Covert Communication over ICMP

Georgios Merezas, Lars Wüstrich*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: georgios.merezas@tum.de, wuestrich@net.in.tum.de

***Abstract*—In the ever-expanding landscape of the internet, robust security measures are critical components of any modern technology. As our dependence on digital networks continues to grow, so does the sophistication of malicious actors seeking to exploit vulnerabilities that arise in all complex systems. One of the many tools of these malicious actors is ICMP tunneling, a method used for the establishment of hidden channels within network environments. These channels allow hidden and unfiltered communication of infected machines with potential attackers. The paper explores the structure of ICMP packets and how it facilitates them to tunnel TCP connections. Furthermore, it examines some existing applications of ICMP tunneling, like botnets. Finally, it proposes some countermeasures to ICMP tunneling.**

***Index Terms*—ICMP tunneling, covert channels, botnets**

1. Introduction

In the increasingly interconnected world that we live in, the internet has become an integral part of our lives. As everything in our lives becomes digitalized, the interest of malicious actors in stealing and exploiting digital data also increases. Command and Control (CnC) techniques are used to send commands and receive data from infected machines, which allows an attacker to continuously communicate with them [1]. This can be lucrative when one can steal business secrets or mine digital currency on compromised devices [2]. State actors, too, have a use for remote access to machines, either to spy or to launch Distributed Denial of Service (DDoS) attacks against foreign businesses as a form of economic protectionism [3].

One way to achieve remote access to a host and keep the communication hidden is to use ICMP tunneling [4]. This paper should provide insight into cybersecurity challenges that were first reported in 1997 [4] [5]. Additionally, it should highlight how simple network status messages can be exploited in unintended ways. The contribution of this paper is: (a) an explanation of ICMP tunneling, (b) an exploration of its applications, (c) a summary of existing proposals to prevent ICMP tunneling.

Section 2, introduces ICMP and ICMPv6, with their standard use cases. In Section 3, we first explain how ICMP tunneling works and then outline its capabilities. In Section 4, existing applications of ICMP tunneling are explored. Section 5, introduces preventative measures to combat ICMP tunneling.

2. Background

Internet Protocol (IP) is a network layer protocol, that forms the basis of the Internet. It is used to transmit IP packets globally between hosts. An IP packet consists of an IP header and IP data. IP data, i.e. the content that is transmitted by a given packet, is appended after the header (encapsulated within the IP packet). Such data is for example ICMP or higher layer protocols. The IP header specifies the source and destination addresses, if it is a fragment, packet length, and other packet information. [6]

User Datagram Protocol (UDP) is a simple transport layer protocol. It specifies the source and destination ports of the datagram and a checksum for error detection. The ports of UDP can be used to multiplex the communication between two IP hosts. The only feature of UDP that can be used for data loss, is the fact that error messages about IP packets are sent when a packet is lost in transit. [7]

Transmission Control Protocol (TCP) is a widely implemented transport layer protocol. Similar to UDP it has ports for multiplexing and a checksum for error detection. However, TCP has a distinct advantage over UDP in that it builds the connection at the start of the communication and destroys it at the end. Most importantly, during the communication, it acknowledges segments it has received (a segment being sent as an IP packet). This feature builds the basis for congestion and flow control. When the sender does not have enough acknowledgments for the segments that have already been sent, it starts sending them at a slower rate to avoid possible congestion in the network between it and the receiver. When the sender receives an acknowledgment for a fragment that was sent too long ago or does not receive an acknowledgment in some predetermined time, it tries to resend it. This means that a TCP connection is reliable, and can fail only when a packet can never reach its destination, e.g. in the event of a complete network shutdown. [8]

Internet Control Message Protocol (ICMP) is a network layer protocol part of the IP suite. It is primarily used for error reporting and the exchange of control information. ICMP messages are encapsulated within IP packets as if they were a higher-level transport protocol, like TCP or UDP. However, they are an integral part of IP and are processed as a special case. Unlike TCP and UDP, ICMP packets do not specify ports. [9]
ICMP is used by network utilities like `ping` [10] to send ICMP Echo Requests and receive corresponding ICMP Echo Reply messages. It is used to test if a host is reachable on an IP network. Another utility, `traceroute` [11], sends out many IP packets after each other, incrementing

 doi: 10.2313/NET-2024-04-1_14

their Time-To-Live (TTL). The TTL is decremented by each router on the path to the destination. When the TTL of a packet reaches 0, the last reached router sends an ICMP Time Exceeded message to the sender. This way, the utility maps the path between the source machine and a remote host.
An ICMP messages consist of a header and content sections. The header is 4 bytes (B) long. It contains the type (1B), the code (1B), and the checksum (2B). The type specifies if the ICMP message is an ICMP Echo Request, an ICMP Echo Reply, an ICMP Time Exceeded message, etc. The code provides further information about the type, e.g., why time was exceeded. The checksum is used for error checking. The content section varies depending on the type and code of the ICMP message, and it has no preset length.
ICMPv6 is the version of ICMP used in conjunction with IPv6. It serves similar purposes to ICMP in IPv4, except for some enhancements to adapt to the features of IPv6. For example, ICMPv6 has Neighbor Discovery Protocol (NDP) instead of the ARP protocol in IPv4. ARP is used to find the Layer 2 address of devices whose Layer 3 (IP) address is known. NDP has the same functionality, in addition to many other improvements. Another example is an ICMPv6 Packet Too Big message. It exists because only the sender of the IPv6 packet can fragment it, whereas IPv4 packets can be fragmented by any router on the way to the destination.
Because the principal idea and structure of ICMPv6 is the same as ICMPv4, in this paper the explained principles of ICMP tunneling will be relevant for both IPv4 and IPv6 communication.

3. ICMP Tunneling

ICMP tunneling can be accomplished with any type of ICMP message [4]. Most tools use ICMP Echo Request or ICMP Echo Reply packets, which are normally used to test host reachability. This makes it hard to filter against malicious uses and, therefore, very appropriate to use for hidden communication. [12]

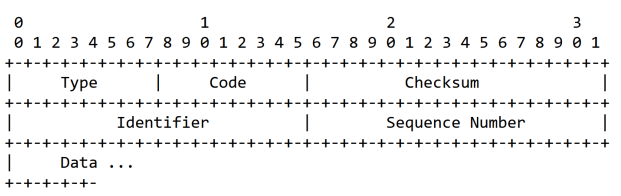

Figure 1: ICMP Echo Request and Reply as defined in RFC792 [9]

3.1. Prerequisites

The structure of ICMP allows for varied length data after the header for type 0 (Echo Reply) and type 8 (Echo Request) ICMP packets, as can be seen in Figure 1. This is defined in RFC792 [9] to allow for flexible network testing. For example, big ICMP Echo Request packets can be sent to test network load [4]. There are no specifications for the type of content in the data section of ICMP Echo Request/Reply packets. This enables malicious actors to use ICMP to transmit data to infected machines with a smaller network footprint that may go unnoticed by firewalls and other detection tools [12].

For ICMP tunneling to work, root access is needed [13]. For more rudimentary programs than the ones discussed further, root access is not always necessary. The main prerequisite is that the tunneling program is installed on the device that will be used for communication. This can be either because it was infected by a malicious actor, or due to the user of the machine wishing to hide their communication from the local system administrator.

3.2. How it works

Using ICMP Echo Request and Reply packets, a malicious attacker can embed information into the data section and achieve communication. An infected machine can listen for incoming requests and read the data that was sent. It can then confirm with an ICMP Echo Reply that the message was received.

3.2.1. Functional principle. ICMP tunneling does not work out of the box simply by using kernel sockets. A separate program needs to be written to facilitate this communication. Such a program should be able to embed the required content into ICMP packets and send them. It then should be able to listen for incoming ICMP packets and read the content of those participating in the disguised communication. It should also be able to process the embedded data and continue communicating.

An example of a program that communicates using ICMP tunneling can embed TCP/UDP content into the data section of ICMP Echo Request/Reply packets. As most programs are already created to work with either TCP or UDP, the two main transport layer protocols, such an implementation can have wide applications.

It can be designed as a tool that intercepts some program's outgoing packets or all packets on some port. It then translates them into ICMP Echo Request packets by embedding each transport layer part of the packet into the data section, before sending it out. The same tool on the infected machine listens for ICMP Echo Requests with certain predefined features that mark it as a tunneled connection (e.g. a specific predefined Identifier). Then, it intercepts the ICMP packet, extracts the TCP/UDP content, and sends that to a process on the same machine. Thus, a TCP/UDP packet is transmitted between two machines with ICMP. Essentially, such a tool has the implementation of a kernel socket with the addition of embedding the packet into an ICMP Echo Request. Ptunnel [13] is a program that uses this concept. [12]

This type of ICMP tunneling requires administrative or root privileges to be able to run. The aforementioned ICMP tunneling tool Ptunnel, for example, requires root access to be able to use raw sockets, i.e., without specifying if they are TCP or UDP [13]. This makes it harder to achieve the end goal, but not overall impossible.

3.2.2. Multiplexing. Figure 1 shows that ICMP Echo Request/Reply packets contain an Identifier and a Sequence Number. These can mark a packet as a tunneled connection, as mentioned previously. Additionally, they

 doi: 10.2313/NET-2024-04-1_14

can be used to multiplex the communication, i.e., have parallel streams that contain data for different purposes. The hosts that are performing the communication can match an incoming ICMP Echo Reply to an ICMP Echo Request they made and, therefore, know which stream this packet belongs to. Usually, ICMP Echo Request/Reply communication already happens this way in, for example, the `ping` utility.

Additionally, embedding UDP/TCP content in the data section can lead to more parallel streams. For example, one could treat the Identifier concatenated with the source port of UDP as the 'source' and the Sequence Number concatenated with the destination port of UDP as the 'destination'. The ports of UDP and TCP are 2B or 16 bits long [8] [7], and so are the Identifier and Sequence Number (see Fig. 1). In such a scheme, the port numbers could go up to $2^{(16+16)}$ or 2^{32}, instead of just 2^{16} as they are in TCP and UDP. Such high numbers would allow for greater multiplexing capabilities.

3.3. Capabilities

While ICMP packets are similar to UDP and TCP, they have distinct features that impact the quality of the communication between hosts. In this analysis, we assume that the ICMP communication is performed using ICMP Echo Request and ICMP Echo Reply packets that embed UDP or TCP into the data section, which can be seen in Figure 1.

3.3.1. Payload size. The size of an ICMP Echo Request or Reply packet, outside artificial size restrictions that may be imposed by the OS or a firewall, is only limited by the size of the IP packet that it is part of. For Ethernet communication, the Maximum Transmission Unit (MTU) size is typically 1500B [14]. The MTU defines the largest possible size for the contents of the Ethernet frame, i.e., the maximum size of a single IP packet. The IP packet itself has a minimum header size of 20B [6]. A TCP header has a minimum size of 20B [8], while UDP is only 8B [7]. That means the maximum content size for TCP is 1460B and for UDP 1472B. With the addition of the ICMP header between the IP and TCP/UDP headers, the maximum content size is reduced by 4B. Additionally, due to the use of ICMP Echo Request/Reply packets, we need to account for the 4B of the Identifier and the Sequence Number. So, the final content that we can transmit has a size of 1452B for a tunneled TCP packet and 1464B for a tunneled UDP packet. This is only 8B less than the standard TCP/UDP communication.

3.3.2. Reliability. ICMP packets do not differ significantly from UDP packets. They both lack flow or congestion control, unlike TCP. The ports of UDP can be simulated with the Identifier and Sequence Number of ICMP Echo Request/Reply packets. However, they have one major difference. ICMP error messages are not sent about other ICMP messages to prevent "the infinite regress of messages about messages, etc." [9]. This has consequences for the reliability of ICMP communication. If an ICMP message is lost, the sender will never be notified about it and, therefore, cannot retry.

In the given schema, we do not have normal ICMP Echo Request/Reply packets. By embedding TCP or UDP content into the data section, we can use features of these transport layer protocols that can affect the reliability of the communication. If we use UDP, we do not achieve more reliability because UDP does not have mechanisms for tracking sent and received segments. If we embed TCP, we can use its features to create a unique, reliable ICMP communication protocol.

The reliability features of TCP explained in Section 2 can be implemented by an ICMP tunneling tool to make the connection reliable. This transmission channel has the disadvantage of not receiving error messages when packets inevitably get lost in transit. Otherwise, it acts just like a standard TCP connection. That is because usual IP communication does not guarantee that a control message is returned in case an IP packet is not delivered [9]. Some failure needs to happen twice for the sender to not retry to send a packet after it fails: when sending the IP packet and when the ICMP error packet is sent back. In our case, only one failure needs to occur for a packet to be lost. This means that packets are resent more often, and the average segment rate is lower.

4. Applications

The use of ICMP tunneling for communication has the advantage of being hard to detect and filter against by firewalls and Intrusion Detection Systems (IDS) [12]. Logically, most applications of ICMP tunneling are those that should not be detected by third parties or are performed by malicious actors. These are so-called covert channels.

A covert channel is a communication channel that is "not intended for information transfer at all" [15]. Therefore, applications that can or do use ICMP tunneling use it as a covert channel.

A prime example of a covert channel is communication with a backdoor on an infected system. This can provide complete access to the system without the firewall blocking the channel. A more subtle covert channel would be having an infected computer with software that sends out stolen confidential data at random intervals [4].

4.1. Botnets

One of the ways that malicious actors use covert channels is for CnC communication of botnets [1]. Botnets are networks of infected computers that each run one or more bots. They are created and controlled by a botmaster who issues commands to these bots. These bots were usually centrally controlled using the IRC messaging protocol [16]; nowadays, botnets are more often deployed as peer-to-peer networks due to more sophisticated detection methods and to avoid having one point of failure. These collections of infected computers can communicate with each other, perform DDoS attacks, and send spam. They can also allow the attacker to have remote access to all of the affected devices. Some examples of BotNets include Mirai [17] [18], Mariposa [19], and others. [20] [21]

A botnet that uses ICMP tunneling is Pingback. Unlike the discussed implementation, the malware does not embed TCP into ICMP for communicating, rather it has

 doi: 10.2313/NET-2024-04-1_14

its own data in ICMP Echo Request packets, as can be seen in Figure 2. The malware is hidden inside a malicious `oci.dll` file. This file is normally loaded with two other `.dll` files by the `msdtc` Windows service. This service loads an ODBC library to support Oracle databases. The library tries to load three Oracle ODBC DLLs, one of which is the `oci.dll`. When the malware is running, it listens for ICMP Echo Request packets with sequence numbers 1234, 1235 or 1236, and can execute shell commands remotely. [22]

```
struct ICMPData {
   char  cmd[10];          // bot command (see appendix for more details)
   char  args[512];        // extra parameter, but haven't seen it used by the malware
   char  cmd_line[258];       // command line(see appendix for more details)
   unsigned long  dest_port;  // destination port
   char  dest_addr[4];        // destination IP address
}
```

Figure 2: Pingback data struct, published in [22]

4.2. Data theft

Another use of covert channels is the breach of data confidentiality. An attacker can extract data from a machine and send it out over a covert channel by infecting it. Thus, private and confidential data is stolen without any detection by the user or the network administrator. This application is particularly harmful when it comes to state actors performing espionage on foreign governmental organizations [3]. Similarly, businesses can commit illegal economic espionage and acquire business secrets to gain a competitive market advantage. Criminal organizations can steal private information and blackmail its owners into paying them ransom.

A practical application is Cobalt Strike [23], a commercial remote access tool that is used to "execute targeted attacks and emulate the post-exploitation actions of advanced threat actors." Amongst many features, it has the capability to communicate with ICMP.

4.3. Integrated systems

Covert channels can also infiltrate integrated systems that were previously offline [24]. For example, systems in self-driving cars, or Internet of Things (IoT) devices, like home assistants or security cameras. These networks are based on the IP suite and are, therefore, vulnerable to ICMP tunneling, amongst other covert channels. Their cybersecurity is usually much more lax than that of more sophisticated systems. This is due to their widespread use allowing for greater possibilities of social engineering, and the hardware vulnerabilities arising from the lesser focus on security during their development [25]. Attacks against them include the threat of spying through cameras and microphones. More dangerous threats have the horrible potential of ending multiple peoples' lives if the attacker can gain unrestricted access to the mechanical controls of a car.

5. Prevention

There are preventative measures that limit the possibilities for communication using an ICMP tunnel [4]. Each one of them can be circumvented, or sometimes, a measure can disrupt the normal user [4]. A sensible combination of them can prevent all but the most sophisticated attacks.

The main prerequisite for ICMP tunneling is the variable data of ICMP Echo Request/Reply packets, both in content and in length. Operating systems have preset lengths and content for the ICMP Echo Request packets that they generate. For example, Linux has a standard data section length of 56B for ICMP Echo Requests, while Windows has 32B [5]. An IDS can filter against all ICMP packets that do not match the content of these two standards. Thus, the attacker needs to fragment the data into more ICMP packets to pass the filter. Sometimes, though, such a filter is disruptive. Big ICMP Echo Request packets are helpful to test whether a network is capable of carrying them [4]. Inspecting large packets for suspicious content is even more difficult, especially if the covert channel uses encryption. Determining if something is encrypted is not always a fail-safe method [4].

Another way to limit the communication over ICMP packets is to have stateful firewalls or NAT devices [4]. These track all ICMP traffic. If an ICMP Echo Request is sent, the identifier and sequence number are saved. Only a matching ICMP Echo Reply is let through to the original host. Other firewalls create their own ICMP packets that mirror the ones sent out by the host but with their own data. If they receive an ICMP Echo Reply, they then create an ICMP Echo Reply that matches the ICMP Echo Request sent by the host. This effectively disrupts the communication channel.

There also exists a proposal for a kernel module that scans ICMP packets for any malicious content [4]. This solution was proposed with the assumption that stateful firewalls are too resource-heavy to be implemented on personal machines. It was tested on 2003 hardware and had acceptable performance at the time. Nowadays, it should have very minimal overhead.

Another complex proposal is presented by Sayadi [5]. The detection includes two stages. In stage 1, the following three steps are performed on an ICMP packet: (a) the packet is checked for preset Linux and Windows lengths, (b) the method tries to match the packet against only one existing ICMP Echo Requests, (c) it is checked if there is an absence of a spike of ICMP messages. If all checks, performed after each other, succeed, then the message is regarded as normal. If either of the three points fails, then Stage 2 is triggered. Stage 2 tries to randomly pattern match against the standard Linux and Windows content. If it fails, the message is regarded as part of a covert channel.

The evolving field of machine learning can also be used to filter out covert channels in ICMP [26]. The proposal by Cho [27] is a promising new addition to existing tunneling prevention methods that use machine learning. Their algorithm has a 99.9% accuracy in detecting covert channels, an improvement over older proposals [28], [29].

All of the above methods have different approaches to dealing with a tunneled ICMP connection. The proposals on machine learning are promising, and their universality needs to be explored further [26]. A balanced approach of defensive methods needs to be taken, as at-scale implementation of machine learning results in many false positives [26].

6. Conclusion

ICMP tunneling is used for establishing covert channels on IP networks. Using fact that data of the ICMP Echo Request/Reply packets is not standardised in general, a tool can embed its own content into ICMP packets. An example application can embed TCP content, thus trasmiting TCP packets without the network administrators observing it. This connection can be multiplexed using fields of the ICMP Echo Request/Reply header and the TCP ports. The reliability of such a communication is comparable to normal TCP, with the packet rate being slightly slower and dependent on the rate of packet loss.

Since ICMP tunneling is used to hide communication on networks, a tunneled connection between two hosts results in a covert channel. Applications that make use of covert channels can use ICMP tunneling to achieve it. Some existing examples include botnets, like Pingback, and remote access backdoors, like Cobalt Strike.

There are proposed theoretical solutions to prevent ICMP tunneling, but practical applications are lacking. Many proposals exist that have been tested in simulated environments and perform very well. The field of machine learning also has a lot of promising research, which however still needs to be practically implemented on a wide scale. Further research of practical applications of these preventative measures and how they can be effectively combined together has to be conducted, to ensure that ICMP tunneling is not a threat to modern networks.

References

[1] D. D. Jovanović and P. V. Vuletić, "Analysis and Characterization of IoT Malware Command and Control Communication," in *2019 27th Telecommunications Forum (TELFOR)*, 2019, pp. 1–4.

[2] H. Dhayal and J. Kumar, "Botnet and P2P Botnet Detection Strategies: A Review," in *2018 International Conference on Communication and Signal Processing (ICCSP)*, 2018, pp. 1077–1082.

[3] J. Ford and H. S. Berry, "Leveling Up Survey of How Nation States Leverage Cyber Operations to Even the Playing Field," in *2023 11th International Symposium on Digital Forensics and Security (ISDFS)*, 2023, pp. 1–5.

[4] A. Singh, O. Nordström, C. Lu, and A. L. M. dos Santos, "Malicious ICMP Tunneling: Defense against the Vulnerability," in *Information Security and Privacy, 8th Australasian Conference, ACISP 2003*, 2003, pp. 226–236.

[5] S. Sayadi, T. Abbes, and A. Bouhoula, "Detection of Covert Channels Over ICMP Protocol," in *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 2017, pp. 1247–1252.

[6] J. Postel, "Internet Protocol," RFC 791, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc791

[7] J. Postel, "User Datagram Protocol," RFC 768, Aug. 1980. [Online]. Available: https://www.rfc-editor.org/info/rfc768

[8] W. Eddy, "Transmission Control Protocol (TCP)," RFC 9293, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9293

[9] J. Postel, "Internet Control Message Protocol," RFC 792, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc792

[10] "ping(8) - Linux man page," https://linux.die.net/man/8/ping, [Online; accessed 2024-03-02].

[11] "traceroute(8) - Linux man page," https://linux.die.net/man/8/traceroute, [Online; accessed 2024-03-02].

[12] K. Stokes, B. Yuan, D. Johnson, and P. Lutz, "ICMP Covert Channel Resiliency," in *Technological Developments in Networking, Education and Automation*, K. Elleithy, T. Sobh, M. Iskander, V. Kapila, M. A. Karim, and A. Mahmood, Eds. Dordrecht: Springer Netherlands, 2010, pp. 503–506.

[13] D. Stødle, "Ping Tunnel," https://www.cs.uit.no/~daniels/PingTunnel/, 2004, [Online; accessed 2023-12-17].

[14] "IEEE Standard for Ethernet," *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, pp. 1–7025, 2022.

[15] B. W. Lampson, "A Note on the Confinement Problem," *Commun. ACM*, vol. 16, no. 10, p. 613–615, oct 1973. [Online]. Available: https://doi-org.eaccess.tum.edu/10.1145/362375.362389

[16] J. Govil and J. Govil, "Criminology of BotNets and their detection and defense methods," in *2007 IEEE International Conference on Electro/Information Technology*, 2007, pp. 215–220.

[17] "Mirai BotNet Source Code," https://github.com/jgamblin/Mirai-Source-Code, 2016, [Online; accessed 2024-02-27].

[18] G. Gallopeni, B. Rodrigues, M. Franco, and B. Stiller, "A Practical Analysis on Mirai Botnet Traffic," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 667–668.

[19] P. Sinha, A. Boukhtouta, V. H. Belarde, and M. Debbabi, "Insights from the analysis of the Mariposa botnet," in *2010 Fifth International Conference on Risks and Security of Internet and Systems (CRiSIS)*, 2010, pp. 1–9.

[20] M. Singh, M. Singh, and S. Kaur, "TI-16 DNS Labeled Dataset for Detecting Botnets," *IEEE Access*, vol. 11, pp. 62 616–62 629, 2023.

[21] "List of Botnets," https://netacea.com/glossary/list-of-botnets/, 2021, [Online; accessed 2024-02-27].

[22] L. Macrohon and R. Mendrez, "Pingback: Backdoor At The End Of The ICMP Tunnel," https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/backdoor-at-the-end-of-the-icmp-tunnel/, 2021, [Online; accessed 2024-02-27].

[23] "Cobalt Strike," https://attack.mitre.org/versions/v11/software/S0154/, 2017, [Online; accessed 2024-02-27].

[24] A. Ondov and P. Helebrandt, "Covert Channel Detection Methods," in *2022 20th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 2022, pp. 491–496.

[25] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, "An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 250–10 276, 2020.

[26] Z. Sui, H. Shu, F. Kang, Y. Huang, and G. Huo, "A Comprehensive Review of Tunnel Detection on Multilayer Protocols: From Traditional to Machine Learning Approaches," *Applied Sciences*, vol. 13, no. 3, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/3/1974

[27] D. Cho, D. Thuong, and N. Dung, "A Method of Detecting Storage Based Network Steganography Using Machine Learning," *Procedia Computer Science*, vol. 154, pp. 543–548, 2019, proceedings of the 9th International Conference of Information and Communication Technology [ICICT-2019] Nanning, Guangxi, China January 11-13, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919308555

[28] A. N. Mahajan and I. Shaikh, "Detect covert channels in TCP/IP header using Naive Bayes," *International Journal of Computer Science and Mobile Computing*, vol. 4, pp. 881–883, 2015.

[29] T. Sohn, J. Seo, and J. Moon, "A study on the covert channel detection of TCP/IP header using support vector machine," in *Information and Communications Security: 5th International Conference, ICICS 2003, Huhehaote, China, October 10-13, 2003. Proceedings 5*. Springer, 2003, pp. 313–324.