Composing DDoS Attacks on Gaming Servers

John Sipahioglu
Department of Computer Science
Kent State University at Stark
North Canton, OH, USA
jsipahio@kent.edu

Younghun Chae
Department of Computer Science
Kent State University at Stark
North Canton, OH, USA
ychae@kent.edu

Abstract*-- Distributed denial of service (DDoS) attacks constantly threaten network servers, with attackers motivated by various factors, ranging from trolling to extortion attempts. These attacks can have severe financial implications for victims, regardless of the attackers' motivations. In recent years, game servers have been particularly vulnerable to DDoS attacks. This paper focuses on the exploration of DDoS attack methods utilizing Kali Linux. The study investigates two primary categories of DDoS attacks: application-level attacks and network floods. Using Metasploit Framework and hping3, various network flood attack types are demonstrated after conducting an initial survey to identify the target server. The executed attacks include SYN floods, ICMP reflection, and UDP floods. Our findings indicate that UDP floods, explicitly targeting the game server's designated port, proved to be the most effective in disrupting server performance. Although the objective of service denial was not fully achieved, players could still participate in the game, albeit with some latency. Conversely, the SYN flood and ICMP reflection attacks were ineffective in causing any disturbance to the player's experience.

Keywords—DDoS attacks, network floods, reconnaissance, Kali Linux, Metasploit Framework, hping3

I. Introduction

A Denial of Service (DoS) attack is an attack where the aim is to render the service being attacked unavailable. A Distributed Denial of Service attack occurs when multiple computers (sometimes referred to as vectors) launch a DoS attack simultaneously. These attacks work by either overloading the network, by making more requests than the network or server can handle, or by overloading the application layer by making computationally expensive requests that eat up computational resources [1]. In the context of game servers, this can result in the legitimate server users being booted due to network overload. While a DDoS attack's goal is to cause grievance to others, sometimes it is also meant as a distraction to launch a more invasive hacking attack. Even those connected to the servers could be infected by an attacker using a DDoS attack as a distraction method [2].

The move to virtual work and online gaming provided increased opportunities for cyber-attackers. While cybercrime rings may be using these attacks as part of ransomware and data theft attacks, the ease and low price with which a DDoS attack can be set up allows ordinary internet users to set up DDoS attacks [3]. The once popular game Titanfall has been virtually ruined and was pulled from the market due to DDoS attacks, while Titanfall 2 also suffers heavily from DDoS attacks but survives because "it also includes an absolutely outstanding campaign and is still worth a pickup as a single player game" [4].

On Jan. 15, 2023, the 24 Hours of Le Mans Virtual tournament was the target of several DDoS attacks that caused backlash for the host of the event, Motorsports Games. The event featured a prize of US $250,000, and participants included many real-world racing drivers, including 2-time F1 World Champion Max Verstappen. While leading the event by over a minute, Verstappen was disconnected and had dropped to 17th place by the time he reconnected, ultimately retiring from the race. This occurred after an earlier admission from the organizers that servers had suffered a security breach [5].

DDoS attacks are shown here to have pervasive impacts on the online gaming industry, rendering some games unplayable and causing chaos for online tournaments. This research aims to show how DDoS attacks can be performed, as well as steps that can be taken to combat them. Afterward, the steps to perform a DDoS attack on a home lab will be explained and executed. Afterward, the results of that attack and the methods used to counteract it will be reported and discussed.

II. Literature Review

DDoS attacks are often initiated when a malicious actor infects other internet hosts to install the tools needed to execute the attacks. These hosts are referred to as zombies. When it is time for the attack, a start signal is sent to these zombies to begin the attack. Attacks can be generalized into software exploits and flooding attacks. While attacks can be single or multi-sourced, they can be difficult to differentiate since attackers often spoof source addresses. A reflector attack is an attack that spoofs the source IP to match the victim and has unknowing hosts send SYN-ACKs, or ICMP replies to the victim [6].

A. Application-level DDoS attacks

Application-level attacks are more sophisticated than flooding attacks and often look like legitimate requests. These attacks exploit software weaknesses or overload the server's computational power. One such example was an attack on the e-commerce site WeaKnees.com, where a competitor launched an attack that made HTTP requests for many large image files, which crippled the site for two weeks and cost millions in revenue [7]. The most seen defense against application-level attacks is CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) puzzles. Any user

attempting a connection must complete one of these puzzles before the connection or request will be handled. However, these puzzles are not user-friendly and may drive away legitimate users. Other issues with CAPTCHA puzzles are the advent of image-recognition systems and reusing tokens for passed CAPTCHAs. The other options are often statistics based, looking to classify traffic as malicious based on factors such as time spent on the page, browsing repetition/cycles, uptime/downtime, arrival distribution, and IP distribution [8].

B. Network flooding

In flooding attacks, an army of zombies, called a Botnet, send a large amount of traffic to the host being targeted. The goal is to slow the response time of the server or crash the server altogether. Spoofing of IP addresses makes it harder to find the zombie computers and nearly impossible to determine the original perpetrator of the attack. These can target the application layer (section 2A) or the network/transport layer. This can include TCP floods, UDP floods, reflection, and amplification. These types of attacks are generally easier to handle and not as common these days due to these limitations [9].

C. Detection methods

Detection methods can be signature or anomaly-based. Signature-based detection tries to match network traffic with the patterns found in known attack types. Anomaly-based detectors look for statistical deviations in network traffic to identify an attack. While signature-based methods are more accurate, they are less extensible, and thus, anomaly-based protocols are the most widely used [10]. One anomaly-based method for detecting DoS attacks is clustering, where normal data is fit into clusters, and any new data into the system that does not fit into the existing clusters is considered an anomaly [11]. Deep neural networks (DNNs) have recently been applied to detect network anomalies. The black-box nature of DNNs makes their use in a corporate environment difficult due to the difficulty in explaining their predictions. In [12], a DNN model that produces relevance scores for its input data was proposed as a model that gives additional information along with its prediction.

D. Prevention methods

The most critical part of preventing a DoS attack is to catch it as soon as possible and to stop the attack as close to its source as possible [9]. As previously discussed, CAPTCHA is the most common tool used to prevent attacks at the application level [8]. Mahjabin et al. provide a survey of prevention methods, which we will summarize here [13].

1) *Ingress/egress filtering*: This method works at the edge routers of a subnet and filters out packets that do not fall within the expected range of IP addresses. This makes it more difficult to spoof IP addresses in the DoS attack, as the edge router will filter out most of them. However, intelligent spoofing can overcome this, and complex topologies and mobile networks complicate implementation.

2) *Martian address filtering*: This method works on inbound routers and will prevent the delivery of packets from IPs from a specified range and unallocated. This method is only as effective as its specified address range and, like the ingress/egress filtering, can be defeated if IPs are spoofed from an acceptable range.

3) *Source address validation*: This is another protocol implemented on the inbound routers, and it works by preventing the delivery of packets showing an interface match on the source and destination route. This method can produce a high false-positive rate.

4) *Route-based packet filtering (RPF)*: This method requires the Border Gateway Protocol (BGP) to be implemented on the core routers and will filter packets based on their route information. Its primary drawbacks are the additional network protocols that need to be implemented and the possibility that route information can be spoofed if the BGP session information is known to the attacker.

5) *Source address validity enforcement (SAVE)*: SAVE is an improvement on the RPF protocol that forces routers to send updated source info to destination routers. However, it also requires additional modifications to the existing router structure, and partial implementation does not guarantee success.

6) *Hop-count filtering*: Uses time-to-live (TTL) information to count hops by packet. This protocol depends on the router at the victim's location and thus is a last line of defense. It is a lightweight technique that has low storage demands. However, attackers can forge the valid hop count, and false positives can occur to different TTL values based on differing operating systems.

7) *History-based filtering*: Another method using the victim's local routers, this protocol uses the traffic history to inform which traffic is valid. This is effective against bandwidth attacks but does not work well when attackers can simulate regular traffic.

8) *Path identifier-based filtering*: Once an attack is identified, this method looks to learn the attacker's path to block packets from that path. This works well if the path can be detected, but it requires many routers' involvement and can lead to false positives for a small identification field.

9) *Packet-score*: This method uses statistical analysis to assign each packet a score based on a profile value analysis. Bayes' theorem calculates a conditional legitimate probability (CLP) to determine if the packet is real traffic or part of an attack. This method can identify packets even if the IPs are not spoofed. However, it requires a significant memory overhead and well-defined model for traffic to base its profile value analysis on.

III. Attack Methods

A server for the game Garry's Mod sandbox was set up and subjected to DoS attacks. According to the product page on the game distribution platform Steam, Garry's mod is a physics sandbox where you can spawn objects and weld them together [14]. Due to the constraints of a home lab and the dangers of exposing a server to an external network, there will be up to only three attackers at once. It is important to remember that actual DDoS attacks can use thousands of zombies as part of a botnet and that this is a small demonstration of an attack.

The Garry's Mod Sandbox server was set up on a virtual machine using 4 GB of RAM and two processor cores. The

operating system used was Ubuntu Server 22.04 LTS. The utility steam cmd was used to download and install the standalone server files. The server runs on the default port for Garry's Mod, which is 27,015. The server has no particular configuration to attempt to prevent any DoS attacks.

The attackers will be virtual machines running Kali Linux and using the Metasploit Framework included in Kali. There will be up to three attackers at one time due to the ram limitations of the host computer. Additionally, there will only be one "player" during most of the attack. Again, this is due to lab limitations and network safety. The player will be connected to the game server through the legacy browser for Garry's Mod multiplayer. The player will attempt to spawn objects, move around, and move the objects around the map. The Windows 10 desktop being used by the player also hosts all the virtual machines.

The types of attacks are as follows: single-source SYN Flood DoS attack, multi-source SYN flood DDoS attack, reflection attack, a single source UDP flood DoS attack, and a multi-source UDP flood DDoS attack. The procedures for the attacks are presented in the following sections. For all commands shown, it is assumed that the root account is being used (signified by #); `sudo` command could also be used instead.

A. Reconnaissance

We must first survey to identify a potential target to simulate an attack. Using Nmap, a network scan will be performed to identify open ports. The open ports will then be searched to see if any match standard game server ports using hping3's scan feature. Once a target is identified, additional recon will be performed to identify the target's operating system and other open ports that may create other vulnerabilities.

B. Single-source SYN DoS

In this attack, only one of the Kali Linux attackers will be launching a SYN flood against the game server. This will be executed by using the "auxiliary/dos/tcp/synflood" tool in the Metasploit Framework included in Kali Linux. The RHOSTS field will be set to that of the game server identified by the reconnaissance, along with the RPORT equal to the port of the game being targeted. The commands required are shown below.

```
msf > use auxiliary/dos/tcp/synflood
msf > set RHOSTS <target IP>
msf > set RPORT <target PORT>
msf > exploit
```

Here, `<target IP>` is the IP address of the game server and `<target PORT>` is the port that the game being targeted is running on. In the SYN flood, the SYN packets are sent with random, fake source addresses, so when the server sends back its SYN-ACK packet it will never receive back the ACK to complete the three-way handshake.

C. Multi-source SYN DDoS

This attack will follow the same framework as the single-source attack. The same commands will be run in Kali Linux using Metasploit Framework, with the RHOST and RPORT fields all being set to those of the game server. This way, the attack will be executed by multiple sources. This will not only allow for greater traffic, but in a real attack, this would also help cut off attempts to stop the attack as it would require identifying and blocking more source paths, which is already more difficult due to the spoofed IP addresses [6].

D. ICMP reflection

This time, the attacker will be sending ICMP ping packets to another host, but with the IP spoofed to match the IP of the game server being attacked. This way, the ping responses will be sent to the game server. This will be done using hping3 in Kali Linux. The command issued is the following:

```
# hping3 --icmp --flood <mirror IP> \
--spoof <target IP>
```

The `--icmp` signifies that we are sending ICMP ping packets and the `--flood` signifies that we are flooding our target with as many packets as can be sent and automatically dropping any responses. The `<mirror IP>` represents the IP address of the intermediary computer in the reflection attack, while `<target IP>` is the IP address of the game server. The `--spoof` signifies that we are setting the source IP of our ICMP packets to that of the game server. This will lead to the mirror sending its ICMP responses back to the game server instead of sending them back to our VM launching the attack.

E. Single-source UDP flood

The previous attacks either attempted to exploit the TCP (section B and C) protocol, or simply flood the network with enough traffic to compromise the connection (section D). Here, a targeted attack on the UDP protocol and port will be executed. Since UDP is a connectionless service, it cannot be exploited by targeting a protocol application, as was the case with the SYN handshake in TCP. Here, the aim will simply be to congest the UDP port being targeted with as much traffic as possible. The tool hping3 will be used for this. The following command is executed for the attack:

```
# hping3 --udp --flood <target IP> /
  -p <target port> --rand-source
```

The `--udp` flag indicates to send UPD packets, and `--flood` specifies that packets should be flooded to the target. The `<target IP>` is the IP address of the game server, and `-p <target port>` indicates which port to direct the attack against. Finally, `--rand-source` indicates to spoof the source address of the packets.

F. Multi-source UDP flood DDoS

Like the multi-source SYN flood attack, this approach executes will execute the command shown in section E on three Kali Linux VMs at the same time. The intended consequence is to inundate the network with an increased volume of packets. In a genuine attack scenario, this would pose challenges in terms of mitigation, as the packets would originate from multiple sources, making it more challenging to halt the attack effectively [6].

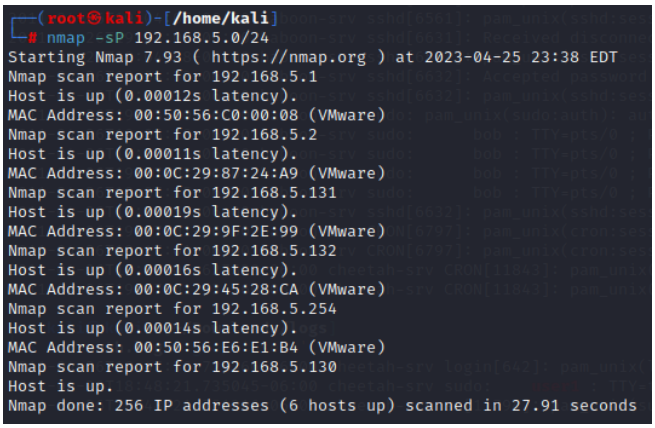

Fig. 1. Output from host scan using nmap

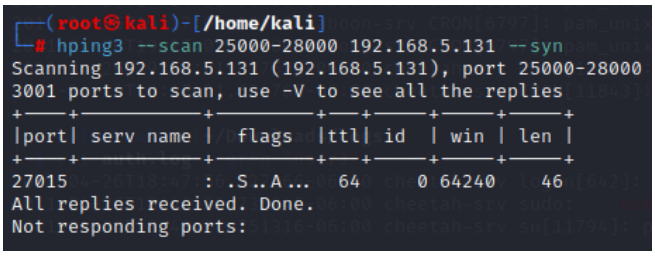

Fig. 2. hping3 showing port 27015 open for potential target 192.168.5.131

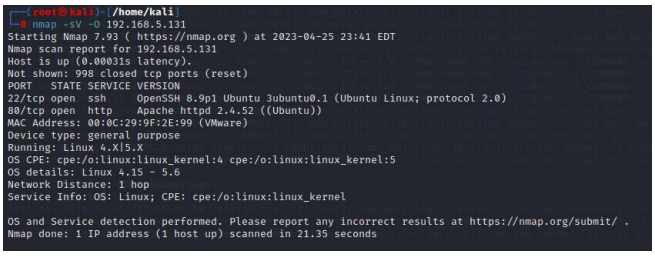

Fig. 3. Detailed nmap scan of target, showing the open TCP ports 22 and 80 and the Linux kernel 4.15-5.6 operating system

IV. Results

The gameplay performance scale will be a rating between 0 and 4, where 0 represents a disconnection, 1 represents performance so poor the game is considered unplayable, two is considered barely playable, 3 is average performance, and 4 represents a smooth experience. The packet rate per second for each attack will also be displayed (see Table 1). The overall traffic to and from the server, the client-server communication (in both directions), the total incoming traffic to the server, and the total outgoing traffic from the server will be presented.

A. Reconnaissance

In the initial phase, a network scan was conducted to identify the operational hosts within the network, with the Nmap command line tool. Results are shown in Fig. 1. Subsequently, the hping3 scan functionality was employed to detect open ports within the typical range utilized by game servers. Among the identified hosts, the IP address 192.168.5.131 was found to have an open port 27015, commonly associated with the source game Garry's Mod. Fig. 2 showcases the output of the hping3 scan. With a potential target now identified, a subsequent iteration of Nmap was utilized to gather additional information regarding the target, such as the operating system in use and any other accessible ports. The scan revealed an open port 22 running the OpenSSH service and port 80 for an Apache web server. Furthermore, Nmap deduced that the operating system employed a version of the Linux kernel between 4.15 and 5.6. The details of this can be found in Fig 3.

B. Single-source SYN DoS

As discussed in section III, this attack features a single source flooding the target with randomly sourced SYN packets that will never have the final ACK packet sent back. The aim is to overwhelm the server by causing it to wait indefinitely for many ACK packets from the sources of the initial SYN packets. The gameplay performance was unaffected by this attack, however. There was no lag or any change in the user experience. The gameplay rating, in this case, would be a four out of four. The total packet flow rate increases to 2,748.81 packets per second. Compared to the original packet flow rate of 128.76 packets per second, this is 2034% increase in total network traffic. Despite this, since Garry's Mod uses UDP, the TCP SYN flood seems ineffective in this case. There is no significant difference (0.03% increase) in the flow rates between the game server and the actual client on the server.

C. Multi-source SYN DDoS

The force of the three attackers can be seen in the packet analysis, as the total flow for this attack is 6,026.18 packets per second, a 119% increase from the single source attack and a 4580% increase from the original flow rate. However, despite the increased traffic, the gameplay experience is still unaffected and would be rated a four once again. The flow rate between the client and server is not changed in any statistically significant way during both SYN floods, with there being only a 2.2% decrease in this case and a 0.03 increase in the single source.

D. ICMP reflection

The results of this attack are immediately seen the flow rate; the traffic explodes to 91,343.4 packets per second. This represents 709 times the traffic created by just the client and server. Despite this, the effects are not felt by the player on the game server, as the experience is still rated as a four, with no drop off in performance noted. There is also no significant change in the flow from the server to the client, like in the previous attacks, as there was only a 0.12% decrease in flow rate. There is a small but noticeable drop in the client to server communication, about a 6.24% decrease.

E. Single-source UDP flood DoS

Once again, there is no notable change in performance on the server, gameplay would be scored as four out of four once again. The packet flow rate was 47,300.41. Notably different from the other attacks; this attack does not cause an increase in responses from the server, as UDP is a connectionless protocol. Unlike in the TCP SYN flood, which attempted to take advantage of a protocol weakness, the aim here was to disrupt through the sheer amount of incoming traffic for the server to handle. As such, the incoming flow rate of 47,232.16 is the

TABLE I. DISPLAY OF FLOW RATES AND GAMEPLAY RATINGS FOR ATTACKS CARRIED OUT

Attack Type	Gameplay Rating	Total Flow	Total Incoming	Incoming to UDP 27015	Client to Server	Total Outgoing	Server to Client
None	4	128.76	62.52	62.3	62.3	65.99	65.99
SYN DoS	4	2,748.81	1,372.72	63.32	63.32	1,375.66	66.01
SYN DDoS	4	6,026.18	3,258.34	61.24	61.24	2,768.06	64.53
Reflection	4	91,343.4	45,667.92	58.41	58.41	45,675.93	65.91
UDP DoS	4	47,300.41	47,232.16	47,230.58	57.34	68.9	65.98
UDP DDoS	2.5	115,402.92	115,336.84	115,336.75	58.15	66.16	66

TABLE II. LIST OF DISPLAY FILTERS USED TO GET PACKET COUNTS FOR EACH FLOW MEASURED

Flow Type	Wireshark Display Filter
Total	ip.addr==192.168.5.131
Total Incoming	ip.dst==192.168.5.131
Incoming to UDP 27015	ip.dst==192.168.5.131&&udp.dstport==27015
Client to Server	ip.dst==192.168.5.131&&udp.dstport==27015&&ip.src==192.168.5.1
Total Outgoing	ip.src==192.168.5.131
Server to Client	ip.src==192.168.5.131&&udp.srcport==27015&&ip.dst==192.168.5.1

highest measured so far, with the incoming flow comprising 99.86% of the total flow during this attack. The communication from the server to the client only saw a decrease of 0.01 packets per second, which is barely any difference. However, the flow rate from the client to the server decreases by 7.96%.

F. Multi-source UDP flood DDoS

This is the first time that a drop in performance has been detected in the player's connection. There is noticeable lag when moving around, as the player will "teleport" backwards a few steps when moving a large distance. Additionally, the "Physgun" tool wielded by the player for moving props around the map becomes notably less effective, as props can no longer be flung long distances. Usually, the Physgun emits a blue laser that allows the player to grab and manipulate objects that have been spawned on the map, including the rotation, movement, and even tossing objects across the map. When the game lags in this attack, however, these functionalities become limited.

The packet flow rate here is 115,402.92 packets per second, the highest by far. Additionally, almost all of this is incoming flow, with the incoming flow rate being measured at 115,336.84 packets per second, or 99.94% of the overall flow. The gameplay rating in this case was a 2.5, as while the game was still playable, it was a much less fun experience and some of the functionality of the game was compromised. Like the other attacks, the flow from the server to the client was hardly affected, with the flow only differing by 0.01 packets per second. Like the ICMP reflection and UDP flood, which both send a large amount of UDP traffic, there is a noticeable, but still small, drop in communication from the client to the server. In this case, there was a decrease of 6.66%.

G. Discussion

All attacks demonstrated here failed to achieve their goal of denying service to the end user, with only one demonstrating a notable disruption. Here we present some theories as to why that is. First, the amount of traffic that a game server (or any public server) expects to face is massive. Thus, the abilities of one, or even a few machines, to interrupt the server is very limited in comparison to what a real server can handle. Two, Garry's Mod uses UDP instead of TCP, which leads the SYN flood attacks useless unless they are so large, they clog the network itself and not just the server, as the TCP and UDP ports are different. This also explains why the ICMP reflection attack was ineffective. While it uses UDP, ICMP operates on a different port than the Garry's Mod server.

Finally, we succeeded by targeting the specific port used by the game server while using the correct UDP protocol. Still, this required a massive amount of traffic being generated by multiple hosts. If the goal were to deny service altogether, in other words, to boot players from the server, the amount of traffic would need to be increased significantly while still taking this targeted approach. True attacks use botnets, which can allow the attacks to be launched by hundreds to even thousands of hosts [9]. If a disruption to a server that is only serving a single client can be compromised by three VMs, it is easy to see how an entire botnet could succeed in rendering an entire server unavailable.

V. CONCLUSION

DDoS is a highly prevalent type of attack used against network servers. The motives for these attacks can vary from trolling, extortion, or even a dirty competitive tactic. No matter their reason, they can be highly costly to their victims. DDoS attacks may focus on overloading the application level by

overusing CPU and memory resources, or they can work by overloading the network by flooding an amount of traffic the server cannot handle. Here, the network flood style of DDoS attack was demonstrated against a game server. We demonstrated basic reconnaissance and executed multiple attacks using the Metasploit Framework and hping3. The SYN flood attacks using Metasploit, as was the ICMP reflection using hping3, were ineffective. However, a UDP flood on the UDP port used by Garry's Mod server was effective in causing some inconvenience to the server user. Future work would include testing how various detection and mitigation techniques work against various attacks.

REFERENCES

[1] UK National Cyber Security Centre. "Denial of Service (DoS) guidance." ncsc.gov.uk. https://www.ncsc.gov.uk/collection/denial-service-dos-guidance-collection (accessed Mar. 24, 2023).

[2] Microsoft 365. "DDoS Attacks in Gaming." Microsoft.com. https://www.microsoft.com/en-us/microsoft-365-life-hacks/privacy-and-safety/ddos-attacks-in-gaming#:~:text=So%2C%20what%20does%20DDoS%20mean,of%20information%20to%20the%20server. (accessed Mar. 24, 2023).

[3] B. Khan. "The Gaming Industry's Latest Challenge: DDoS Protection." a10networks.com. https://www.a10networks.com/blog/the-gaming-industrys-latest-challenge-ddos-protection/ (accessed Mar. 24, 2023).

[4] A. Chalk. "After years of struggling against DDoS attacks, Titanfall is being removed from sale." pcgamer.com. https://www.pcgamer.com/after-years-of-struggling-against-ddos-attacks-titanfall-is-being-removed-from-sale/ (accessed Mar. 24, 2023).

[5] G. Cluley. "Hackers disrupt 24 Hours of Le Mans Virtual esports event." bitdefender.com. https://www.bitdefender.com/blog/hotforsecurity/hackers-disrupt-24-hours-of-le-mans-virtual-esports-event/ (accessed Mar. 24, 2023)

[6] A. Hussain, J. Heidemann, C. Papadopolous. "A Framework for Classifying Denial of Service Attacks." Proceedings of ACM SIGCOMM, Karlsruhe, Germany, Aug 2003, pp. 99-110

[7] M. Srivatsa, A. Iyengar. "Application-Level Denial of Service" in Encyclopedia of Cryptography and Security. Boston, MA, USA. Springer. pp. 42-44.

[8] H. Beitollahi, G. Deconinck. "Tackling Application-layer DDoS Attacks." Procedia Comp. Sci. vol. 10, pp. 432-441, 2012, doi: 10.1016/j.procs.2012.06.056. [Online] Available: https://www.sciencedirect.com/science/article/pii/S1877050912004139

[9] S. T. Zargar, J. Joshi and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," in IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2046-2069, Fourth Quarter 2013, doi: 10.1109/SURV.2013.031413.00127.

[10] I. Ozcelik, Y. Fu and R. R. Brooks, "DoS Detection is Easier Now," 2013 Second GENI Research and Educational Experiment Workshop, Salt Lake City, UT, USA, 2013, pp. 50-55, doi: 10.1109/GREE.2013.18.

[11] M. Ahmed, A.N. Mahmood, J. Hu. "A survey of network anomaly detection techniques." J. Netw. Comp. Appl. vol. 60, pp. 19-31, Jan. 2016. doi: 10.1016/j.jnca.2015.11.016

[12] K. Amarasinghe, K. Kenney and M. Manic, "Toward Explainable Deep Neural Network Based Anomaly Detection," 2018 11th International Conference on Human System Interaction (HSI), Gdansk, Poland, 2018, pp. 311-317, doi: 10.1109/HSI.2018.8430788.

[13] T. Mahjabin, Y. Xiao, G. Sun, W. Jiang. "A survey of distributed denial-of-service attack, prevention, and mitigation techniques." Int. J. Distrib. Sensor Net. vol. 13, no. 12, 2017, doi: 10.1177/1550147717741463

[14] "Garry's Mod on Steam". steam.com. https://store.steampowered.com/app/4000/Garrys_Mod/. (accessed May 29, 2023)