electronics

MDPI

Article

Detecting Parallel Covert Data Transmission Channels in Video Conferencing Using Machine Learning

Ofir Joseph [1], Avshalom Elmalech [1] and Chen Hajaj [2,3,*]

[1] Information Science Department, Bar-Ilan University, Ramat Gan 5290002, Israel
[2] Department of Industrial Engineering and Management, Ariel University, Ariel 4076414, Israel
[3] Data Science and Artificial Intelligence Research Center, Ariel Cyber Innovation Center, Ariel University, Ariel 4076414, Israel
* Correspondence: chenha@ariel.ac.il; Tel.: +972-7472-33019

Abstract: Covert communication channels are a concept in which a policy-breaking method is used in order to covertly transmit data from inside an organization to an external or accessible point. VoIP and Video systems are exposed to such attacks on different layers, such as the underlying real-time transport protocol (RTP) which uses Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) packet streams to punch a hole through Network address translation (NAT). This paper presents different innovative attack methods utilizing covert communication and RTP channels to spread malware or to create a data leak channel between different organizations. The demonstrated attacks are based on a UDP punch hole created using Skype peer-to-peer video conferencing communication. The different attack methods were successfully able to transmit a small text file in an undetectable manner by observing the communication channel, and without causing interruption to the audio/video channels or creating a noticeable disturbance to the quality. While these attacks are hard to detect by the eye, we show that applying classical Machine Learning algorithms to detect these covert channels on statistical features sampled from the communication channel is effective for one type of attack.

Keywords: AI/ML for communication and networking; covert channels; hole punching; security; privacy and content protection; machine learning; cybersecurity

Citation: Joseph, O.; Elmalech, A.; Hajaj, C. Detecting Parallel Covert Data Transmission Channels in Video Conferencing using Machine Learning. *Electronics* **2023**, *12*, 1091. https://doi.org/10.3390/electronics12051091

Academic Editor: Harald Vranken

Received: 22 January 2023
Revised: 19 February 2023
Accepted: 21 February 2023
Published: 22 February 2023

1. Introduction

The COVID-19 pandemic has rapidly changed the landscape of certain professions and motivated many industries to adopt the concept of remote work [1,2]; even after the COVID-19 restrictions were removed, the usage of virtual meeting technologies such as Voice over Internet Protocol (VoIP) has become the new norm for several organizations after understanding its efficiency in day-to-day work as an alternative to physical meetings. This change in the work environment applies to the private and the government sectors and exposes critical information or infrastructure to a greater surface of attack [3–5]. This increased surface results from connecting the internal organization assets to a public domain, such as the Internet, or creating new connections between different organizations required for remote work.

Voice over IP (VoIP) and Video Conferencing (VC) systems, which enable the backbone of remote work methods, are simple to implement. Still, their vulnerability to attacks is greater than traditional telephone services. The audio and video services rely on "rich" communication protocols such as UDP and TCP [1,2], which allows for a substantial attack surface from the ability to traverse between NAT and Firewalls using punch holes, steganographic methods, or creating covert communication channels. Manifesting cyber-attacks on VoIP and VC systems allows the attacker to spread malware inside the attacked domain or leak sensitive data, which is a key challenge in connecting different organizations to the same communication medium [4]. One of the reasons for the increased vulnerability of

organizations to cyber-attacks on VoIP and VC systems is that the communication service used for voice or video channels is established by their client systems inside the organizational defense parameter in a way that is regarded as safe since a user authentication method was used to verify its creation.

A covert channel data transmission attack over real-time protocols (RTP) uses the existence of these communication services to covertly create a channel that is reliable and hard to detect using traditional methods without damaging or interrupting existing communication between the parties [6,7]. This challenge is more prominent when connecting closed organizations under different regulations or security guidance. Once any of these organizations have been compromised, meaning that attackers have gotten a foothold in a way that enables them to extract information [8] or run scripts, the communication channel used for VoIP or VC can enable them to expand their control to other closed off organization, making them valuable as well.

This type of Point-to-Point (P2P) architecture is being implemented in various sectors, from government to utilities, and is considered complex to defend and regulate. One common method of ensuring a safe connection between organizations is data filtering on the channel using a data-scheme filter. However, RTP channel filtering requires dedicated hardware and is not a commercial off-the-shelf product. This method of ensuring the security of the connection between the organizations may lead to costly in-house development and is effective to a certain degree. Another method to establish a cross-organization connection is to connect by extending WAN/LAN and the client infrastructure of one of the organizations to the other rather than directly between communication infrastructures, which will limit the accessibility to the end users by creating constraints on the availability of local termination in each of the different sites.

The different approaches to negate the threat of parallel covert transmission channels between organizations is described in Table 1.

Table 1. Secure communication methods comparison.

	Machine Learning	**Network Extension**	**Rule-Based Protection**	**Ad Hoc Filter**	**Combined ML and Filter Solution**
TCO	Low	High	High	High	High
Hardware	Not required	Required	Required	Required	Required
Compatibility	Agnostic	Required	Required	Required	Required
Custom Development	Not required	Not required	Not required	Required	Required
Detect parallel covert channels	Yes	Not Relevant	Unable	Unable	Yes

While Ad Hoc protocol filtering and Rule-Based solutions, such as firewalls, require dedicated hardware and compatibility with different VC software and hardware solutions, a Machine Learning based solution will have a dramatically lower cost of ownership and is agnostic to the VC solution chosen by the organization. The compatibility required and the rule-based nature of these detection systems will have, as stated before, a limited ability to detect parallel covert transmission channels which use a NAT punch hole to avoid a policy-breaking method of attack.

Using Machine Learning to detect is a growing method [9–11] in various fields and systems, becoming popular with the rise of distributed applications and systems such as the ones described in the Internet of Things paradigm and its variants [12].

The contribution of the proposed study is twofold. First, we demonstrate an innovative cyber-attack using covert communication on RTP channels, which rely on VoIP or VC applications for network address translation (NAT) traversal. Second, we discuss

why defending against this attack with common off-the-shelf tools is so challenging and suggest a simple-to-implement method using classical machine learning (ML) algorithms to detect this covert side-channel communication. The suggested detection method can be implemented easily and provide additional monitoring capabilities on a connection between organizations on either P2P architecture or one created using a third-party call manager and can improve the reliability of more complex detection systems which focus on anomalies in different layers.

The rest of the work is organized as follows: Section 2 details the related work; Section 3 describes the methodology and technical approach; Section 4 describes the simulation and results; the results are described in Section 5. Finally, Section 6 concludes this work.

2. Related Work

The literature review presented in this section is intended to provide a background for the core issues on which this work is based. The review includes a review of VOIP attacks; a review on detecting and preventing such attacks using ML models; an analysis of bypassing NAT and punch hole attacks; a review of NAT as a security function in communication networks; a review of circumventing NAT using UDP hole punching; and a review of the third-party host role in UDP hole punching.

2.1. VoIP and VC Systems Attacks

Telecommunication networks are susceptible to many threats, which are lower parts of the Open Systems Interconnection (OSI) model [5]. Among these threats are application layer threats, such as creating a steganographic RTP covert channel by embedding packets into an existing stream on a legitimate channel. Previous work demonstrating such an attack presented a new micro-protocol coupled with RTP-based steganographic embed data in the content [7]. Other threats, which illustrate the growing number of risks due to the open nature of telecommunication applications, include hijacking registration on the session layer, allowing an attacker to block incoming calls, redirect, replay or end calls at will. An example of such an attack can be found in previous work, which demonstrates a novel adaptive real-time model that tracks the changes in the system that acclimates to possible DDoS attacks [13]. Dupasquier et al. [8] describe an analysis of information leakage from encrypted Skype conversations by using the dynamic time warping (DTW) algorithm, frequently used in speech processing.

2.2. Detecting and Preventing Attacks Using ML

VoIP and VC systems attacks are an anomaly in nature; they create a small variance to the natural data spread between end devices; this misuse of the communication or the system medium can be correctly classified by using ML methods but is hard to detect using normal manual monitoring methods. Misuse of telecommunication networks can also be defined as unauthorized use; this attack method can be achieved by manipulating or attacking the Session Initiation Protocol (SIP), a signaling protocol used for initiating, maintaining, and terminating the communication session between the parties.

The threat of SIP-based attacks in a VoIP network was addressed by the work presented by Nazih et al. [3], which introduces an efficient detection of attacks using a linear l1-SVM Classifier. Previous work proposed a rule-based detection system effective against SIP and RTP-based attacks such as fake instant messaging and call hijacking [14]. In contrast, others proposed a method to distinguish normal and fraudulent behavior based on user profiles [15]. This concept of detecting fraudulent behavior was explored by [16] by using ML-based techniques based on users' statistical data.

Although using ML-based methods to detect attacks is not new, previous research in this field did not propose utilizing such methods for identifying the potential of creating a NAT Traversal covert transmission channel by using UDP or TCP punch holes created by the authenticated service itself.

2.3. NAT Bypassing and the Gap in the Reviewed Literature

The concept of bypassing the constraints of NAT by an attacker has been explored in previous works. Kai et al. [17] describe a NAT punching scheme in which a client sends dedicated punching packets to map the rules of the NAT. The work of Halkes [18] presents a study of the efficacy of UDP hole punching on the Internet in the context of point-to-point networks, similar to the architecture described in this work. The results showed that UDP hole punching is an effective method that most peers behind NAT or Firewalls allow hole punching to work, and that 80% hole punching attempts were successful. Choi et al. [19] propose a UDP hole punching method based on Time-To-Live (TTL) scheme in order to reduce the load on the call registrar for NAT traversal. This method was analyzed using a Markov chain model and significantly alleviated the load on the registrar.

Web Real-Time Communication (WebRTC) is a growing method of establishing communication between organizations; the work of Reddy [20] explores the potential of UDP hole punching to traverse between firewalls, which is considered problematic and may result in traffic block if not addressed by more sophisticated security and enforcement solutions. Similar concepts of traversing between Firewalls and in general have been discussed in existing literature for better performance and automation of manual configuration processes [21–25]. A private case of bypassing firewall regulation using UDP hole punching is also reviewed by Gbur et al. [26] work which employ a new protocol (QUIC) [27].

The QUIC protocol presents a new approach to combine encryption and transport layer stream into a single protocol in order to lower latency and improve security, exposing stateful firewalls to UDP hole-punching bypass attacks. This topic was further reviewed by Chatzoglou et al. [28], and others [29,30]. Furthermore, Ru et al. [6] describe the vulnerabilities existing in network protocols that can be used for malicious purposes. Hole punching is not limited to the on-premise compute and communication hubs; a work by Moyer [31] explores TCP hole punching between functions running on isolated containers in a cloud environment in order to achieve better performance and reduce costs for computing jobs. Similar concepts propose a combination of STUN protocols and UDP hole punching in peer-to-peer communication to reduce latency and delay [32]. Although UDP punch holes can reduce overall delay and latency, they can lead to misuse [33–35]. The method described in Muller et al. [36] proposes an autonomous way to NAT traversal, which establishes application connection to peers behind NAT by using fake ICMP signaling messages.

The key contribution of the research is the ability to establish a UDP punch hole without using a third-party host, which was assumed required to relay traffic or establish UDP states between the two parties. Although certain limitations are described as part of the theoretical approaches described in the work, the possibilities add to the attack surface and the users' responsibilities in establishing the communication channel.

Another method uses NAT Slip-streaming concepts, allowing an attacker to remotely access any TCP/UDP service bound to any system behind a victim's NAT [37]. Still, to the best of our knowledge, the usage of VoIP service to manifest this attack is yet to be thoroughly explored in terms of attack and defense. Although the different building blocks have been discussed in previous research works, to the best of our knowledge, a VC or VoIP application used to manifest a communication layer side channel attack and its correct identification by ML is yet to be thoroughly explored.

2.4. NAT as a Security Function in Communication Networks

NAT is a common tool in many networks today and is a basic feature of many Internet Service Providers (ISPs). The typical use of NAT is to interconnect a local network to an external public one, such as the Internet, NAT translates IP header information, substituting external (public) addresses for internal addresses in IP packets that need to transit the public network. NAT accomplishes this by providing either a static or dynamic external IP address. Network Address Translation is used as an Internet security measure by never using the sender's IP address for Internet access.

Network Address Translation technology was developed as a solution for the ever-increasing need for more IPv4 addresses. Specific ranges of IP addresses (described in RFC 1918 [38]) are designated as internal only, in other words, not route-able over the Internet. Anyone can use those addresses for private networks, reducing the number of public addresses that must be purchased. The basic function of NAT is to allow different components in the same local network environment to have a single global IP address. This function of having a single global IP address translates into two main advantages, one being connecting a large number of hosts to the global Internet using a smaller number of public (external) IP addresses, thereby conserving IP address space. The other advantage is enhancing security for private networks by keeping internal addressing private from the external network. However, certain complications are added when using NAT, such as the inability to determine the host destination of a packet from the peer connection because of the one global IP address. This may result in networks using NAT dropping the packet or labeling it as unauthorized.

2.5. Circumventing NAT Using UDP Hole Punching

UDP hole punching is a common technique for communication services to establish a UDP-based connection for systems protected behind NAT. The name originates from the function of the operation, which creates a hole in the network defense, such as firewalls or NATs to allow a packet from the external domain to meet the desired client inside the internal network. The process of the connection and its implications, resulting in the creation of the UDP punch hole, is described by Gianchandani [39].

WLOG, we assume system A and System B are connected. In contrast, system C acts as the third-party host responsible for connecting system A and system B communication applications. In chronological order:

1. System A and System B UDP packets are sent to the host (i.e., System C);
2. The UDP packets from System A and System B go through their respected NATs;
3. The NAT on each side rewrites the source IP address to the NAT's globally reachable IP address;
4. System C records the IP address and the ports of System A and B based on their requests;
5. According to the port mapping generated, System C directs the UDP packets to both System A and System B global IP addresses;
6. When the UDP packet from each system goes through the opposite system NAT, a note is made on the NAT itself;
7. The note allows the incoming packets to be accepted to the connection to be established between the two parties.

For an attack to succeed, some legitimate reason for initiating the communication between the parties has to be made. In our work, WLOG, we used the SKYPE connection to initiate the connection.

2.6. Third-Party Host Role in UDP Hole Punching

Two key methods exist to establish a UDP hole-punching connection, either with or without using a third-party host. The third-party host's role is to connect the clients behind the NAT; however, as a function of the application or the method used, this type of connection can be established without a third-party host, as shown in previous research such as in the work of Muller et al. [36].

A security analysis conducted by Silvanovich as part of Google Project Zero [40] examined the attack surface of Zoom, which uses a third-party server to establish multi-user conference calls. The usage of third-party servers for establishing calls, especially when used by a different control plane from the organizations conducting the call, can greatly benefit from security mechanisms such as authentication and end-to-end encryption, requiring the attacker access to the third-party server or a high level of knowledge usually available only through reverse engineering. While the Google team was unable to suc-

cessfully exploit the RTP channel vulnerabilities, they have described them as especially concerning and were still able to many elements of exploitation. Furthermore, the analysis has concluded that an attacker would be able to exploit them with sufficient investment.

On the other hand, Skype uses a different technique to establish communication services, allowing the UDP punch hole; it does not use a dedicated Skype-owned third-party host owned but rather uses the user appliance to act as the 3rd host. This method is commonly used in closed organizations that are not directly connected to external internet services but rely on them as a communication infrastructure to interconnect among themselves.

3. Methodology and Technical Approach

In this section, we introduce the methods by which a covert transmission channel can be established between two different NAT areas representing different organizations, as illustrated in Figure 1. This can be accomplished by a UDP punch hole created by a Skype service. The demonstrated attacks are able to create a parallel communication channel to the legitimate channel in which a file is transferred between the two parties separated by different NAT areas.

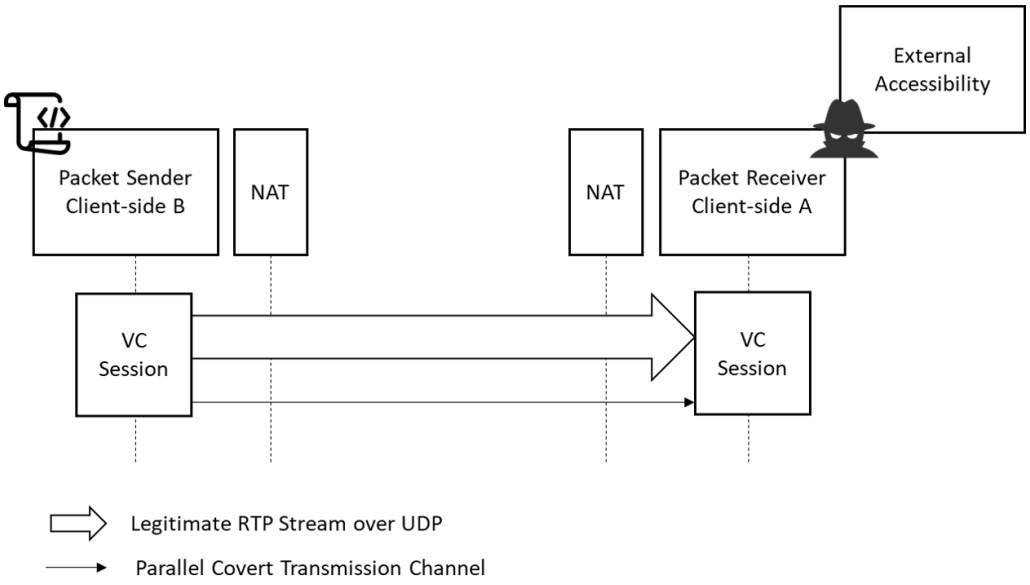

Figure 1. Graphical illustration: Parallel covert transmission channel in a point-to-point architecture.

For these attacks, we assume that the attacker has access to one of the clients and is able to plant a script on the other party. The attacker will try and emulate the legitimate channel features by learning them from the legitimate channel to avoid detection. We used the following methodology to simulate different attacks:

1. Learning the features using syntactic data of legitimate traffic features of the RTP protocol (UDP Packet), which are Length and Interval;
2. Applying statical distribution to the packet transmission interval feature using Gaussian spreading and Linear Interpolation;
3. Manually adjusting the packet length feature to the legitimate channel packet length range.

In order to detect the attack, we used classical ML methods over data collected from the communication ports between the parties to establish a reasonable accuracy if a covert communication channel exists.

3.1. Simulating the Attack

3.1.1. Channel Mimicking Using Gaussian Spreading

In our study, channel mimicking using Gaussian Spreading is used to transfer packets covertly on a side channel. The embedded script on client-side B needs to learn the characteristics of the RTP protocol by listening to the VC port; for the Transmission Interval, a random variable with a Gaussian distribution (i.e., normal deviation) is chosen between the values of the legitimate UDP packet lengths. For the Packet Length, a random float is chosen based on the length of the legitimate channel transmission. The transmission interval calculation is expressed through the following equation:

$$f(x) = a \cdot \exp\left(-\frac{(x-b)^2}{2c^2}\right)$$

where:

- x is the independent variable, representing the new point for the channel mimicking;
- a is the height of the curve;
- b is the position of the center of the peak;
- c is a measure of how wide the curve is, known as the standard deviation The exponent in the equation determines the shape of the curve, The standard deviation (c) determines the width of the curve, with larger values resulting in a wider curve and smaller values resulting in a narrower curve.

Once the side channel is established, based on a UDP punch hole created by the VC service, detecting it using network parameters alone is a challenging task, as can be seen in the 2D spread described in Figure 2. This method only requires the attacker to have a small amount of knowledge about the features of the legitimate channel (Packet Length and Transmission Interval), which are mostly generic per application.

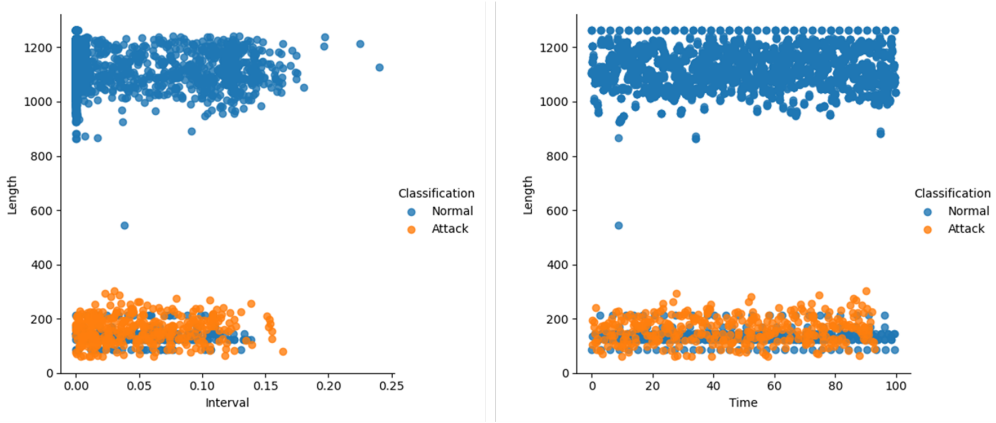

Figure 2. 2D scatter of a Gaussian Spreading based attack.

3.1.2. Channel Mimicking Using Linear Interpolation

A possible alternative is for the embedded script on client-side B to manually mimic the channel using linear interpolation on the transmission interval. The transmission interval is expressed through the following equation:

$$y = y_1 + ((x - x_1)/(x_2 - x_1)) \times (y_2 - y_1)$$

where:

- x is the point at which the function is being estimated, which represents the new point for the channel mimicking;
- x_1 and x_2 are known points on the function, with x_1 being less than x_2, which are the points of the legitimate channel;
- y_1 and y_2 are the values of the function at x_1 and x_2, respectively;

- y is the estimated value of the function at x. This equation works by taking the difference between x and x_1, and scaling it to the range between y_1 and y_2. The result is then added to y_1 to give the final estimate of the function's value at x.

The pseudo-code for the method used to mimic the channel using linear interpolation is as follows:

The attack algorithm for mimicking a channel using linear interpolation, described above in Algorithm 1, can be explained as follows:

1. This algorithm takes in two inputs: *CDFArray*, an array of pairs (X, Y) representing the points on the cumulative distribution function (CDF) curve, and *rand*, a random float between 0 and 1;
2. x_{min} and y_{min} are initialized to 0. These variables will later be used to store the value of X and Y from the previous iteration of the loop;
3. Similarly, x_{max} and y_{max} are initialized to 0. These variables will later be used to store the value of X and Y from the current loop iteration;
4. *last* is initialized to false. This variable will later be used to track whether the value of Y at the current iteration is greater than *rand*;
5. The loop iterates through the pairs in CDFArray. At each iteration, the values of X and Y are assigned to the loop variables X and Y, respectively;
6. If *last* is true, the loop breaks and x_{max} and y_{max} are set to the values of X and Y at the current iteration;
7. If Y is greater than *rand* and *last* is not true, *last* is set to true and x_{min} and y_{min} are set to the values of X and Y at the current iteration;
8. The loop ends;
9. The algorithm performs linear interpolation using the values of x_{min}, y_{min}, x_{max}, and y_{max} to find the value of the continuous random variable and returns the result.

Algorithm 1 Linear interpolation of channel mimicking

Inputs: $CDFArray, rand$
$x_{min} \leftarrow 0$
$y_{min} \leftarrow 0$
$x_{max} \leftarrow 0$
$y_{max} \leftarrow 0$
$last \leftarrow$ **false**
for X, Y in the $CDFArray$ **do**
    if *last* **then**:
        $x_{max} \leftarrow x$
        $y_{max} \leftarrow y$
        break
    end if
    if $y > rand$ and not *last* **then**
        $last \leftarrow$ **true**
        $x_{min} \leftarrow x$
        $y_{min} \leftarrow y$
    end if
end for
return $x_{min} + (x_{max} - x_{min}) * (rand - y_{min}) / (y_{max} - y_{min})$

Similarly to the previous attack method, the packet length of a random float is chosen based on the length of the legitimate channel transmission. The input for the Linear Interpolation function is the cumulative distribution function (CDF) of the packet sizes can be calculated from the WIRESHARK capture as depicted in Figure 3.

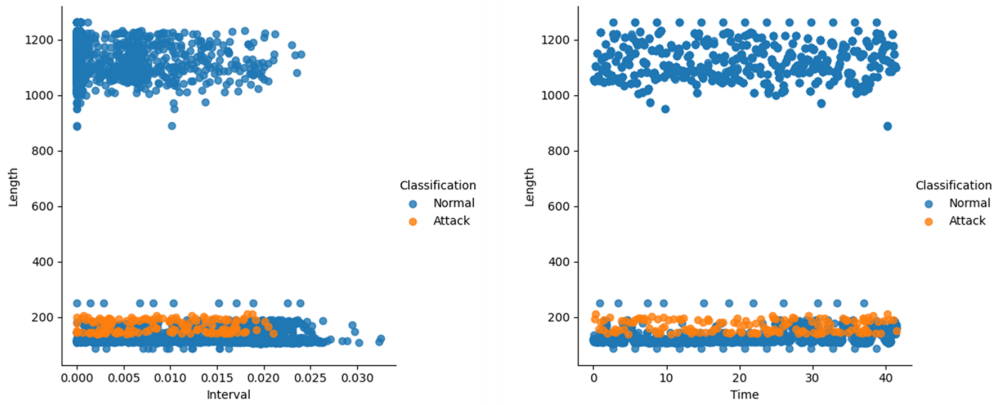

Figure 3. 2D scatter of Linear Interpolation based attack.

3.1.3. Larger Packet Length Reduction for Analysis

As can be seen in Figures 2 and 3, the packet spreading occurs in two major clusters. Large packets with a length greater than 500 and smaller packets with a length lower than 300 (Note that packet length determines the size of the whole packet, including the header, trailer, and the data sent on that packet). This cluster difference results from different traffic types, such as audio and video. For the channel mimicking, we assumed the natural existence of the small packets and ignored the possibility of the larger packet cluster existing naturally in the VC conversation. Since the attacker only targets the small packets, and to better analyze the effect of machine learning analysis of the attack, we have artificially removed the larger packet clusters as can be seen in Figures 4 and 5.

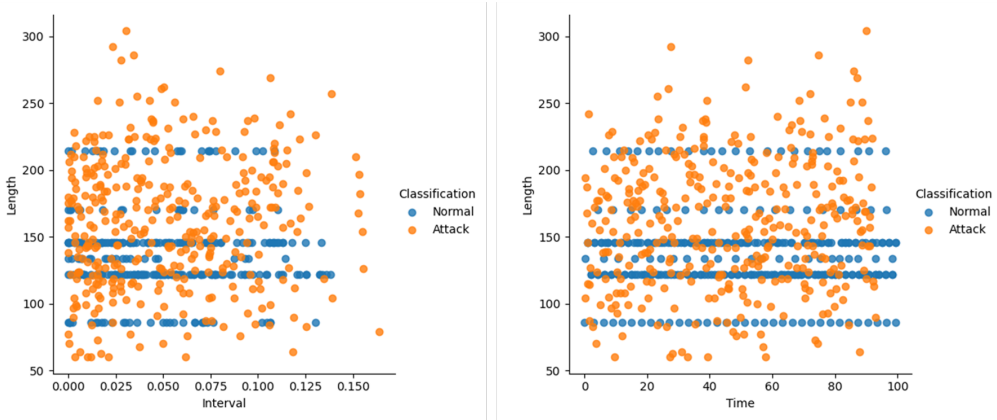

Figure 4. 2D scatter of Gaussian Spreading based attack with only small packets.

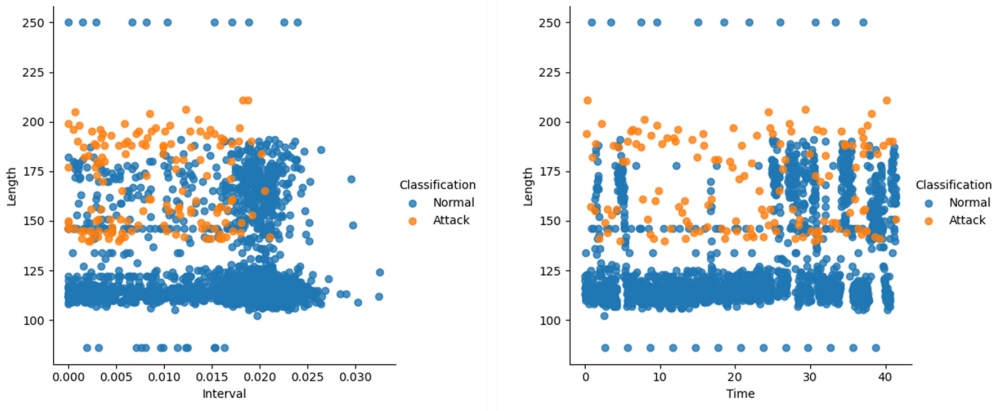

Figure 5. 2D scatter of Linear Interpolation Spreading based attack with only small packets.

As can be seen in Figure 4, the Gaussian Spreading algorithm spread is not as tight as the Linear Interpolation spread seen in Figure 5 and would be more likely to be spotted by a security operations center (SOC) operative in reference to a normal traffic 2D scatter pattern.

4. Experimental Design

A set of experiments was conducted to evaluate the effects of the evasion attacks suggested in Section 3. For clarity purposes, we divide this section as follows: first, in Section 4.1, we describe how we empirically generated the datasets used for our experiments. Next, in Section 4.2, we describe the architecture used for our evaluations, list the different ML models used, and discuss the way we ensured robust and unbiased learning. Finally, in Section 4.3, we present the different evaluation metrics based on which we evaluate the ability of classic ML classification models to detect an attack.

4.1. Data Collection

Data were collected using WIRESHARK on client-side A, which received the data from client-side B. The PCAP file representing this data contained the following features—timestamp, length, source and destination IP, and TTL. Using the method mentioned in Section 3.1, we mimicked the packet length using either Gaussian Spread or Linear Interpolation and manually adjusted the transmission interval of the covertly transmitted packets. In addition, we used the time stamp to create an interval feature between the packets, which was further enhanced by replicating it to create a time series (sliding window). The Gaussian Spreading data set contained 4544 packets, of which 354 were covertly transmitted (6%). The Linear Interpolation data set contained 4631 packets, of which 142 were covertly transmitted (3%). Detailed information about the different datasets is described in Table 2.

Table 2. Detailed datasets description.

Dataset	Linear Interpolation	Linear Interpolation with Only Small Packets	Gaussian Spreading	Gaussian Spreading with Only Small Packets
Normal	4489	2203	4190	307
Covert	142	142	344	355
Covert/Total	3	6	7	53.6

4.2. Experimental Methodology

The network architecture of the experiment we planned simulated a connection between two different organizations, separated by different internal security regulations and employ network protection such as NAT; this is illustrated in Figure 6. The communication connection between the two parties is established over a public domain infrastructure utilizing two different Internet Service Providers, which employ, to some degree, network protection to defend against unauthorized access to hosts from unauthorized clients or to avoid port forwarding attacks. After setting up the connection, we used Skype, which uses a peer-to-peer connection, to punch-through NAT of the different parties.

The connection demonstrated in the experiment simulates the general underlying technical mechanisms used in all VC systems. The Skype channel creates the UDP-based RTP channel, which allows the NAT punch-through and, in addition, acts as the SIP server.

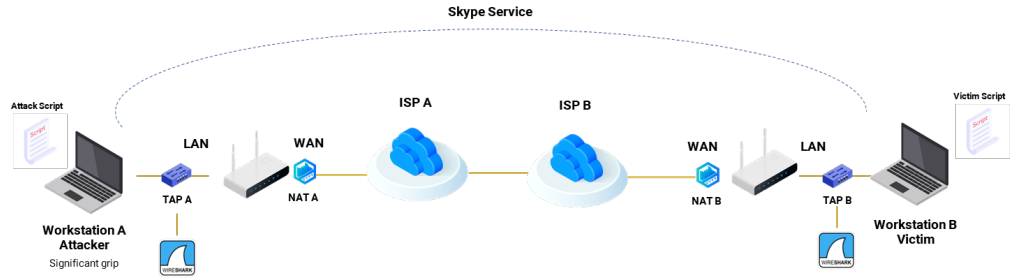

Figure 6. Network architecture of the simulated lab setup; Workstation A and Workstation B act as different organizations separated by different ISPs and NATs. TAP A and TAP B helped monitor traffic as the capture point for the datasets.

After establishing the authorized connection and the parallel data leak channel based on the NAT punch hole, we designed two evaluations. The evaluations, which were purposed to emulate the features of the authorized channel in parallel to the data leak channel and avoid detection by ML tools, were based on two different attacks described in Section 3.1.

Our evaluation was for the Gaussian spreading dataset and the Linear Interpolation. Each evaluation was conducted using these eight classical ML models (XGBoost, AdaBoost, Random Forest, Gradient Boosting, K-nn, and Decision Tree). To account for variations, the experiments were executed using 10-fold cross-validation in a stratified way, such that the portion of malicious instances was identical on each fold's test set.

4.3. Evaluation Metrics

To evaluate the models' performances, we considered well-known metrics commonly used in the literature (accuracy, precision, recall, and F1-score). Note that, in this paper, we tackle imbalanced datasets where accuracy is insufficient. Thus, we also look at the precision, recall, and *F1-score*. The formal definitions of the above metrics are as follows:
Accuracy: the fraction of the classification samples correctly classified:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall: the total number of True Positives (TP) among all actual positive samples ($TP + FN$):

$$Recall = \frac{TP}{TP + FN}$$

Precision: The ratio of True Positive (TP) samples of the total classifications that were positive ($TP + FP$):

$$Precision = \frac{TP}{TP + FP}$$

F1-score: a measure of a model's accuracy on a dataset. It evaluates the binary classification systems, which classify samples as positive or negative. The *F1-score* combines the Precision and Recall into one metric for the model's performance according to the harmonic mean of the model's Precision and Recall:

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

5. Experimental Results

The data set used for the algorithm comparison was based on the channel mimicking using Gaussian spreading and Linear Interpolation, as described in Section 3.1. Results were obtained using the scikit-learn 1.2.0 API, which offers a variety of machine-learning packages. The results of the different models' performance are reported for Linear

Interpolation-based attack in Tables 3 and 4 and for Gaussian Spreading-based attack in Tables 5 and 6.

Table 3. Models' performance for Linear Interpolation (standard deviation in brackets).

Classifier	Accuracy	Precision	Recall	F1-Score
XGBoost	0.98 (0)	0.8 (0.12)	0.63 (0.14)	0.7 (0.12)
AdaBoost	0.98 (0)	0.82 (0.13)	0.67 (0.14)	0.73 (0.12)
RandomForest	0.98 (0)	0.86 (0.13)	0.56 (0.14)	0.67 (0.14)
GradientBoosting	0.98 (0)	0.82 (0.12)	0.62 (0.15)	0.7 (0.12)
KNeighbors	0.97 (0)	0.62 (0.12)	0.42 (0.08)	0.49 (0.07)
DecisionTree	0.98 (0)	0.73 (0.06)	0.7 (0.09)	0.71 (0.05)

Table 4. Models' performance for Linear Interpolation with only small packets (standard deviation in brackets).

Classifier	Accuracy	Precision	Recall	F1-Score
XGBoost	0.97 (0)	0.78 (0.11)	0.68 (0.13)	0.71 (0.09)
AdaBoost	0.97 (0)	0.84 (0.11)	0.68 (0.1)	0.75 (0.09)
RandomForest	0.97 (0.01)	0.85 (0.11)	0.57 (0.12)	0.68 (0.11)
GradientBoosting	0.97 (0.01)	0.83 (0.12)	0.62 (0.14)	0.7 (0.11)
KNeighbors	0.94 (0.01)	0.59 (0.12)	0.41 (0.11)	0.47 (0.09)
DecisionTree	0.96 (0.01)	0.73 (0.12)	0.61 (0.8)	0.66 (0.08)

Table 5. Models' performance for Gaussian Spreading (standard deviation in brackets).

Classifier	Accuracy	Precision	Recall	F1-Score
XGBoost	0.99 (0)	0.97 (0.03)	0.95 (0.02)	0.96 (0.02)
AdaBoost	0.99 (0)	0.98 (0.02)	0.94 (0.02)	0.96 (0.02)
RandomForest	0.98 (0)	0.88 (0.04)	0.83 (0.05)	0.85 (0.03)
GradientBoosting	1 (0)	1 (0.01)	0.95 (0.03)	0.97 (0.01)
KNeighbors	0.97 (0.01)	0.82 (0.06)	0.7 (0.07)	0.76 (0.05)
DecisionTree	0.99 (0.07)	0.93 (0.05)	0.95 (0.02)	0.94 (0.04)

Table 6. Models' performance for Gaussian Spreading with only small packets (standard deviation in brackets).

Classifier	Accuracy	Precision	Recall	F1-Score
XGBoost	0.96 (0.02)	0.98 (0.03)	0.95 (0.02)	0.97 (0.02)
AdaBoost	0.96 (0.01)	0.99 (0.01)	0.94 (0.03)	0.97 (0.01)
RandomForest	0.89 (0.03)	0.91 (0.05)	0.89 (0.03)	0.9 (0.03)
GradientBoosting	0.97 (0.02)	0.99 (0.01)	0.94 (0.03)	0.97 (0.01)
KNeighbors	0.76 (0.05)	0.84 (0.07)	0.69 (0.04)	0.76 (0.04)
DecisionTree	0.94 (0.01)	0.95 (0.03)	0.94 (0.03)	0.95 (0.01)

The reduction of the larger packets had a small effect in terms of precision and recall, in which the results for smaller packet sizes were more precise and had a better recall and F1 score in some of the algorithms. Furthermore, one can see that the standard deviation is much higher for the evaluations of the linear interpolation attack, showing that these attack instances are more heterogeneous than the ones generated by the Gaussian Spreading based attack. The combination of the mean and standard deviation allows one to extract further statistical features of each model, such as confidence interval and paired *t*-test using statistical tools such as [41].

We decided to apply a balanced class weight to the models. Balanced class weight is a technique used to manipulate the error weight in each class by a factor opposite to the ratio of the minority class in the dataset. As a result, using a balanced class weight

technique helps improve the model's performance on the minority class (which is mostly the more important one) and prevents it from being biased towards the more frequent class. The results for the balanced weight are reported for Linear Interpolation based attack in Tables 7 and 8 and for Gaussian Spreading based attack in Tables 9 and 10.

Table 7. Models' performance for Linear Interpolation after applying class weight (standard deviation in brackets).

Classifier	Accuracy	Precision	Recall	F1-Score
XGBoost	0.98 (0)	0.72 (0.12)	0.74 (0.14)	0.72 (0.12)
AdaBoost	0.98 (0)	0.82 (0.13)	0.67 (0.14)	0.73 (0.12)
RandomForest	0.98 (0)	0.84 (0.17)	0.49 (0.15)	0.61 (0.16)
GradientBoosting	0.98 (0)	0.82 (0.12)	0.61 (0.14)	0.69 (0.12)
KNeighbors	0.97 (0)	0.62 (0.01)	0.42 (0)	0.49 (0.02)
DecisionTree	0.98 (0)	0.71 (0.13)	0.61 (0.1)	0.65 (0.09)

Table 8. Models' performance for Linear Interpolation with only small packets after applying class weight (standard deviation in brackets).

Classifier	Accuracy	Precision	Recall	F1-Score
XGBoost	0.97 (0.01)	0.72 (0.09)	0.75 (0.13)	0.73 (0.09)
AdaBoost	0.97 (0.01)	0.84 (0.11)	0.68 (0.1)	0.75 (0.09)
RandomForest	0.96 (0.01)	0.83 (0.12)	0.51 (0.08)	0.63 (0.09)
GradientBoosting	0.97 (0.01)	0.83 (0.13)	0.62 (0.14)	0.7 (0.11)
KNeighbors	0.94 (0.01)	0.59 (0.12)	0.41 (0.11)	0.47 (0.09)
DecisionTree	0.96 (0.01)	0.71 (0.05)	0.6 (0.11)	0.64 (0.08)

Table 9. Models' performance for Gaussian Spreading after applying class weight (standard deviation in brackets).

Classifier	Accuracy	Precision	Recall	F1-Score
XGBoost	0.99 (0)	0.93 (0.04)	0.95 (0.02)	0.94 (0.04)
AdaBoost	0.99 (0)	0.98 (0.02)	0.94 (0.03)	0.96 (0.02)
RandomForest	0.98 (0)	0.89 (0.05)	0.83 (0.04)	0.85 (0.04)
GradientBoosting	1 (0)	1 (0)	0.95 (0.02)	0.97 (0.01)
KNeighbors	0.97 (0)	0.82 (0.06)	0.7 (0.07)	0.76 (0.05)
DecisionTree	0.99 (0)	0.95 (0.04)	0.93 (0.03)	0.94 (0.03)

Table 10. Models' performance for Gaussian Spreading with only small packets after applying class weight (standard deviation in brackets).

Classifier	Accuracy	Precision	Recall	F1-Score
XGBoost	0.96 (0.02)	0.98 (0.04)	0.95 (0.02)	0.97 (0.02)
AdaBoost	0.96 (0.01)	0.99 (0.01)	0.94 (0.03)	0.97 (0.01)
RandomForest	0.88 (0.04)	0.9 (0.06)	0.88 (0.04)	0.89 (0.04)
GradientBoosting	0.97 (0.01)	0.99 (0.01)	0.94 (0.03)	0.97 (0.01)
KNeighbors	0.76 (0.05)	0.84 (0.07)	0.69 (0.05)	0.76 (0.04)
DecisionTree	0.94 (0.01)	0.95 (0.03)	0.94 (0.03)	0.95 (0.01)

Using a balanced class weight technique had little to no effect on the overall performance of the different models. In terms of attack, it seems that, in correlation to the 2D scattering in Figures 4 and 5, detecting channel mimicking attacks which are based on Linear Interpolation is more arduous due to the lower accuracy, recall, and precision of the models on the datasets in comparison to the channel mimicking attacks based on Gaussian Spreading.

Most of the algorithms used were able to reach high and near-perfect accuracy.

AdaBoostClassifier has achieved a precision score of 0.98 on the Gaussian Spreading data set and 0.82 on the Linear Interpolation dataset, and a high F1 score, meaning that the number of false alarms is very small. Since the attack packets are being transmitted over a period of time in the covert transmission channel, the lower detection rate will be, to some degree, negated by the number of transmitted attack packets. This argument suggests that the AdaBoost algorithm is more favorable to SOC operation. Still, an ad hoc analysis is required for each organization's goals in employing a detection system in general.

6. Conclusions and Future Work

The malicious use of a parallel covert communication channel may cause severe damage to traditionally closed organizations, for example, in the defense sector. Data leaks can risk the integrity of plans and data; in addition, the existence of such a channel may allow the surface to conduct service denial attacks via malware spreading. This work has supplied a proven method for detecting parallel covert communication channels using legitimate communication channels punch holes to transmit data between two NAT-separated organizations. Although connecting to a third-party server can mitigate the risk, it is not common in the government sector due to the organizational blind spot it creates, in addition to having its own vulnerabilities to punch hole attacks, making the assessment technocratically valid in various architectures. Although identifying cybersecurity risks in VC or VoIP systems is thoroughly examined, parallel communication channels are mostly overlooked and can be exploited easily; simple and basic ML methods can effectively detect them and at least pose more constraints on an attacker.

The method presented in this work can reduce the attack surface of covert transmission channels using video conferencing communication to leak data or spread malware between different internet-closed organizations or cloud-based solutions towards an on-premise data center environment. However, to better defend the organizational border from embedded attacks or covert transmission, a more holistic approach is required for real-time applications, which often use rich protocols such as UDP and complex data schemes such as video or audio.

Figure 7 illustrates a connection architecture for different organizations not under the same regulation. This particular architecture, in which two or more organizations are connected, poses a threat to each organization involved.

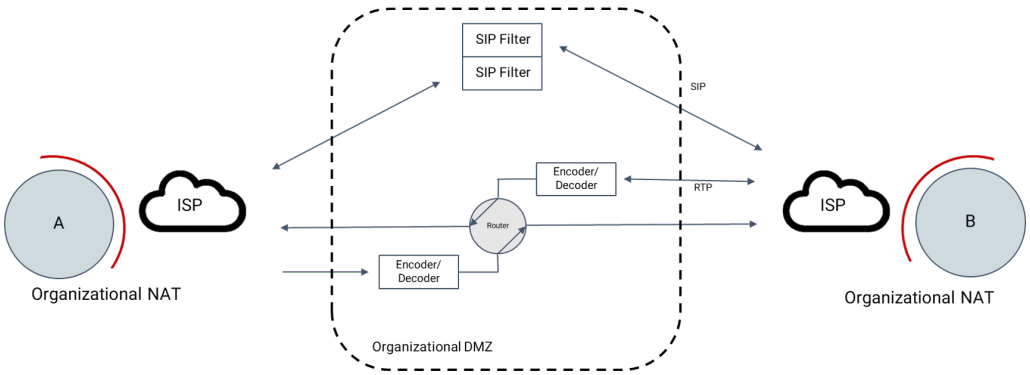

Figure 7. Graphical illustration: Holistic Approach for RTP connection between organizations connected by a demilitarized zone (DMZ).

The holistic approach we offer employs anomaly detection and data scheme filtering for the SIP channel, as well as encrypting RTP channel between the parties. This will, in practice, create more constraints for the attack in creating unauthorized or parallel data leak and spreading channels; these methods, in addition to existing security aspects, already consider basic "off the shelf" tools such as NAT, Firewalls, and Line Encryption. For embedded attacks, a more comprehensive solution will be required, such as video and audio encoding and decoding; this is an ad hoc solution for more threatened organizations

and may affect user experience, system performance, and the overall cost of ownership for the solution.

The trade-off between the possible protection level based on the architecture should be correlated to a threat survey conducted by the organization. This survey should reflect the possible risk of creating a connection to different organizations. As previously mentioned, the solution can be based on the simple employment of software solutions such as machine learning for anomaly detection to a third-party demilitarized zone sub-network to manage the connection between the different parties.

7. Limitations

The work presented in this paper described the usage of a parallel covert transmission channel by transmitting a small text file between the endpoints. This demonstration described the general vector of attack using the described method of parallel transmission. The attack has two key limitations, the first being content filtering. In our demonstration, we have transmitted the file as clear text between the endpoints; this method of attack is susceptible to detection, especially when transmitting text files unless the attack encrypts the file before transmission, which requires some encryption algorithm on the end devices. We assume this is of little difficulty and in the realm of possibilities when manifesting the suggested attack. The second limitation is the call duration for transmitting larger files; in order for the transmission to remain covert, it has to be in parallel to an actual legitimate transmission, which in our case is an actual call between two users. In the case of larger files, some form of buffering mechanism will have to be used; this will allow for continuing the transmission on different calls and remain under the assumptions of the channel mimicking.

Even when improving the attack and overcoming these limitations, the statistical features of the channel, i.e., packet length and transmission interval, will remain under the same key assumptions we have described in this work and are still exposed to being detected by the machine learning algorithms we have suggested.

Author Contributions: Conceptualization, O.J.; Methodology, C.H.; Software, O.J. and C.H.; Validation, A.E.; Formal analysis, A.E.; Investigation, O.J.; Resources, A.E.; Data curation, C.H.; Writing—original draft, O.J., A.E. and C.H.; Visualization, O.J.; Supervision, A.E. and C.H.; Project administration, A.E. and C.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We thank Ron Posti (Ben Gurion University of the Negev) and Sean Galantzan (Tel Aviv University) for their assistance with setting up the lab environment and participation in the tests conducted as part of this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Brynjolfsson, E.; Horton, J.J.; Ozimek, A.; Rock, D.; Sharma, G.; TuYe, H.Y. *COVID-19 and Remote Work: An Early Look at US Data*; Technical Report; National Bureau of Economic Research: Cambridge, MA, USA, 2020.
2. Marshall, D.T.; Shannon, D.M.; Love, S.M. How teachers experienced the COVID-19 transition to remote instruction. *Phi Delta Kappan* **2020**, *102*, 46–50. [CrossRef]
3. Nazih, W.; Hifny, Y.; Elkilani, W.; Abdelkader, T.; Faheem, H. Efficient detection of attacks in SIP based VoIP networks using linear L1-SVM classifier. *Int. J. Comput. Commun. Control.* **2019**, *14*, 518–529. [CrossRef]
4. Okereafor, K.; Manny, P. Understanding cybersecurity challenges of telecommuting and video conferencing applications in the COVID-19 pandemic. *Int. J. Eng. (IJITE)* **2020**, *8*, 6 .
5. Naeem, M.M.; Hussain, I.; Missen, M.M.S. A survey on registration hijacking attack consequences and protection for Session Initiation Protocol (SIP). *Comput. Netw.* **2020**, *175*, 107250. [CrossRef]

6. Ru, K.; Zheng, Y.; Feng, X.; Wang, D. The Side-Channel Vulnerability in Network Protocol. In Proceedings of the 2021 the 11th International Conference on Communication and Network Security, Weihai, China, 3–5 December 2021; pp. 1–8.
7. Azadmanesh, M.; Mahdavi, M.; Shahgholi Ghahfarokhi, B. A reliable and efficient micro-protocol for data transmission over an RTP-based covert channel. *Multimed. Syst.* **2020**, *26*, 173–190. [CrossRef]
8. Dupasquier, B.; Burschka, S.; McLaughlin, K.; Sezer, S. Analysis of information leakage from encrypted Skype conversations. *Int. J. Inf. Secur.* **2010**, *9*, 313–325. [CrossRef]
9. Andoni, M.; Robu, V.; Flynn, D.; Abram, S.; Geach, D.; Jenkins, D.; McCallum, P.; Peacock, A. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renew. Sustain. Energy Rev.* **2019**, *100*, 143–174. [CrossRef]
10. Asharf, J.; Moustafa, N.; Khurshid, H.; Debie, E.; Haider, W.; Wahab, A. A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions. *Electronics* **2020**, *9*, 1177. [CrossRef]
11. Abu Al-Haija, Q.; Krichen, M.; Abu Elhaija, W. Machine-learning-based darknet traffic detection system for IoT applications. *Electronics* **2022**, *11*, 556. [CrossRef]
12. Heidari, A.; Navimipour, N.J.; Unal, M. A Secure Intrusion Detection Platform Using Blockchain and Radial Basis Function Neural Networks for Internet of Drones. *IEEE Internet Things J.* **2023**, 1 . [CrossRef]
13. Semerci, M.; Cemgil, A.T.; Sankur, B. An intelligent cyber security system against DDoS attacks in SIP networks. *Comput. Netw.* **2018**, *136*, 137–154. [CrossRef]
14. Wu, Y.S.; Bagchi, S.; Garg, S.; Singh, N. Scidive: A stateful and cross protocol intrusion detection architecture for voice-over-ip environments. In Proceedings of the International Conference on Dependable Systems and Networks, Florence, Italy, 28 June–1 July 2004; IEEE: New York, NY, USA, 2004; pp. 433–442.
15. Olszewski, D. A probabilistic approach to fraud detection in telecommunications. *Knowl.-Based Syst.* **2012**, *26*, 246–258. [CrossRef]
16. Kilinc, H.H. A case study on fraudulent user behaviors in the telecommunication network. *Electrica* **2021**, *21*, 74–84. [CrossRef]
17. Lin, K.; Jia, C. A punching scheme for crossing NAT in end hopping. *Wuhan Univ. J. Nat. Sci.* **2012**, *17*, 539–543. [CrossRef]
18. Halkes, G.; Pouwelse, J. UDP NAT and Firewall Puncturing in the Wild. In *Proceedings of the International Conference on Research in Networking*; Springer: Cham, Switzerland, 2011; pp. 1–12.
19. Choi, K.; Kong, K.S.; Chung, K.S.; Park, D.S.; Gil, J.M. TTL-Based UDP Hole Punching Scheme in SIP Network. In *Multimedia and Ubiquitous Engineering*; Springer: Cham, Switzerland, 2014; pp. 147–152.
20. Reddy, T.; Patil, P.; Wing, D.; Ver Steeg, B. Webrtc udp firewall traversal. In Proceedings of the IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI), Zurich, Switzerland, 26–27 January 2015.
21. Gruenter, E.; Meier, M.; Niederberger, R.; Petri, F. Dynamic *Configuration of Firewalls Using UDP Hole Punching*; Technical Report; D-Grid Integrationsbericht Fachgebiet: Julich, Germany, 2006; pp. 3–5.
22. Ganguly, A.; Boykin, P.O.; Figueiredo, R. Techniques for low-latency proxy selection in wide-area P2P networks. In Proceedings of the 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), Atlanta, GA, USA, 19–23 April 2010; IEEE: New York, NY, USA, 2010; pp. 1–8.
23. Doğan, F.T. High Throughput Udp-Based Peer-To-Peer Secure Data Transfer. PhD Thesis, Bilkent Universitesi, Ankara, Turkey, 2018.
24. Prasanna, R.; Chandrakumar, C.; Nandana, R.; Holden, C.; Punchihewa, A.; Becker, J.S.; Jeong, S.; Liyanage, N.; Ravishan, D.; Sampath, R.; et al. "Saving Precious Seconds"—A Novel Approach to Implementing a Low-Cost Earthquake Early Warning System with Node-Level Detection and Alert Generation. *Informatics* **2022**, *9*, 25. [CrossRef]
25. Oistrez, T.; Grünter, E.; Meier, M.; Niederberger, R. A reliable and fast data transfer for grid systems using a dynamic firewall configuration. In *Proceedings of the European Conference on Parallel Processing*; Springer: Cham, Switzerland, 2008; pp. 94–102.
26. Gbur, K.Y.; Tschorsch, F. A QUIC (K) Way Through Your Firewall? *arXiv* **2021**, arXiv:2107.05939.
27. Iyengar, J.; Thomson, M. QUIC: A UDP-based multiplexed and secure transport. In *RFC 9000*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2021.
28. Chatzoglou, E.; Kouliaridis, V.; Karopoulos, G.; Kambourakis, G. Revisiting QUIC attacks: A comprehensive review on QUIC security and a hands-on study. *International J. Inf. Secur.* **2022**, 1–19. [CrossRef]
29. Joarder, Y.; Fung, C. A Survey on the Security Issues of QUIC. In Proceedings of the 2022 6th Cyber Security in Networking Conference (CSNet), Rio de Janeiro, Brazil, 24–26 October 2022; IEEE: New York, NY, USA, 2022; pp. 1–8.
30. Soni, M.; Rajput, B.S. Security and performance evaluations of QUIC protocol. In *Data Science and Intelligent Applications*; Springer: Cham, Switzerland, 2021; pp. 457–462.
31. Moyer, D.W. Punching Holes in the Cloud: Direct Communication between Serverless Functions Using NAT Traversal. Ph.D. Thesis, Virginia Tech, Blacksburg, VA, USA, 2021.
32. Thu, H.T.T.; Park, J.; Won, Y.; Kim, J. Combining stun protocol and udp hole punching technique for peer-to-peer communication across network address translation. In Proceedings of the 2014 International Conference on IT Convergence and Security (ICITCS), Beijing, China, 28–30 October 2014; IEEE: New York, NY, USA, 2014, pp. 1–4.
33. Liu, G.; Liu, D.; Hao, S.; Gao, X.; Sun, K.; Wang, H. Ready Raider One: Exploring the Misuse of Cloud Gaming Services. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Copenhagen, Denmark, 26–30 November 2022; pp. 1993–2007.
34. Flaagan, T. Traversing NAT: A Problem. Master's Thesis, Dakota State University, Madison, SD, USA, 2021.

35. Duarte, E.P., Jr.; Cardoso, K.V.; de Mello, M.O.; Borges, J.G. Beware: NAT Traversal is a Simple and Efficient Approach to Open Firewall Holes. *Abakós* **2020**, *8*, 29–41. [CrossRef]
36. Muller, A.; Evans, N.; Grothoff, C.; Kamkar, S. Autonomous nat traversal. In Proceedings of the 2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P), Delft, The Netherlands, 25–27 August 2010; IEEE: New York, NY, USA, 2010; pp. 1–4.
37. NAT Slipstreaming v2.0. Available online: https://samy.pl/slipstream/ (accessed on 21 January 2023).
38. Rekhter, Y.; Moskowitz, B.; Karrenberg, D.; Groot, G.d.; Lear, E. *Rfc1918: Address Allocation for Private Internets*; Silicon Graphics, Inc.: Mountain View, CA, USA, 1996.
39. Circumventing NAT Using UDP Hole Punching. Available online: https://highaltitudehacks.com/2013/06/13/circumventing-nat-using-udp-hole-punching/ (acceseed on21 January 2023).
40. Zooming in on Zero-click Exploits. Available online: https://googleprojectzero.blogspot.com/2022/01/zooming-in-on-zero-click-exploits.html (acceseed on 21 January 2023).
41. Rodríguez-Fdez, I.; Canosa, A.; Mucientes, M.; Bugarín, A. STAC: A web platform for the comparison of algorithms using statistical tests. In Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Istanbul, Turkey, 2–5 August 2015.