standard p-value notion, commonly used in statistics. Martin-Lof proved that there exists a universal test for randomness smaller than any other test up to a multiplicative constant. Unfortunately, universal tests are not computable, and have to be approximated using non-universal tests called p-values. In the literature of significance testing, the p-value is defined as the probability of observing a point in the sample space that can be considered more extreme than a sample of data. This p-value serves as a measure of how well the data supports or not a null hypothesis. The smaller the p-value, the greater the evidence against the null hypothesis.Users of transduction as a test of confidence have approximated a universal test for randomness (which is in its general form, non-computable) by using a p-value function called strangeness measure [1] (or non-conformity score [10]). In truth, there is more than a single definition of strangeness measure, and in general, its definition depends on the base model used to construct the TCM. The general idea is that the strangeness measure corresponds to the uncertainty of the point being measured with respect to all the other labeled examples of a class: the higher the strangeness measure, the higher the uncertainty.

4.4 TCM-KNN

Now, I'll introduce the formal description of TCM-KNN problem for the DDoS detection. In the next section, I will further give the improved TCM-KNN algorithm for our anomaly detection based on this section. To my knowledge, it has never been applied to prevent DDoS in an IoT network.

Let's assume there isa intrusion detection training set $\{(x_1,y_1),...,(x_n, y_n)\}$ of n elements, where $X_i = \{x_i^1,x_i^2,....,x_i^n\}$ is the set of feature values (such as the connection duration time, the SYN error numbers, etc.) extracted from the raw network packet (or network flow such as TCP flow) for point i andy_i is the classification for point i , taking values from a finite set of possible classifications like different types of DDoS attack.

First, a measure called the individual strangeness measure will be assigned to every point. It defines the strangeness of the point in relation to the rest of the points. Let's employ the definition of [11] for strangeness: the strangeness α_i of a point i with respect to a class yis given in Equation 1, where D_i^y is the sequence of distances between point i and points in the class y, D^y_{ij} being the j-th shortest distance. At the same time, D_i^{-y} represents the sequence of distances between i and points in other classes (different from y), D_{ij}^{-y} being the j-th shortest distance. So, the strangeness measure for a point i with label y is defined as-

$$a_{iy} = \frac{\sum_{j=1}^{k} D_{ij}^{y}}{\sum_{j=1}^{k} D_{ij}^{-y}} \text{ ----------(1)}$$

where k is the number of neighbors used. Thus, the measure for strangeness is the ratio of the sum of the k nearest distances from the same class to the sum of the k nearest distances from all other classes. This is a natural measure to use, as the strangeness of a point increases when the distance from the points of the same class becomes bigger or when the distance from the other classes becomes smaller [2].

The p-value is calculated as shown in Equation 2, where # represents the cardinality of the set given as an argument.

$$t(z1, z2, \cdots, zn) = \frac{(\#[i = 1, \cdots, n : a_{iy} \geq a_{ny}])}{n} \text{ ----------(2)}$$

If z_n is the point in question, the function t() will measure the probability of having points already in the class with strangeness greater than or equal to that of zn. In general, a p-value is the maximum probability under the null hypothesis of the test statistic assuming a value equal to the observed outcome or a value just as extreme or more extreme (with respect to the direction indicated by alternative hypothesis) than the observed outcome. So the smaller the p-value is, the smaller is the chance that the test statistic could have assumed a value as incompatible with the null hypothesis if the null hypothesis ("class y is a good fit for point i.") is true.

The algorithm TCM-KNN attempts to place a new point in each class of the problem. While doing that, it may force the updating of some of the α values for the training examples (concretely, this happens whenever the distance between the training example and the new point is less than the largest of the k distances that are used to compute the α). It then computes one p-value for each of the attempts (i.e., for each class placement). It then predicts that the point belongs to the class with the largest p-value, with a confidence equal to the complement of the second p-value. The algorithm for TCM-KNN is shown in Figure 4.

Let k as the number of nearest neighbors to be used; m as the number of training points; c as the classes; r as the points to be classified

for i = 1 to m do

calculateD_i^y , D_i^{-y} and store

end for

calculate α for all training points and store

for i = 1 to r

do Calculate the dist vector as the distances of the new point from all training points

for j = 1 to c do

for every training point t classified as j do

if j D_{tk}>dist(t)

recalculate the alpha value of point t

end

for every training point t classified as non- j do

if j D_{tk} − >dist(t)

recalculate the alpha value of point t

end

Calculate alpha value for the new point classified as j

Calculate p-value for the new point classified as j

end

Calculate alpha value for the new point classified as j

Calculate p-value for the new point classified as j

end

Figure 4: The TCM-kNN algorithm

4.5 Modified TCM-KNN algorithm alongside honeypot

In standard TCM-KNN, we are always sure that the point we are examining belongs to one of the classes. However, in DDoS detection, we don't need to assign a network packets to a certain class, we only need to decide whether the network packet is safe or unsafe. Therefore a modified definition of α as has been used-

$$a_{iy} = \left(\sum_{j=1}^{ik} D_{ij}^{y} \right)$$

This new definition will make the strangeness value of a point far away from the class considerably larger than the strangeness of points already inside the class. This definition has been firstly employed by authors in [10] as a measure of isolation and adopted by authors in [12] to detect outliers. In general classification cases, using the α values, we can compute a series of p-values for the new point for the classes y = {1,2,3,...,c} . We call the highest p-value in this series max p . This provides a way of testing the fitness of point γ for each class y with a confidence of at least δ = 1−τ . Selecting a confidence level δ (usually 95%), we can test if p ≤ τ, in which case, we can declare the point an anomaly. Otherwise, we declare it's normal. Specifically for our anomaly detection task, there are no classes available, the above test can be administered to the data as a whole (they all belong to one class, i.e., the normal class). Doing that, of course, requires a single α_i per point (as opposed to computing one per class), and the τ used directly reflects the confidence level δ that is required. Also, max p is just the p-value of point i to be diagnosed computed using all the normal training data.

The process of the new simplified TCM-KNN algorithm for anomaly detection is depicted in Figure 5:

Parameters: k (the nearest neighbors to be used), m (size of training dataset),

τ (preset threshold), r (instance to be determined)

for i = 1 to m

    calculate strangeness α according to equation for each one in training dataset and store;

end

calculate the strangeness for r according to equation ;

calculate the p-values for r according to equation ;

if ($p \leq \tau$)

    determine r as anomaly with confidence $1-\tau$ and return;

else

    claim r is normal with confidence $1-\tau$ and return;

Figure 5: Modified TCM-KNN algorithm for DDoS Detection

The framework includes two phases: training phase and detection phase. In the training phase, the honeypot will be used to collect network behaviors. This data will mainly be attack data and then we will collect normal network behaviors and classify them into two classes. Then comes the detection phase where all the real-time data collected from the networkwould be directed to the detection engine based on TCMKNN, benign or malicious traffic would be determined.

4.6 Simulation Platform

In order to emulate different devices I decided to opt for COOJA, a Contiki Operating System emulator

4.6.1 Contiki OS

Contiki is an open source operating system for sensor network developed at the Swedish Institute of Computer Science since 2004. Among the available network simulation tools, Contiki operating system holds powerful simulating and communication methodology for the IoT microcontrollers, named ‘motes’. Contiki runs as a virtual machine over an operating system handled by VMware player. So, it is highly portable and efficient for code backing up [4]. To keep the memory overhead down in the resource limited devices, event-driven programming is applied in the operating system [5].

4.6.2 COOJA

COOJA is a Contiki network emulator. It stands out from other emulators by allowing cross-level simulation in the WSN. It enables simultaneous simulation from low level regarding that for sensor node hardware to high level regarding that for node behavior. With this simulation environment, developers can see their applications run in large-scale networks and also tune the emulated hardware in extreme detail [3].

4.7 Data Normalization

Data normalization is the process of intercepting and storing incoming data so it exists in one form only. This eliminates redundant data and protects the data's integrity. The stored, normalized data is protected while any appearance of the data elsewhere is only making a reference to the data that is being stored and protected in the data normalizer.
The normalizer's job is to patch up the incoming data stream to eliminate the risk of evasion as well as ambiguities. The monitor then views the data in its pure, protected and normalized form. Varying forms of normalization exist on levels of increasing complexity. The complexity is due to the set of requirements that must be met to achieve normalization. The most basic is known as First Normal Form, which is often abbreviated 1NF. It is followed by Second Normal Form, or 2NF, Third Normal Form, or 3NF and can continue increasing in forms and complexity as required or desired

As a step of data preprocessing, attribute normalization is essential to detection performance. However, many anomaly detection methods do not normalize attributes before training and detection. Few methods consider normalizing the attributes but the question of which normalization method is more effective still remains.

Normalization benefits:

Normalization plays a key role in the security of a network, provided that normalization extends to every protocol layer. One of the major benefits is the forced integrity of the data as data normalization process tends to enhance the overall cleanliness and structure of the data. Normalization significantly contributes to the fortification of a network, especially in light of typical networks' three main weak points: traffic handling, inspection and detection.
The normalization process for the raw inputs has great effect on preparing the data to be suitable for the training. There are many types of data normalization. It can be used to scale the data in the same range of values for each input feature in order to minimize bias. Data normalization can also speed up training time by starting the training process for each feature within the same scale.[23]

Different attribute normalization techniques:

1) Z-score normalization: Thistechnique uses the mean and standard deviation for each feature across a set of training data to normalize each input feature vector. The mean and standard deviation are computed for each feature. The transformation is given in the equation

$$x' = \frac{(x_i - \mu_i)}{\sigma_i}$$

This produces data where each feature has a zero mean and a unit variance. Normalization technique is applied to all the feature vectors in the data set first; creating a new training set and then training is commenced. Once the means and standard deviations are computed for each feature over a set of training data, they must be retained and used as weights in the final system design. It is a preprocessing layer in the neural network structure.

2) This Min-Max Normalization: This method rescales the features or outputs from one range of values to a new range of values. More often, the features are rescaled to lie within a range of 0 to 1 or from -1 to 1. The rescaling is often accomplished by using a linear interpretation formula such

$$\boldsymbol{x' = (x_{max} - x_{min}) \times \frac{(x_i - x_{min})}{(x_{max} - x_{min})} + x_{min}}$$

when (x_{max}- x_{min})=0 for a feature, it indicates a constant value for that feature in the data. When a feature value is found in the data with a constant value, it should be removed because it does not provide any information to the neural network. When the min-max normalization is applied, each feature will lie within the new range of values will remain the same. Min-max normalization has the advantage of preserving exactly all relationships in the data.

3) Median normalization: Themedian method normalizes each sample by the median of raw inputs for all the inputs in the sample. It is a useful normalization to use when there is a need to compute the ratio between two hybridized samples. Median is not influenced by the magnitude of extreme deviations. It can be more useful when performing the distribution.

$$x' = \frac{x_i}{median(a_i)}$$

4) Sigmoid Normalization: The sigmoid normalization function is used to scale the samples in the range of 0 and 1 or -1 to +1. There are several types of non-linear sigmoid functions available. Out of that, tan sigmoid function is a good choice to speed up the normalization process. If the parameters to be estimated from noisy data the sigmoid normalization, method is used.

$$x' = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

5) Mean and Standard Deviation normalization: Another approach for scaling network inputs and targets is to normalize the mean and standard deviation of the training set. This function normalizes the inputs and targets so that they will have zero mean and unity standard deviation. The normalized inputs and targets are returned will have zero means and unity standard deviation.When this method is used to preprocess the training set data, then the trained network is used with new inputs, and the new inputs can be preprocess with the means and standard deviations that were computed for the training set using original data set. It can be calculated

$$y' = (x_i - x_{mean}) \times \frac{y_{std}}{x_{std}} + y_{mean}$$

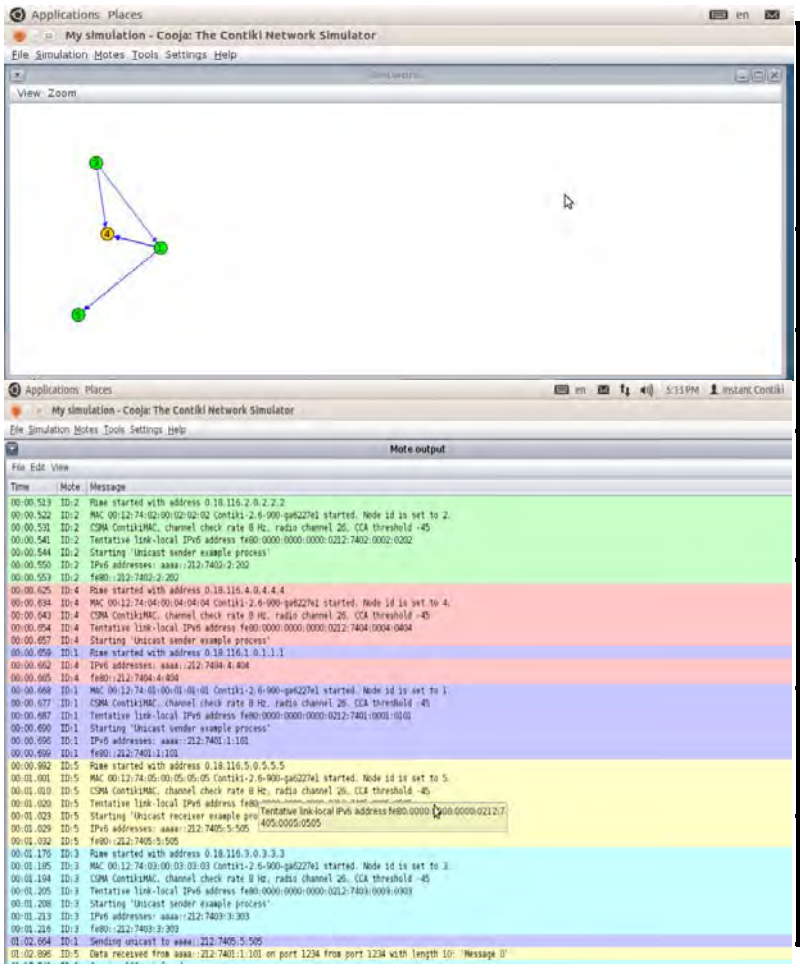
6) Statistical Column Normalization: The statistical column normalization method normalizes each sample with a column normalization value. Calculate the normalization of each column by normalizing the columns to a length of one. Calculate each sample by dividing the normalized column attribute and multiplied by a small bias value.

$$x' = \frac{x_i - n(c_a)}{n(c_a)} \times 0.1$$

CHAPTER 5: Results, Discussion and Conclusion

5.1 Results and Discussion

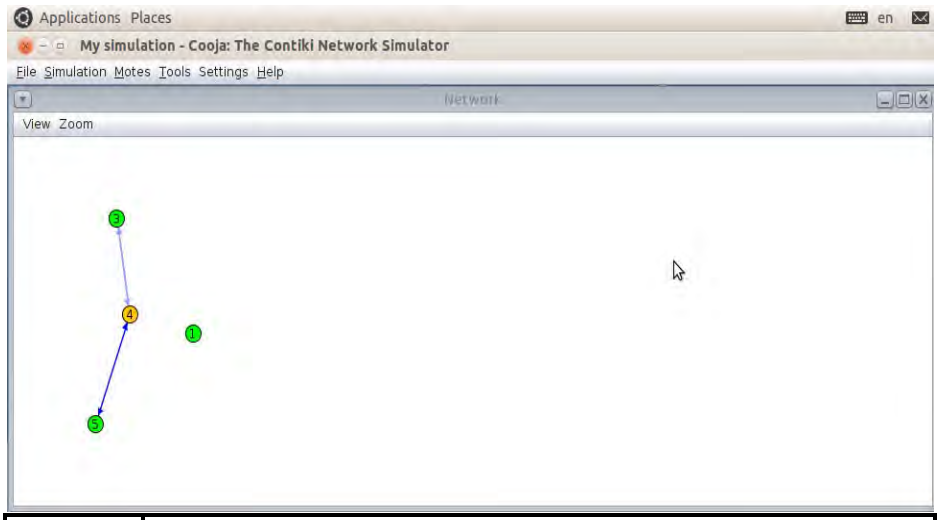
To test the effectiveness of the proposed algorithm, several IoT network scenarios can be constructed in COOJA. To demonstrate and clarify the effect of the proposed algorithm, interactions between motescan be tested with and without the algorithm. The purpose of this is to examine whether the endmote is able to classify and reject the malicious service request.Figure-6 shows the situation happened without the proposed system.



Time (s)	Mote output	
	Mote ID	Message
0.517	2.0	Starting‘Attackerrequest’
0.663	1.0	Starting‘Serve request’
1.786	1.0	Request receivedfrom2.0:‘You are under attack!’
1.888	2.0	Attackerrequest has beenserved byworker 1.0
3.411	1.0	Request receivedfrom2.0:‘You are under attack!’
3.515	2.0	Attackerrequest has beenserved byworker 1.0

Figure 6: Network simulation without the proposed system

However, in the proposed system, the motes will be able to classify the malicious requests and reject them. Figure 7 shows the possible network simulation in the proposed system.



Time	Mote output	
	Mote	**Message**
0.517	5.0	Starting‘Attackerrequest’
0.663	1.0	Starting‘Serve request’
1.180	3.0	Starting‘Attackerrequest’
1.785	4.0	Request receivedfrom5.0: Classify as benign. Request served
1.889	5.0	request has beenserved
3.283	4.0	Request receivedfrom3.0: Classify as benign. Request served
3.428	3.0	request has beenserved
3.660	4.0	Request receivedfrom1.0:classify as anomaly.Request rejected.
3.762	5.0	requestnotserved.
5.658	4.0	Request receivedfrom3.0: Classify as benign. Request served

Figure 7:Possible network simulation in the proposed system

Furthermore, this is a pattern classification system and for more effective detection the classification methods can correspond to the attributes. From the different attribute normalization techniques I have already discussed in section 4.7, many have been employed for anomaly detection. Among them,statistical normalization not only considers the mean scale of attribute values, but also takes into account their statistical distribution and this may help a lot for the detection. In general, for the detection with distance based methods such as TCM-kNN, statistical normalization is the best choice. Figure-8 shows the accuracy rate [21] of different normalization techniques and compares them-

Normalization Method	Accuracy
Z-Score	69.45
Min-Max	69.67
Median	64.31
Sigmoid	69.28
Mean and Standard Deviation	64.44
Statistical Column	72.55

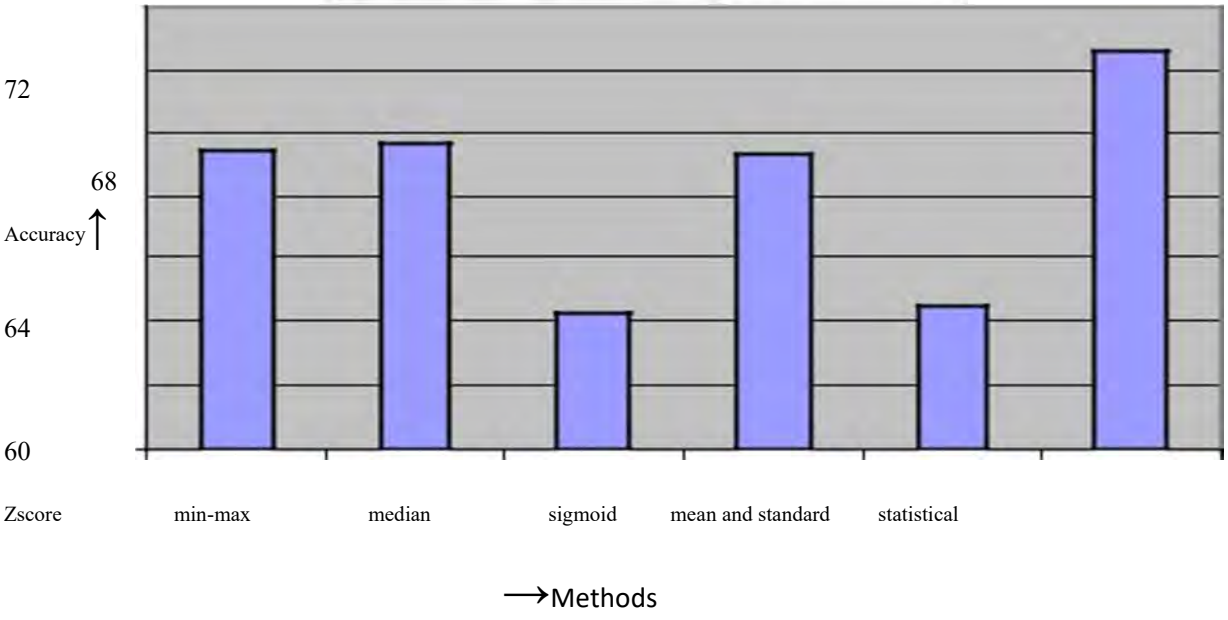


Figure 8: Comparison of Accuracy

Based on these results, statistical normalization should be used with TCM-kNN for DDoS attack detection because statistical attribute normalization can improve a lot the detection results. Also TCM-kNN can achieve good results with statistical attribute normalization since it works well with distance based method.

5.2 Conclusion

In this paper a system has been proposed where honeypot will be used to collect attack data, which will be used in the training phase for the modified TCM-kNN algorithm so that it can classify harmful request. The proposed defending algorithm can effectively help an IoT network to distinguish malicious request from legitimate ones and process them differently. Also, it has been discussed that statisticalnormalization technique should be implemented. Furthermore, statistical normalization is the better choice if the data sample is large.Therefore it is suggested suggest that attribute normalization should always be considered for the classification problem. Although, the computation complexity of *k*-NN for the detection is *Big O*. It is clear that TCM-*k*NN needs a lot of computation if the data is very high-dimensional and the amount of training samples is very large, however,TCM-*k*NN is also light-weight so that it is feasible to periodically retrain the detection model only by incorporating new training data. For future work, optimal and better suited feature selection and instance selection method can be implemented on the training data for more efficient result. They are two important data processing steps in data mining, where the former is aimed at removing some irrelevant and/or redundant features from a given dataset and the latter at discarding faulty data.

Reference:

1. Gammerman, A., and Vovk, V. (2002) Prediction algorithms and confidence measures based on algorithmic randomness theory. Theoretical Computer Science. 287: 209-217.

2. B. Daniel, D. Carlotta, and P. R. James. Detecting outliers using transduction and statistical testing. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, USA, 2006, 55-64.

3. Osterlind, F., A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. "Cross-Level Sensor Network Simulation with COOJA." In Proceedings 2006 31st IEEE Conference on Local Computer Networks, 641–48, 2006. doi:10.1109/LCN.2006.322172.

4. Dunkels, Adam, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. "Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems." In Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, 29–42. SenSys '06. New York, NY, USA: ACM, 2006. doi:10.1145/1182807.1182811.

5. Dunkels, A., B. Gronvall, and T. Voigt. "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors." In 29th Annual IEEE International Conference on Local Computer Networks, 2004, 455–62, 2004. doi:10.1109/LCN.2004.38.

6. Proedru, K., Nouretdinov, I., Vovk, V., Gammerman, A. (2002) Transductive confidence machine for pattern recognition. Proc. 13th European conference on Machine Learning. 2430:381-390.

7. Ho, S.S., and Wechsler, H. (2003) Transductive Confidence Machine for Active Learning, Int. Joint Conf. on Neural Networks, Portland, OR.

8. Vapnik, V. (1998) Statistical Learning Theory, New York: Wiley.

9. Li, M., and Vitanyi, P. (1997) Introduction to Kolmogorov Complexity and its Applications. 2nd Edition, Springer Verlag

10. Vovk, V., Gammerman, A., and Saunders, C. (1999) Machine learning applications of algorithmic randomness. Proceedings of the 16th Intl. Conference on Machine Learning. 444-453.

11. Proedru, K., Nouretdinov, I., Vovk, V., Gammerman, A. (2002) Transductive confidence machine for pattern recognition. Proc. 13th European conference on Machine Learning. 2430:381-390.

12. Peter, E. & Schiller, T. (2008, April 15). *A practical guide to honeypots*. Retrieved from http://www.cs.wustl.edu/~jain/cse571-09/ftp/honey/

13. Weiler, N. (2002).*HoneypotsforDistributedDenialofServiceAttacks*

14. Mirkovic, Jelena, and Peter Reiher. "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms." SIGCOMM Comput. Commun. Rev. 34, no. 2 (April 2004): 39–53. doi:10.1145/997150.997156.

15. Misra, Sudip, P. Venkata Krishna, HarshitAgarwal, AntrikshSaxena, and Mohammad S. Obaidat. "A Learning Automata Based Solution for Preventing Distributed Denial of Service in Internet of Things." In International Conferences on Internet of Things, and Cyber, Physical and Social Computing, 114–22. IEEE, 2011.

16. Han, Chong, JosepMiquelJornet, EtimadFadel, and Ian F. Akyildiz. "A Cross-Layer Communication Module for the Internet of Things." Computer Networks 57, no. 3 (February 26, 2013): 622–33. doi:10.1016/j.comnet.2012.10.003.

17. Cluley, Graham. "These 60 Dumb Passwords Can Hijack over 500,000 IoT Devices into the Mirai Botnet." Graham Cluley. N.p., 10 Oct. 2016. Web. 14 Dec, 2016.

18. Dishon, Robin. "DDoS Attacks Explained." DDoS Attacks Explained. ESET, 21 Oct2016. Web. 13 Dec, 2016.

19. Wei, Jialu. "DDoS on Internet of Things – a big alarm for the future". 14 Dec, 2016.

20. Arshad S., Abbaspour M., Kharrazi M. and Sanatkar H., 2011. An Anomaly-based Botnet Detection Approach for Identifying Stealthy Botnets. International Conference on Computer Applications and Industrial Electronics 2011. IEEE, pp. 564 – 569.

21. Jayalakshm, T. ,Santhakumaran, A. (2011) “Statistical Normalization and Back Propagation for Classification”. International Journal of Computer Theory and Engineering, Vol.3, No.1, February, 2011 1793-8201.

22. Sanjaya K. Panda, Subhrajit Nag and Prasanta K. Jana, “A Smoothing Based Task Scheduling Algorithm for Heterogeneous Multi-Cloud Environment”, 3rd IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC), IEEE, Waknaghat, 11th - 13th Dec 2014.

23. Suprina, D.(January 7, 2013).*The importance of data normalization in IPS*. Retrieved from https://www.helpnetsecurity.com/2013/01/07/the-importance-of-data-normalization-in-ips/