

# Smart City DDoS Attacks Maneuver Based on Identification, Heuristics and Load Balancers

Elie Y. Nasr, Isam Shahrour, Farid E. Nakhle, Daniel R. Sakr, Liliane Y. Karam

**Abstract**— Smart cities are highly digitized cities by nature. They are characterized by large volumes of data stored digitally and large numbers of physical objects and IoT devices with online connections to the Internet. As the use of connected IoT devices constantly increases, so do the security concerns. The biggest challenge that we face nowadays is the hackers' use of IoT devices (camera, DVR, thermostat, etc.) to launch high-profile Distributed Denial of Service (DDoS) attacks as it happened lately on September 2016 with the Mirai botnet [1,2] which generated up to 1.2Tbps of severe wave of network traffic. This kind of botnets not only affected the IoT devices themselves, but also every device connected to the Internet. As such, availability attacks have the greatest destructive effects and are considered the main cause for a smart city blackout. The purpose of this paper is to propose a defensive or self-protective framework used to lessen the impact of zero-day distributed denial of service attacks on the smart city network (IoT devices) and hence to avoid entire blackout. This framework mitigates the zero-day availability attacks based on Identification, Heuristics and Load Balancer in a reasonable time frame. Simulation results obtained after testing the solution showed that the heuristic approach associated with the backup load balancer led to substantial accuracy in mitigating DDoS attacks.

**Index Terms**— Denial of Service, Distributed Denial of Service, Heuristics, Load Balancers, Smart City, Smart City Security, Zero-day attack.

## 1 INTRODUCTION

Regardless of what are the malicious urges of hackers in intruding smart city systems, Denial of Service attacks or availability attacks are considered the most vicious and destructive as they might bring down or paralyze the overall city's network services. The increasing number of attacks and the effects of these happenings show the importance of researching such types of attacks. These attacks have led businesses down, destroyed the economy of a nation and even led to governments being changed. DoS attacks are usually classified into several categories [3] and this depends on the style with which they were implemented: 1) Distributed Denial of Service; 2) Low rate TCP targeted Denial of Service; 3) Reflective Denial of service. These categories encompass DoS attacks like: Smurf, Ping Flood and Ping of Death, TCP SYN Flood, and UDP Flood.

Smart city dimensions rely heavily on Internet of Things (IoT) devices. Thus, exploiting effectively IoT devices vulnerabilities by botnets will definitely lead to a wide range of destructive distributed denial of service attacks.

- Elie Y. Nasr, Ph.D., Department of Information and Communications Technology, American University of Science and Technology, Lebanon. Email: enasr@aust.edu.lb
- Isam Shahrour, Ph.D., Laboratoire Génie Civil et géo-Environnement Lille1/Polytech'Lille University, France . E-mail: isam.shahrour@univ-lille1.fr
- Farid E. Nakhle, Department of Computer Science, American University of Science and Technology, Lebanon. Email: farid.nakhle@live.com
- Daniel R. Sakr, Laboratoire Génie Civil et géo-Environnement Lille1/Polytech'Lille University, France . E-mail: danielsakr87@gmail.com
- Liliane Y. Karam, Department of Information and Communications Technology, American University of Science and Technology, Lebanon. Email: likaram@aust.edu.lb

Unlike conventional devices, IoT devices are exposed to three different types of availability attacks: 1) hardware availability, 2) network availability and 3) cloud availability. Consequently, currently adopted DDoS mitigation techniques against IoT attack vectors for availability have proven inefficient as considerable discrepancies and inconsistency of IoT devices have become apparent. For example, an IoT device can be forced to use more power as a result of exchanging multiple keys over ZigBee smart connection and thereby causing battery drain. Likewise, IoT devices are susceptible to radio jamming due to the use of radio networking (Wifi, Zigbee, Bluetooth, 3G)[4].

The biggest challenge that we face nowadays is the use of IoT devices (camera, DVR, thermostat, etc.) to launch a high-profile Distributed Denial of Service attacks as it lately happened with the Mirai botnet. Recent attacks showed that IoT devices vulnerabilities were most efficiently used by botnets to unveiling a variety of Distributed Denial of Service (DDoS) attacks. This kind of botnet not only affects the IoT devices, but also everybody connected to the Internet. To reveal the severity of the DDoS attacks caused by IoT botnets, we hereby list some recent incidents that took place late 2016.

On September 30, 2016, the blog of the security researcher Brain Kerbs (kerbsOnSecurity.com) started experiencing a 632 Gbps DDoS attack from an IoT botnet. Akamai, the leading Content Delivery Network (CDN) operator which provided the DDoS attack mitigation, was accused after three days by its customers for the scarcity of their internet bandwidth as Akami platform could not handle anymore the technical means required to avoid the attack. [5, 6]

On September 22, 2016, the French cloud computing company (OVH) experienced a series of DDoS attacks with the severest single attack reaching up to 799 Gbps. Later on September 23rd, it experienced another DDoS generating up to 1.5Tbps coming from around 146000 cameras, printers and

DVRs. The IoT devices were sending from 1 to 30 Mbps of traffic each [7].

On October 21st, Dyn the leading managed DNS provider and Internet Performance Management Company, experienced on its DNS servers several waves of DDoS attacks from 100000 internet connected IoT devices with the severest attack generating 1.2Tbps of traffic. [8]

Regardless of the huge amount of damages brought about from the above three attacks, our challenge is to look forward and find a solution to mitigate such attacks by protecting not only the IoT devices, but the overall network infrastructure. In the three previous attacks all IoT internet connected devices were found mainly infected by Mirai malware.

The amount of networked IoT device will certainly increase in years to come. Currently there are up to 15 billion internet connected IoT devices in use and it is predicted to become 200 billion in 2020. We expect to see in the future more damaging IoT DDoS attacks because the source code of the Mirai malware is freely available on the Internet [1]. The Mirai source code has been published on a community hack forum as open-source by its suspected author Paras Jha using the moniker online name "Anna-senpai" [2].

In just only two weeks after the release of Mirai source code, the number of infected devices has become more than double as it has increased drastically from 213000 to 483000. After examining the IP addresses of the infected IoT devices, one could notice that the source of the attack of the botnet is distributed over 164 counties. High profile attack densities were found in USA, China, Brazil, and Vietnam. Mirai malware generates not only conventional HTTP, TCP, or UDP traffic, but also exploits legitimate protocols like GRE IP (used for peer to peer VPN), GRE ETH, DNS, STOMP to flood against a specific target during a DDoS attack [9,10].

Mirai is designed to exploit default and hardcoded credentials and self-propagates by scanning the Internet. It utilizes a dictionary of generic and device-specific default credentials to infect IoT devices. Also, non-secure by default ports due to Universal Plug and Play (UPnP) were considered as easy target for the Mirai botnet. Several devices like those were made by the Chinese company XiongMai Technologies, can be accessed through a web interface - (IP/Login.htm) - navigating to "DVR.htm" without device credentials and prior to login [11].

Flashpoint security firm estimated over 515,000 vulnerable devices with hardcoded credentials were found actively in use on early October 2016.

The following sections describe in details the problem statement, the challenges, our contributions, the solution methodology and results.

## 2 PROBLEM STATEMENT

Conventional mitigation methods against DDoS attacks which depend deeply on Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), Network Access Controls (NAC) and firewalls functionalities have proven deficiency, and the attack examples provided above recognize this fact.

Preventive measures from firewalls (use of Access Control Lists, use of rate limiting, destination based black hole filtering, dropped packets, port errors, dictionary logon attacks etc.), IDSs (buffer overflow attacks, fragmentation and replay attacks, cached content change, file integrity check etc.), Secure Socket Layer (certificate errors, DoS attacks, session drop) can jointly be overwhelmed when encountering severe waves of DDoS attacks each generating up to 1.2Tbps of traffic as experienced by the Mira malware.

System administrators should not rely totally on the aforementioned protection mechanisms as they could not: a) compensate for weak identification and authentication mechanisms; b) conduct investigations of attacks without human intervention; c) perceive the contents of your organizational security policy; d) compensate for weaknesses in network protocols; e) analyze all of the traffic on a busy network; f) deal always with some problems involving packet-level attacks. Yet, no perfect mitigation for this problem exists [12, 13, 14].

## 3 CHALLENGES AND CONTRIBUTIONS

Our main challenging task in this study is to find an efficient, defensive and innovative solution that might lessen the impact of distributed denial of service attacks on the smart city network (IoT devices) to avoid entire blackout.

To date, a considerable amount of research has been made to try and mitigate DOS and DDOS attacks. Most solutions consist of scaling up the infrastructure, or trace the source of the attack to finally block it. Our research seeks to enhance this process by providing a mechanism of blocking or isolating the source of the attack, not only in an automated, smart procedure, but also by using heuristics in an effective distributed way using load balancers to isolate the attackers in a virtually real time manner. Hence our research had the scalability and the isolation combined to eventually mitigate the attack effectively.

The defensive mechanism that we proposed for DDoS protection relies on two load balancers in which the first one uses a heuristic approach, and the second acts as backup to produce a solution in a reasonable time frame. In brief, traffic is received by the main load balancer which, based on reputation and heuristics, sends the user via the new visitor route or the trusted visitor route. In case of a DOS attack, the main load balancer will be flooded / gone offline. Hence the backup load balancer will fill the void. The backup load balancer does not have a route for new users, and thus, blocks all new requests (hence blocks the attack) and safely allow trusted users to enter the network. (See details in section proposed solution)

Two very important questions arise here. Why are we adopting the heuristic approach? And why are we using load balancer to stop DoS or DDOS?

## 4 WHY ADOPTING THE HEURISTIC APPROACH?

Wikipedia defines heuristic as technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail

to find any exact solution [15]. Usually heuristic algorithms are used for problems that have low time complexity and cannot be easily solved [16]. Due to the lack of a unified algorithm (P class) or method to solve the problem of multifaceted DDoS attacks, the problem falls into the NP class. The variety of the DDoS attack unexpected-parameters makes the problem non-deterministic [17]. Class NP consists of all those problems whose solution can be found in polynomial time on a non-deterministic Turing machine. Yet, such machine does not exist. A zero-day attack is a threat aimed at exploiting software or hardware vulnerability before the vendor becomes aware of it and before it becomes known to security experts. These attacks are among the hardest to mitigate, and thus, the use of heuristic technique becomes a must.

## 5 WHY USING LOAD BALANCERS TO STOP DDoS ATTACKS?

Using a firewall is simply out of the equation as it requires simple pre-defined rules to function. Based on these rules it will take the decision of letting the traffic go or block it.

So the remaining equation is IPS vs Load Balancers. We choose load balancers to specifically stop DOS attacks because we take advantage of their smart way to manage traffic. A DOS attack does not always have a malicious pattern. It could simply be a normal request, repeated so many times, really fast. Hence, an IPS could let it pass. On the other hand, the IPS does not know how to manage traffic in an effective way. While load balancers have memory queues that save requests and split them over other peered load balancers or target servers. This is an advantage for load balancers as they can withhold and survive a huge amount of requests while the IPS might be subject to DOS attack itself and fail quickly.

The main advantage of load balancers is, not only they can be peered together, but they are also able to process requests, change their routes based on a certain condition, and even simply drop them if we wanted. Peered load balancers are also able to forward request to other load balancers that have different conditions and different routes to different servers. Taking advantage of all these features, load balancers will allow us to collect and analyze as many requests as possible, and allow them all to pass into our network.

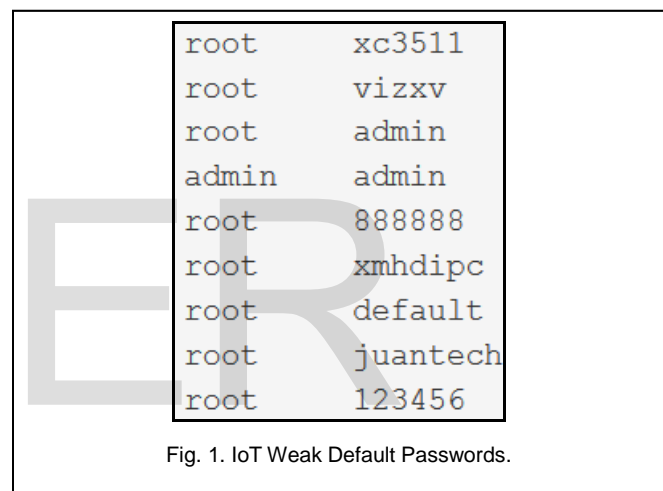
In case of an attack, the gateway load balancer will fail and hence redirect all traffic to its peered load balancer that have a set of conditions in order to process the requests or drop them. This way, the second load balancer becomes the gateway, and it will only allow requests from trusted users and therefore surviving the attack by dropping all other untrusted requests.

## 6 UNDERSTANDING THE BOTNET STRUCTURE

The term botnet originates from the terms “roBOT NETwork”. A bot is a malicious-software installed into a PC or a device that runs automated attacks over a network. The infected device itself is known as a bot or zombie. A botnet is a network of malware-infected devices (zombies), which are controlled by cybercriminals. Intended attackers search for vulnerable internet connected devices, infect them with spam and mal-

ware to launch distributed denial-of-service attacks. Up until recently, botnets were formed of contaminated PCs and laptops. What made recent attacks (kerbsOnSecurity.com, OVH, Dyn) different is that the botnets were embraced of what has become recognized as the “Internet of Things” devices.

The infected IoT devices or bots (DVRs, Webcams, Baby monitors, electronic thermostat etc.) are used to perform command based DDoS attacks. Bots communicate with each other with adversaries through Command Control (C2) Servers to either flood various types of traffic or change their IP addresses for different intervals. The Scanning tools are used to scan IoT devices for vulnerabilities. For instance, in the case of Mirai botnets bots were found communicating with the C2 server to scan across TCP port 23 and port 2323 and use different brute force technique for guessing passwords and store them in the reporting server. The attacker uses these guesses through the C2 server to plague the IoT devices with the malicious code (malware).



root	xc3511
root	vizxv
root	admin
admin	admin
root	888888
root	xmhdipc
root	default
root	juantech
root	123456

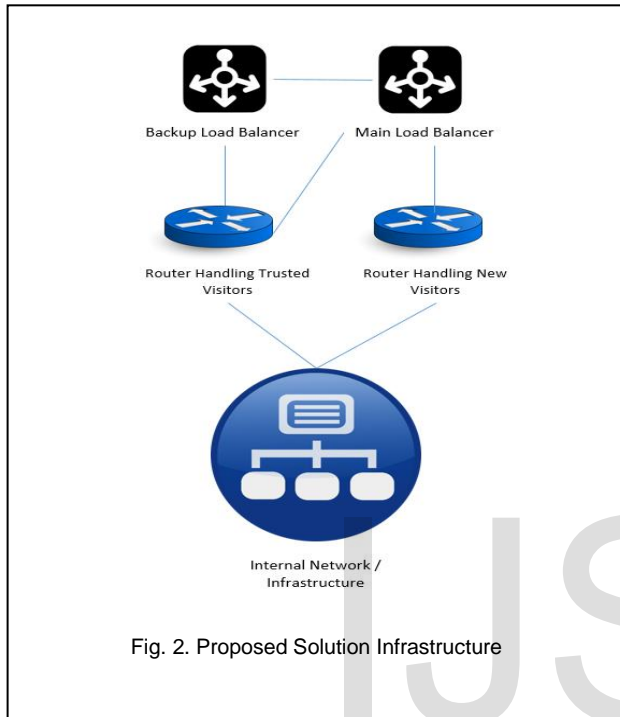
Fig. 1. IoT Weak Default Passwords.

Figure 1 above shows a sample of weak default passwords grabbed as a result of Mirai’s scanning vulnerable IoT devices. At this moment, the loader queries these credentials to log on to vulnerable IoT devices and order them to download, execute the malware and launch the attack. Some IoT devices have no permanent storage media; the malware stays resident into their volatile memory (RAM) and hold onto launching attacks as long as they are not being reset.

## 7 PROPOSED SOLUTION

Our solution consists of keeping track of our visitor’s identification (such as IP), and separate our regular visitors from new ones. Why does IoT device IP such as electronic thermostat, printer, or webcam request Tweeter or Facebook page access? This could be achieved by adding a smart router / load balancer that distribute data depending on the visitor’s reputation as illustrated in figure 2 and figure 3 respectively. Traffic is received by the main load balancer which, based on reputation and heuristics, sends the user via the new visitor route or the trusted visitor route. In case of a DOS attack, the Main Load Balancer will be flooded and gone offline. Hence the

backup load balancer will fill the void. The backup load balancer does not have a route for new users, and thus, blocks all new requests (hence blocks the attack) and safely allow trusted users to enter the network. This solution saves services from going offline, and increase clients trust by staying available regarding the attack [18]. However, the downside is that during the attack phase, some legitimate new visitors will be left out as well since they are not in the trust zone yet.



Our solution consists of two main parts: the physical infrastructure of the network, and the algorithm identifying legitimate trusted traffic from the poisoned / infected traffic. For the infrastructure, we use two Ubuntu servers. Each of them is running HAPROXY to serve as a load balancer. Having two (or more) load balancers is a critical step within this solution, as they are configured in a failover manner (load balancer peering), where the main load balancer is always functioning until an attack occurs (see configuration in figure 4). Once an attack occurs, naturally, the first load balancer will be flooded with requests and hence the backup load balancer will take place. The trick is that the second load balancer only redirects traffic to servers that process only trusted requests and ignore all the others [19], hence it will hardly be processing any dummy requests and thus remain functioning normally.

```
Stream LoadBalancer {
    server normalloadbalancer.example.com;
    server 192.168.43.40 backuploadbalancer sum_of_weights > metric;
}
```

Fig. 4. Load Balancers Configuration

The second part of the solution is our algorithm to classify traffic as trusted or untrusted. For that, we test the request by having it pass through a set of rules, where each rule has its own weight. For each rule the request is successfully tested, the request gains the weight of this rule in an incremental manner. Once it is done, if the request has a specific total number of points, the request is then trusted. For instance, we define a set of rules as shown in table 1 below:

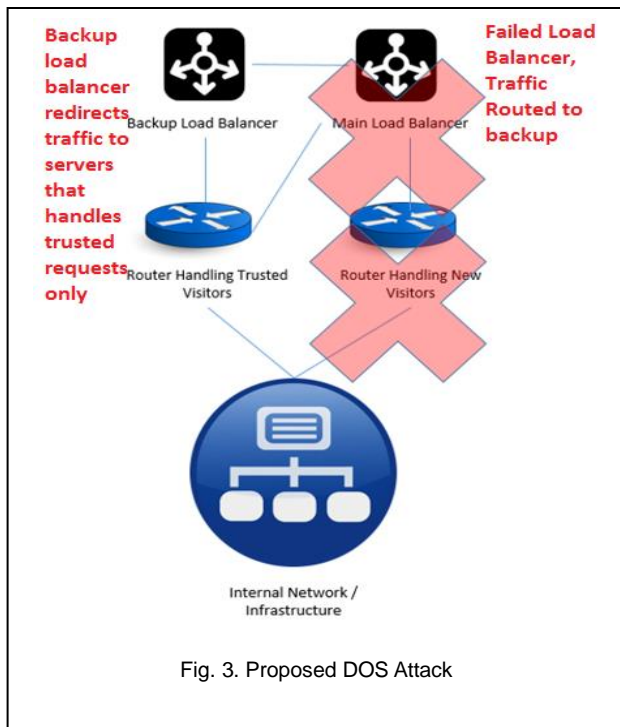


TABLE 1  
LOAD BALANCER RULES

RULE	The visitor's IP	The visitor's user agent	The visitor's geo-location	The visitor's browser's fingerprint	The visitor's credentials (If logged in)	The service requested
WEIGHT	W1	W2	W3	W4	W5	W(6)

Where: weight  $W(x)$  is a defined integer.

Two approaches can be used to assign weights to rules: The first is based on human supervised training, trials and errors and the second is adaptive. (See Section 8.2 Assigning Weights to Rules).

### 8 METHODOLOGY

- 1- The request is sent from an IP that access our services daily. The request gains  $W1$  point.
- 2- The request's User Agent has previously accessed our services. The request gains  $W2$  point (total is now  $W1 + W2$ ).
- 3- The visitor identified is in the same country our services is, and the same location his IP usually requests from, the request gains  $W3$  (total is now  $W1 + W2 + W3$ ).
- 4- The visitor's browser fingerprint is not identified. The

number of points remains  $W1 + W2 + W3$ .

5- The visitor is not logged in. The number of points remains  $W1 + W2 + W3$ .

As a result, if  $W1 + W2 + W3 > X$ , the request is marked as trusted and thus allowed to pass.

Else, the request is dropped, and further requests from the same source are no longer needed to be tested, they are dropped automatically. Note: the rules and weight are to be customized per service, as each service can provide more evidence and different ways of identifications of legitimate requests.

## 8.1 Configuring Load Balancers

HAProxy (High Availability Proxy) is free and open source TCP/HTTP Load balancer software run under the Linux platform. Our network, consisting of two load balancers: a main load balancer (allowing all traffic to be forwarded to the services) and a backup load balancer (protecting our services from attacks, allowing only requests from trusted sources).

The two load balancers are configured as follows:

```
#####
# SECURITY CUSTOM CONFIG #
#####
# HARD CODED PROOF OF CONCEPT #
#####

frontend data-in
    bind *:80

    acl allow_data hdr(host) -i security.proof
    use_backend security if allow_data

backend security
    balance roundrobin
    mode http
    stick-table type ip size 20k peers failover_balancer
    server server 192.168.43.50:80 check

peers failover_balancer
    peer loadBalancerNormal 192.168.43.30:1024 check
    peer lbb 192.168.43.40:1024 check
```

Fig. 5. Sample main load balancer configuration: Allowing all traffic to be forwarded to our services

```
frontend data-in
    bind *:80

    acl network_allowed src 192.168.43.200 192.168.43.99
    acl allow_data hdr(host) -i security.proof
    http-request deny if !network_allowed
    use_backend security if allow_data

backend security
    balance roundrobin
    mode http
    stick-table type ip size 20k peers failover_balancer
    server server 192.168.43.50:80 check

peers failover_balancer
    peer lbb 192.168.43.40:1024 check
    peer loadBalancerNormal 192.168.43.30:1024 check
```

Fig. 6. Sample backup load balancer configuration: Allowing traffic only from trusted sources.

Figure 5 and 6 display the configuration of both load balancers. We can see that on their input (frontend data-in), both load balancers are bound to port 80 as our test service is web based. Once the load balancers receive a request targeting the endpoint "security.proof", the code under "backend security" will run.

However, we can see that on the backup load balancer (the right side configuration of figure 5), a variable is introduced called "network\_allowed" with the source ("src") parameter, marking 192.168.43.200 as allowed (trusted client). This is the IP of tester 1. In this particular load balancer (backup load balancer), before using the "backend security" the load balancer will deny the request if its source is not in the "network\_allowed" list.

In backend security, we notice that the requests are forwarded to the web server on port 80 holding the IP 192.168.43.50. Naturally, the check parameter keeps knowledge whether or not the server is online and available on the selected port.

Finally, the failover\_balancer defines the peered load balancers. This section helps the traffic to be redirected from the normal load balancer to the backup load balancer when the normal load balancer fails. When a DOS attack occurs, if the normal load balancer fails, the backup load balancer will take over and drop all requests from untrusted sources, allowing only trusted sources to reach the service and hence protecting it from malicious attacks.

## 8.2 Assigning Weights to Rules

Two methods can be used to assign weights to rules:

a) Pre-assigned: This method is based on human supervised training, trials and error. The security administrator has to be fully aware of all network activities, OS configuration and services vulnerabilities.

b) Adaptive: This method is considered more robust in which the weights of the load balancer rules are determined based on polling several vulnerability metrics from the internal networks after conducting vulnerability scanning analysis.

Security metrics are indicators used to provide contextual quantitative measure for the security characteristics of an information system. To improve the security of a system, we have to measure it. The Common Vulnerability Scoring System (CVSS) was designed by a team of security professional and by the National Institute of Standard and Technology (NIST). It is a vulnerability scoring system designed to provide a standardized and open framework for rating software vulnerabilities. The Common Vulnerability and Exposure (CVE) is the industry standard for vulnerabilities and exposure names, namely the CVSS and the Common Weakness and Enumeration (CWE) which provides a list of software weaknesses.

To determine the current and historical vulnerabilities associated with our network services, applications, and operating systems we used Nessus vulnerability scanner. We are interested in medium and high vulnerability scores to raise alert of the target service, to determine the attack paths that potential attackers are trying to exploit and thus, to be able to figure out the weight. Nessus can be fed by a variety of vulnerability

suppliers like CVE, OSVDB, Cert, NVD to calculate the severity vulnerability score of our custom made network. Nessus assigns the following severities to vulnerabilities found based on the CVSS score as shown in table 2 below:

CVSS score	Severity level
0.0 - 3.9	Low
4.0 - 6.9	Medium
7.0 - 10.0	High

Sample output (figure 7) of Nessus Scanning detected non-compliant results.

```
192.168.20.16|unknown (0/tcp)|21156|Security Hole|"Maximum password age" :
[FAILED]\n\nRemote value: 42\nPolicy value: 182\n\n\n
192.168.20.16|unknown (0/tcp)|21156|Security Hole|"Enforce password history" :
[FAILED]\n\nRemote value: 0\nPolicy value: 5\n\n\n
192.168.20.16|unknown (0/tcp)|21156|Security Hole|"Account lockout threshold" :
[FAILED]\n\nRemote value: 0\nPolicy value: 3\n\n\n
192.168.20.16|unknown (0/tcp)|21156|Security Hole|"Account lockout duration" :
[FAILED]\n\nRemote value: 30\nPolicy value: 60\n\n\n
```

Fig. 7. Sample output of Nessus Results

Querying Nessus output using Pandas tool to determine CVE, CVSS and severity level we got the result shown in table 3:  
Hosts = df1.groupby(("Desc','Host').agg ({CVE}, {Severity}),{CVSS})

Vulnerability Description	Host	CVE	Severity	CVSS
Frame injection	192.168.20.12	CVE-2013-1571	Low	3.4
Data replay	192.168.20.13	CVE-2016-5968	Medium	5.0
Bit flipping	192.168.20.14	CVE-2005-0039	Medium	6.4
Vector replay attack	192.168.20.15	CVE-2016-6582	Critical	9.1
session hijacking	192.168.20.16	CVE-2017-6549	Critical	9.3

As a result we correlate vulnerability assessments with load balancer data to determine the weight. The rule weight assigned to the load balancer should be inversely proportional to the CVSS score. Recorded high CVSS scores of services require assigning low weights to rules correlated to those services and thus, the load balancer must not allow the corresponding requests to pass to sever cluster for execution.

## 9 DDoS ATTACK AND RESULTS

### 9.1 Network Setup and Customization

In order to simulate a DDoS attack and test the reliability of our

solution, we made use of the network lab of the Information and Communications Technology department at the American University of Science and Technology.

We customized the network and divided it into two zones (External and Internal), configured the load balancers and integrated them into the network entry points (See figure 8 below).

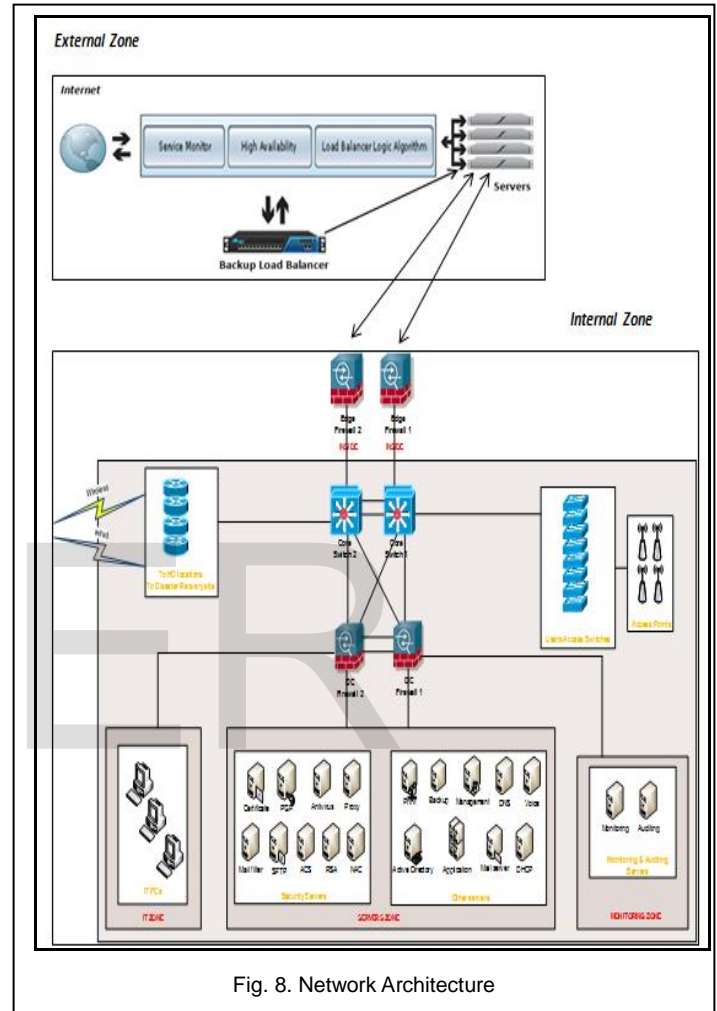


Fig. 8. Network Architecture

### 9.2 External Zone

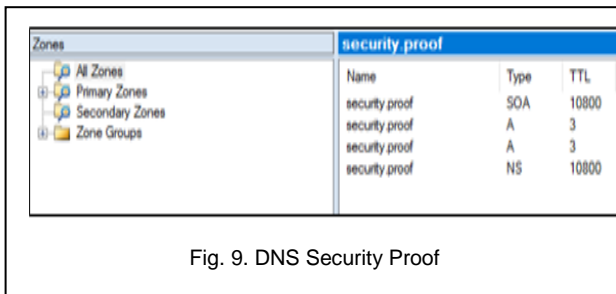
The external zone was configured as follows:

- 1- A main load balancer used to forward all traffic in a timely manner to the webserver. The main load balancer is called Normal Load Balancer (LBN) on our network and it was assigned the IP of: 192.168.43.30
- 2- A backup load balancer was also configured. The backup load balancer is peered with the main load balancer. Its job is to forward only the trusted requests to the server and drop all other foreign requests. It was assigned the IP 192.168.43.40
- 3- A DNS server was added to give the website a domain that the two load balancers will host together, peered one to another for failover. The DNS IP is 192.168.43.99
- 4- A web server considered as the main server providing the service to users. The webserver's IP is 192.168.43.50

5- A client computer, tagged as untrusted (new visitor). The untrusted client IP is 192.168.43.201

6- A client computer, tagged as trusted. The trusted client IP is 192.168.43.200

In addition, two computers were used to perform an attack over used domain assigned in the DNS server as security.proof as shown in Figure 9.



Zones			
security.proof			
Name	Type	TTL	
security.proof	SOA	10800	
security.proof	A	3	
security.proof	A	3	
security.proof	NS	10800	

Fig. 9. DNS Security Proof

### 9.3 Internal Zone

The internal zone was customized as below to mitigate the attack or lessen its impact in case the external zone (our solution) fails to handle it.

- The Edge Firewall internal zone physical interface is connected to the Layer 3 Core switch physical interface.
- Redundant aggregator routers are connected to the core switch. For example, two routers are used for the wired connection and the other two routers are used for the wireless connection. Head office locations and disaster locations are connected to these aggregator routers via an encrypted VPN tunnel connection.
- Physical servers, blade servers, network and security appliances etc. are physically connected to the core switches.
- User access switches or distribution switches are physically connected to the core switches. PCs, printers, faxes and other peripherals are physically connected to these layer 2 switches.
- The wireless access points are connected to the user access switches. The wireless LAN controllers (WLC) are connected to the core switches and are used to monitor the access points.
- An internal firewall or Data Center (DC) firewall is also implemented to segregate the internal corporate zones logically. For example, IT zone, servers zone, monitoring and management zone.

### 9.4 Attack Simulation

In order to instantiate a Denial of Service attack against our network, we made use of the following tools:

1- The open source Low Orbit Ion Cannon (LOIC) platform. LOIC was developed by Praetox Technologies to conduct network stress testing [20], Denial of Service attack as well as Distributed Denial of Service (DDoS) attacks. It has been widely used by Anonymous as DDoS tool since it can generate huge amount of illegitimate TCP, UDP, or HTTP network traffic which causes performance degradation and potentially a service shut down. Over 30000 downloads were recorded during the month of December 2010 when Anonymous organized attacks on the websites of companies and organizations that opposed Wikileaks [21].

2- The bash script Pentmenu developed by pentbox. It is

designed to perform network pen testing functions. It is commonly installed on most linux distributions.

3- The Slowloris DDoS attack software developed by Robert "Rsnake" Hansen. It lets a single computer to take down a web server like IIS, Apache 1.x and 2.x.

Two computers were used to attack the domain (load balancer):

1- A computer running windows and LOIC (Low Orbit Ion Cannon) [<https://sourceforge.net/projects/loic/>] (Attacker 1).

2- Another computer running Kali Linux and using Pentmenu to perform the attack [<https://github.com/GinjaChris/pentmenu>] (Attacker 2).

Two other computers (tester 1 and tester 2) were used to check the status of the service when being attacked. Tester 1 is considered a trusted client / visitor.

The first step was to check if the service endpoint is reachable by attacker 1, attacker 2, client 1 and client 2.

The results were as following:

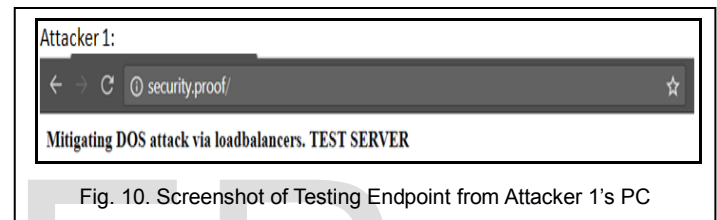


Fig. 10. Screenshot of Testing Endpoint from Attacker 1's PC

Figure 10 shows that the endpoint (<http://security.proof>) is reachable by attacker 1 and all of her/his requests are processed normally before the attack.

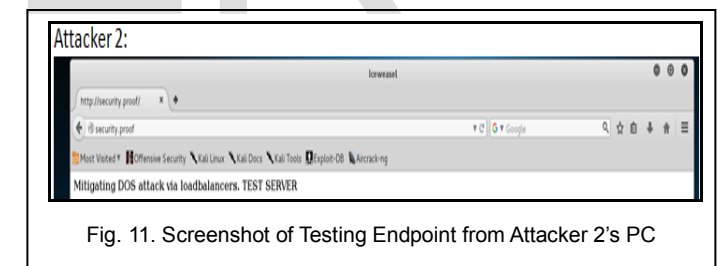


Fig. 11. Screenshot of Testing Endpoint from Attacker 2's PC

Figure 11 illustrates that the endpoint (<http://security.proof>) is reachable by attacker 2 and all of her/his requests are processed normally before the attack.

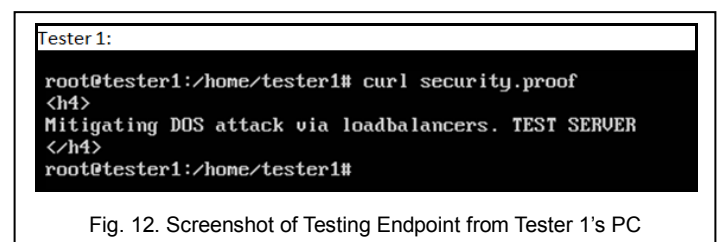


Fig. 12. Screenshot of Testing Endpoint from Tester 1's PC

Figure 12 demonstrates that the endpoint (<http://security.proof>) is reachable by tester 1 and all of her/his requests are processed normally before the attack.

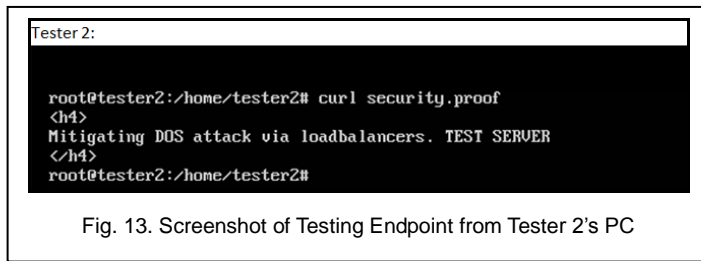


Fig. 13. Screenshot of Testing Endpoint from Tester 2's PC

Figure 13 shows that the endpoint (<http://security.proof>) is reachable by tester 2 and all of her/his requests are processed normally before the attack.

The figures 10, 11, 12 and 13 demonstrate that all of the parties are able to reach and request the server normally without any issues.

Now that the site is reachable by all parties, attacker 1 and attacker 2 start targeting the endpoint.

We have run multiple instances of both attacking software (LOIC and Pentmenu) in order to make the load balancer fail as shown below:

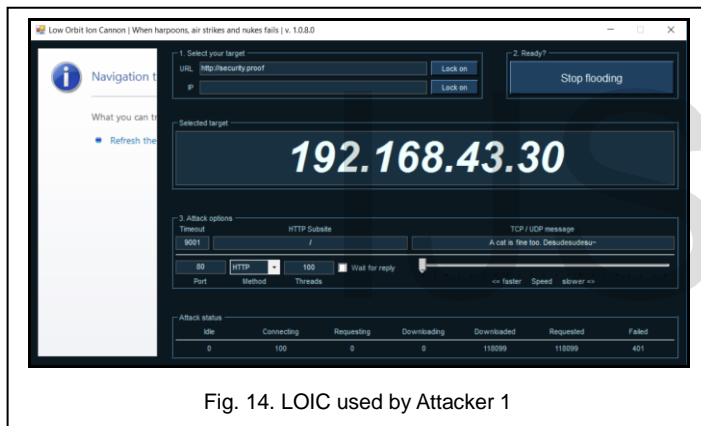


Fig. 14. LOIC used by Attacker 1

Figure 14 is a screenshot of LOIC used by attacker 1, targeting the endpoint (<http://security.proof>) with a massive amount of requests (flooding) as fast as possible.

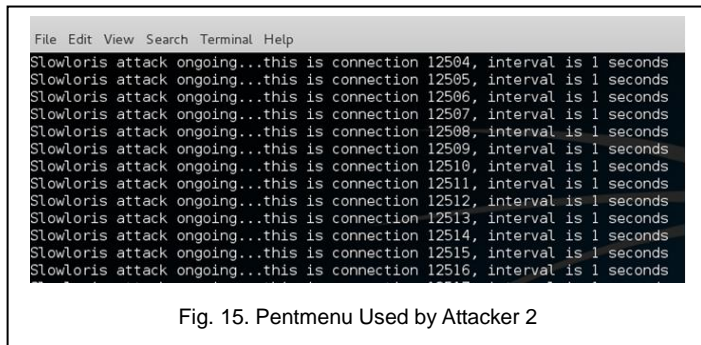


Fig. 15. Pentmenu Used by Attacker 2

Figure 15 is a screenshot of a Pentmenu used by attacker 2, targeting the endpoint (<http://security.proof>) with a Slowloris attack. The Slowloris DDoS software is integrated inside Pentmenu bash script. Slowloris attack is opening as many connections as possible with the server in order to make the

service go down.

Figure 16 shows that requests have already started to fail. Meaning that, for the attacker, the load balancer failed. This will eventually lead for the backup load balancer to take over.

Once the load balancer was down, we tried again to reach the endpoint from each party on the network. Results came as seen below:

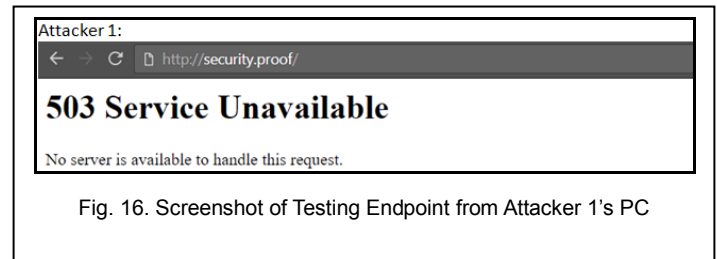


Fig. 16. Screenshot of Testing Endpoint from Attacker 1's PC

Thus the endpoint (<http://security.proof>) is no longer reachable by attacker 1 and all the requests are dropped and are no longer forwarded to the service. It might look like the service went offline and the attack was successful, but our other testers will prove otherwise.

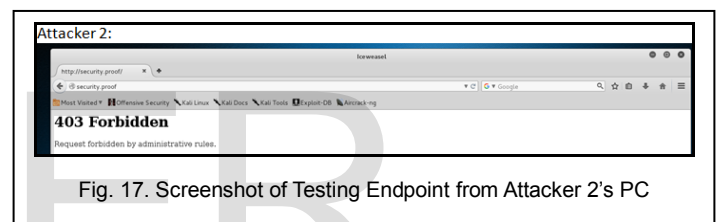


Fig. 17. Screenshot of Testing Endpoint from Attacker 2's PC

Figure 17 shows that our endpoint (<http://security.proof>) is no longer reachable by attacker 2 and all of his requests are dropped and no longer forwarded to the service. It might look like the service went offline and the attack was successful, but our other testers will prove otherwise.

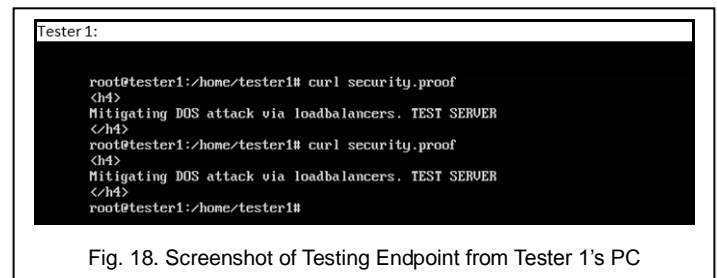


Fig. 18. Screenshot of Testing Endpoint from Tester 1's PC

Figure 18 reveals that the endpoint (<http://security.proof>) is still alive and functioning normally. And since Tester 1 is marked as a trusted user, she/he was still able to reach our service and use it normally without any interruption.



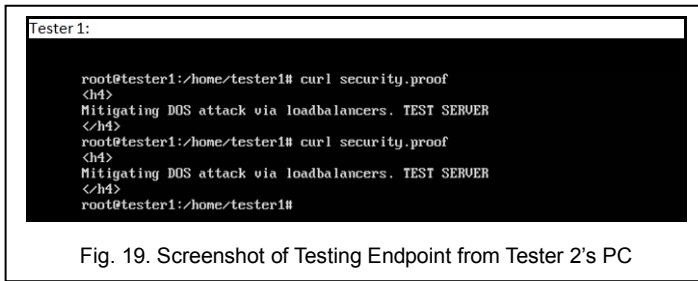


Fig. 19. Screenshot of Testing Endpoint from Tester 2's PC

Figure 19 indicates that the service is no longer accepting requests from tester 2. Although tester 2 is not an attacker, yet, she/he is a new visitor and not marked as trusted. In order to protect our network, the service closed during the attack and turned into a trust based network allowing only trusted visitors to pass (tester 1).

## 10 RESULTS

At first, the endpoint or service (<http://security.proof>) was reachable by all ends / parties on the network: attacker 1, attacker 2, tester 1 and tester 2.

Tester 1 is the only node marked as a trusted client / visitor. Attacker 1 and attacker 2 start the attack together in order to try to have the service failed. The attack seemed to have succeeded and we can clearly see that attacker 1 and attacker 2 are no longer able to reach the service. However, in reality, the attackers have succeeded to take down the first load balancer which routes all traffic to our service. On the other hand, the backup load balancer took over, allowing only trusted visitors' requests to pass whilst attacker 1 and 2 believed that the service is down. To prove this claim, tester 1 visited the service and the result shows that tester 1 was still able to access the service normally without any interruption.

Alternatively, despite the fact that tester 2 is not an attacker; she/he was not able to reach the service either. That's because only trusted sources are now allowed to pass and tester 2 is not marked as trusted.

## 11 CONCLUSION AND RECOMMENDATIONS

Security experts and network administrators used to act based on their proficiencies and practices to mitigate network attacks rather than objective metrics and models. This study provided an innovative mechanism to protect smart city network. It is a defensive in nature relying on a heuristic approach associated with load balancers to mitigate DDoS attacks. Based on the fact that IOT devices can easily be exploited and controlled to perform DDOS attacks, we introduced a new way to mitigate DDOS attacks using an array of load balancers. Normal load balancer allows traffic to pass normally until an attack occurs, and backup load balancer uses weighted, dynamically and smartly assigned heuristics to determine whether or not to allow requests to pass. If the heuristic approach classifies the

request as trusted, it will reach the protected services, otherwise it shall be dropped.

Having implemented the solution thoroughly explained in our research, pave the way to a new subset of recommendations. We suggest that if the following recommendations are aligned together with our solution will definitely lead to significant and robust results in reducing systems vulnerabilities and mitigating attacks.

a- Secure by Design: Security measures should be integrated while designing the IoT products. Survey [31] showed that only 48% of companies focus on securing their IoT products from the beginning of the product development phase.

b- End-to-End protection architecture: Devices and systems must be tolerant to attack, they have to be kept operating under all circumstances.

c- Hardware embedded security: leads to more resilient systems and helps shortening the time needed for services to return to their original states after an attack.

## REFERENCES

- [1] Anna-senpai, Mirai Source Code on GitHub, September 2016, Source: <https://github.com/jgamblin/Mirai-Source-Code>.
- [2] B. Krebs, "Who is Anna-Senpai, the Mirai Worm Author?", January 2017, Source: <https://krebsonsecurity.com/2017/01/who-is-anna-senpai-the-mirai-worm-author/>.
- [3] Comparative Study of Preventive Algorithms Of Ddos Attack by Divyashree Chavan, Colleen Francis, Elbin Mary Thomas, Prathama Moraye. International Journal of Scientific & Engineering Research, Volume 7, Issue 2, February-2016, ISSN 2229-5518.
- [4] A. Mpitiopoulos, D. Gavalas, C. Konstantopoulos and G. Pantziou, "A survey on jamming attacks and countermeasures in WSNs", IEEE Communications Surveys & Tutorials, vol. 11, no. 4, pp. 42-56, 2009.
- [5] B. Krebs, "KrebsOnSecurity hit with record DDoS," in KrebsOnSecurity, 2016. Source: <https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/>.
- [6] B. Krebs, "The Democratization of Censorship," in KrebsOnSecurity, 2016. Source: <https://krebsonsecurity.com/2016/09/the-democratization-of-censorship/>.
- [7] R. Millman, "OVH suffers 1.1Tbps DDoS attack," in News, SC Magazine UK, 2016. Source: <http://www.scmagazineuk.com/ovh-suffers-11tbps-ddos-attack/article/524826/>.
- [8] S. Hilton, "Dyn analysis summary of Friday October 21 Attack," in Dyn, 2016. Source: <http://hub.dyn.com/dyn-blog/dyn-analysis-summary-of-friday-october-21-attack>.
- [9] T. Spring, K. Carpenter, and M. Mimoso, "BASHLITE family of Malware Infects 1 Million IoT devices," in Threat Post, Threatpost, 2016. Source: <https://threatpost.com/bashlite-family-of-malware-infects-1-million-iot-devices/120230/>.
- [10] B. Krebs, "Source code for IoT Botnet 'Mirai' released," in KrebsOnSecurity, 2016. Source: <https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/>.
- [11] B. Krebs, "Europe to push new security rules amid IoT mess," in KrebsOnSecurity, 2016. [Online]. Available: <https://krebsonsecurity.com/2016/10/ikepe-to-push-new-security-rules-amid-iot-mess/>.
- [12] Surviving Distributed Denial of Service Attacks, Liu, S, IEEE Computer, vol. 11, no. 5, pages: 51-53, 2009.

- [13] Defeating Distributed Denial of Service Attacks, Geng and Whinston, IT Professional, vol. 2, no 4, pages: 36 – 42, 2006.
- [14] Defending Against Flooding Based Distributed Denial of Service Attacks: A Tutorial, Communications Magazine, IEEE, vol. 40, no. 10, pages 42-51, 2002.
- [15] [https://en.wikipedia.org/wiki/Heuristic\\_\(computer\\_science\)](https://en.wikipedia.org/wiki/Heuristic_(computer_science))
- [16] 2017[Online]Available:[https://www.researchgate.net/profile/Natallia\\_Kokash2](https://www.researchgate.net/profile/Natallia_Kokash2). [Accessed: 21-Sep-2017].
- [17] A comparison of heuristics algorithm for load balancing in cloud environment by Yogita kaushik, Anup Bhola, C.K Jha. International Journal of Scientific & Engineering Research, Volume 6, Issue 9, September-2015, ISSN 2229-5518
- [18] Sustaining availability of Web Services under Distributed Denial of Service Attacks, Wooyong Lee, Jun Xu, Computers IEEE Transactions on, vol. 52, no. 2, pages: 195-208, 2003.
- [19] Access Control Lists to Protect a Network from DoS Attacks, Dennis Eck, 2003.
- [20] Simulation of DOS, DDOS attacks & Design Test its Countermeasures by Aditi Srivastava, Deepak Chaudhary. International Journal of Scientific & Engineering Research, Volume 5, Issue 1, January-2014, ISSN 2229-5518.
- [21] <https://security.radware.com/ddos-knowledge-center/ddospedia/loic-low-orbit-ion-cannon/>

IJSER