VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

COMPUTER MODELING STUDY PROGRAMME

Master's thesis

Research of methods for covert channel communication identification

Nedeklaruotų komunikacinių kanalų identifikavimo metodų tyrimas

Haroldas Jomantas

Supervisor : assoc. prof. dr. Linas Bukauskas

Vilnius
2025

Summary

Covert communication channels are a covert way of transmitting information using computer network communications. Inspired by the methods used by military strategists in antiquity to secretly transmit messages, these communication channels pretend to carry out normal communication between devices, but insert secret information in redundant or empty spaces in the packets sent. This technique can also manipulate the timing of the sending of such packets. This type of covert communication is most commonly used for data exfiltration, but in practice it can also be used for other functions, such as the covert transmission of commands to an infected device, or as a legitimate means of communicating between devices to prevent the eavesdropper of network traffic from reading the information.

This work focuses on the detection of such communication channels. A covert communication channel, has been developed and which is not yet described in other papers, exploits the normal functions of the IS-IS protocol of a router to transmit sensitive information from one security zone to a less secured zone. After implementing such an undeclared communication channel and testing it in practice, a proposal for detecting and terminating such a communication channel has been presented in this paper.

The work continues with the study of identification methods, focusing on machine learning and neural network models, training them, evaluating their performance and accuracy in identifying undeclared communication channels.

For the identification of covert communication, the best performing machine learning model was identified and showed very good results.However, it is stressed that in order for such methods to work in real systems and not in laboratory conditions, it is necessary to ensure a high variation of the data and to work systematically on the training of such models.

A convolutional neural network was developed to identify covert communication and performed extremely well. The experiments demonstrated the superiority of this type of models over traditional machine learning models and contributed to future research on the accurate and fast identification of difficult-to-detect covert communication channels.

Keywords: Cyber security, Covert storage channels, Covert timing channels, Machine Learning, Convonutional Neural Networks

Santrauka

Nedeklaruoti komunikaciniai kanalai – tai slaptas informacijos perdavimo būdas, naudojant kompiuterinių tinklų komunikacijas. Įkvėpti dar Antikos laikų karo strategų taikomų metodų slaptai perduoti žinutes, šie komunikacijos kanalai apsimeta, kad vykdo įprastą komunikaciją tarp įrenginių, tačiau į siunčiamų paketų nereikalingas ar tuščias vietas įterpia slaptą informaciją. Šis metodas taip pat gali manipuliuoti tokių paketų išsiuntimo laikais. Toks slaptas komunikacijos būdas dažniausiai naudojamas duomenų eksfiltracijai, tačiau praktikoje jis gali būti taikomas ir kitoms funkcijoms, pavyzdžiui, slaptam komandų perdavimui užkrėstam įrenginiui arba kaip legali priemonė komunikuoti tarp įrenginių, siekiant išvengti slapto tinklo srauto klausytojo informacijos perskaitymo.

Šis darbas pagrindinį dėmesį skiria tokių komunikacijos kanalų aptikimui. Tyrimo metu buvo sukurtas ir kituose straipsniuose dar neaprašytas nedeklaruotas komunikacijos kanalas, kuris išnaudoja maršrutizatoriaus IS-IS protokolo įprastas funkcijas, perduodant slaptą informaciją iš vienos saugumo zonos į mažiau saugomą zoną. Implementavus tokį nedeklaruotą komunikacijos kanalą ir išbandžius jį praktiškai, šiame darbe buvo pateiktas pasiūlymas, kaip tokį komunikacijos kanalą aptikti ir nutraukti jo veiklą.

Darbo eigoje buvo tęsiamas identifikavimo metodų tyrimas, koncentruojantis į mašininio mokymosi bei neuroninių tinklų modelius, juos apmokant, vertinant rezultatus bei tikslumą nustatant nedeklaruotus komunikacijos kanalus. Identifikuojant slaptą komunikaciją, buvo nustatytas geriausiai veikiantis mašininio mokymosi modelis, kuris parodė labai gerus rezultatus. Visgi, pabrėžiama, jog norint, kad tokie metodai veiktų realiose sistemose, o ne laboratorinėmis sąlygomis, būtina užtikrinti didelę duomenų variaciją ir sistemingai dirbti su tokių modelių apmokymu.

Identifikuojant slaptą komunikaciją, buvo sukurtas konvoliucinis neuroninis tinklas, kuris rodė itin gerus rezultatus. Eksperimentai parodė šio tipo modelių pranašumą prieš tradicinius mašininio mokymosi modelius ir prisidėjo prie ateities tyrimų tiksliai bei greitai identifikuojant sudėtingai aptinkamus slaptos komunikacijos kanalus.

Raktiniai žodžiai: Kibernetinis saugumas, nedeklaruoti komunikacijos kanalai, neuroniniai tinklai, mašininis mokymasis

Introduction

This research focuses on the identification and detection of covert communication channels within network protocols, with a particular focus on improving security measures against these concealed data transfer methods. Covert channels exploit the standard mechanisms of network protocols to secretly transfer information, making them difficult to detect and posing a substantial threat to network security. This work introduces a novel storage covert channel in IS-IS routing protocol and discusses a detection method for it. The novel aspect of this covert channel lies in its operating in the Data link layer.

The goal of this work is to perform a comprehensive analysis of existing methods for data hiding, transmission, and detection.

To achieve the objective, the research is structured around several key tasks.

- Analysis of current state of the art in the fields of information obfuscation/hiding and covert channel identification.
- A thorough examination of the prevailing state of the art in the domains of information obfuscation/hidding and covert channel identification will be undertaken.
- To conduct experiments in a controlled network environment to test covert data transmission techniques, thereby informing the development of preliminary algorithms for data hiding and detection
- To test and evaluate the analyzed algorithms to determine their effectiveness, using the results to guide further research on advancing the identification of covert communication channels

This structured approach will contribute to enhancing the overall security framework against covert channels in network communication.

Covert channel

In the contemporary era, the majority of communication channels are now subject to either local policies or international restrictions and rules. A designated route within a computer system or network that adheres to these policies is designated as an ”overt” communication channel [10]. Over the years a variety of methods have been devised and tested within the objective of circumventing and violating the security policies of trusted network. The term steganography originates from antiquity, a period in which military secrecy was paramount for achieving victory in warfare [26]. Encrypting such information was a sound security measure, but it was inevitable that the code would eventually be decoded. However, what if the message was never even found? The earliest known examples of cryptography can be traced back to ancient Greece, and within this corpus, we find examples of steganography. In this era, concealing information necessitated rudimentary tools such as a wooden tablet with the message inscribed upon it with a layer of wax employed to obfuscate the message [26]. Only the receiver knows that the information is visible once you melt the wax off. But as the way of people communicating changed thus more complex methods arose to hide information.

Digital media steganography uses media like images, video, and audio files for information hiding. Embedding these observable objects with hidden information is related to tricking the human senses into believing that the media was not altered in any way. Hiding a whole image in a single pixel which is a part of a bigger picture cannot be seen that easily. Text hiding within another text is where methods like white space, syntactic methods, and semantic methods for encoding are used to make the embedded text to be unreadable[1]. This way of hiding malware has proven to be a viable technique for infected programs. Lines that appear visibly empty contain a mix of invisible (non-printable) characters like tabs, spaces, and line feeds that can when converted to binary, form parts of executable code[20].

Various computer system components are also utilized for transferring hidden information, for example utilizing CPU or hard disk arm position [7]. This kind of covert communication requires the receiver to be able to read these system resources. This exemplifies that various types of system security breaches can be a tool for covert channel communication.

A forerunner of digital media steganography is network steganography. It is a younger method of hiding information thus more research is set towards network steganography. Unlike digital media, network steganography utilizes different layer functions of the OSI model or TCP/IP stack thus making it more versatile. Network steganography is limited by transfer speed but the transmission can be executed over a long period of time, hence this kind of communication is much more difficult to detect or examine during a forensic investigation. To compensate for the limited space available for data transfer in digital media steganography, network steganography offers virtually limitless space for such data; however, it does so at slower speeds.

S. Wendzel in his book about information hiding in communication networks formulates that modification of some properties in communication protocols is a condition of every network steganography technique [26]. Another part of information hiding in network communication is utilizing the protocol-specific characteristics that deal with internet communication flaws like errors and delays and those that define the type of information being exchanged, such as queries or file transfer. Combining these techniques with the effort of hiding information in legitimate traffic sums up network steganography.

In research papers, a more encountered term for information hiding is called a ”Covert channel”. There is a strong link between the terms network steganography and covert channels because they differ mainly in context and approach. ”An information transfer path that allows information to be transferred in a manner that violates the security policy of a trusted network is called a covert channel.” [16] So according to Girling’s paper a covert channel specifically refers to paths within a computing environment that are used to transfer information in ways that violate security policies.

One of the leading researchers in covert channels S. Wendzel is a part of a big project that aims to standardize the taxonomy related to covert channel communications. Regarding network covert channels or network steganography taxonomy S. Wendzel released a paper in 2015 and continued updating it over the years. A new revised version of the taxonomy was released in 2021 and will be utilized in this paper because all related research papers use the same taxonomy.

Network steganography leverages various aspects of network communication to hide data transfer. The core network covert channels are called Storage and Timing channels, with the former focusing on altering the state or value of network elements like packet headers, and the latter involving the manipulation of the timing or sequence of packets. These information-hiding patterns serve as the foundation for embedding covert messages within network traffic, demonstrating the versatility and adaptability of network steganography techniques to different network behaviors and properties.

A covert storage channel is a critical hiding method in network covert communication. This method includes altering header fields, payload fields, and other packet attributes. For instance, modifying reserved or unused bits in packet headers or employing least significant bit (LSB) modulation in packet fields are common tactics for hiding information without affecting the network communication's outward appearance. The focus of this paper will be exactly on this type of covert network communication.

The covert timing channel addresses the strategic manipulation of packet timing and sequencing to embed covert messages. Techniques such as altering packet inter-arrival times, manipulating packet order, or varying the number of elements within a flow are examples of how covert messages can be embedded within the regular network traffic, exploiting the variability and flexibility of network communications for information hiding purposes.

Another interesting and worth mentioning covert channel is an unintentional covert channel which in T.Schimdbauer and S.Wendzel's paper is referred to as a ”side channel”. [31]. A side channel forms when the sender leaks sensitive information unintentionally. An example of a network covert channel might be manipulating packet timing to transmit information, whereas the side channel approach discussed in another paper [23] leverages ICMP error message rate limits to infer DNS query source ports, enabling DNS cache poisoning.

1 Related Work

Detecting covert network storage channels presents significant challenges due to the time required for data transfer and the volume of information extracted. Moreover, packet analysis demands more system resources than usual [27]. A widely used approach for detection involves machine learning solutions. However, this introduces another issue: the necessity for large amounts of data to train ML models, as highlighted by [29]. While some data can be simulated in a controlled environment, applying these methods to real-world systems is time-consuming, requiring data collection directly from the system. Furthermore, as noted by S. Wendzel even simple modifications to traffic can enable covert channels to evade detection[40] . Despite these obstacles, the pursuit of research never stops. Increasingly, new research provides various algorithms designed to identify different types of covert channels, underscoring the ongoing effort to find solutions.

Machine learning technologies, particularly some models, can significantly enhance the detection of covert channels. The work of Elsadig M. and, Gafar A. proposes an ensemble model to detect covert channels that exploit packet lengths [14].Their work demonstrates a capability to achieve high accuracy and low error rates, outperforming existing detection methods.

Some recent and highly cited ML-based detection methods are called *GAS* and *SnapCatch*. *GAS* (Generic and Sensitive Anomaly Detection), an anomaly-based detection approach that identifies various types of NCTCs (Network covert timing channels). The key takeaway from the paper is that the detection of NCTCs can be significantly improved by focusing on the short-term variation in timing behavior of network traffic, which offers a more sensitive and generic detection reference that is resistant to major channel disguising techniques and effective even with small traffic samples.

SnapCatch demonstrates good detection accuracy converting network traffic into colored images for feature extraction and leveraging machine learning offers a robust and effective approach for detecting and localizing Covert Timing Channels in network traffic [11]. The methodology involves generating a dataset of network traffic, converting packet inter-arrival times into colored images, extracting image-based features, and training machine learning classifiers for detection.

Although these tools demonstrate significant results in a test environment S. Wendzel and S. Zillien in their article ”Weaknesses of Popular and Recent Covert Channel Detection Methods and a Remedy” explain that these methods can be unreliable when faced with slight modifications to the covert channel’s behavior [40]. Existing detection methods for covert timing channels can be easily bypassed with simple modifications to the covert channel’s behavior, highlighting a need for more robust detection strategies. The proposed remedy is an enhanced ϵ-similarity heuristic for improving the detection of covert timing channels. The paper also suggests that future research could extend to covert storage channels.

As demonstrated in the 2023 paper by M. Schneider, there is a lacuna in the extant literature concerning the establishment of covert channels, with a particular focus on the messages exchanged between routers in routing protocols [32]. Researchers successfully developed and implemented a covert channel within the OSPF routing protocol. This shows the need for monitoring routing protocol anomalies as part of security practice. More research is needed to find other covert channels within IGP protocols like IS-IS or EIGRP, as well as EGP protocols like BGP.

Wireless connections are no exception when it comes to covert channel communications. An interesting work is done by S. Zillien and S. Wendze where they implement and explain several methods to signal hidden information by triggering devices to reconnect to an access point [39]. Their introduced methods use the indirect covert channel. That is a kind of covert channel that doesn't require direct communication with the sender but rather reads the changes in a system component or its state.

Covert channels fundamentally rely on the existence of shared resources within an environment to operate. In real-world systems, this interdependence on shared resources is not just common but essential for the functioning of complex infrastructures. These shared resources, which range from network bandwidth to processing power, form the backbone of distributed systems. Without the availability of such shared assets, the operation of these systems would be unfeasible. The very nature of these environments, where resources are pooled and allocated dynamically, creates opportunities for covert channels to emerge. By exploiting the subtle nuances of how resources are accessed and utilized, covert channels can facilitate unauthorized communication, underscoring the intricate balance between resource sharing for efficiency and the potential for security vulnerabilities.

2 Methodology

The work will carry out a mixed-method of qualitative and quantitative analysis as research methodology. For the research of literature and possible tools to be chose will be used qualitative whereas quantitative methods will be used for evaluation of observable detection quality.

2.1 Router protocols

Internet packets are transferred across multiple hops, moving from one router to another until they arrive at their intended destinations. The internet is divided into autonomous systems (AS), which are defined as a collection of routers operating under a unified routing policy, managed by a single technical authority, and typically using but not limited to a single Interior Gateway Protocol (IGP). As a result, routers within an AS share routing information through IGP, while routers across different AS communicate via Exterior Gateway Protocol (EGP). Router protocols transmit data in little chunks known as packets. In ISO 10589, packets are referred to as protocol data units (PDU) [21]. In the realm of covert storage communications, these packets are of particular interest. These PDUs, which encapsulate the data being transmitted between network devices, could be leveraged for covert communication by embedding hidden messages within them. Unused or optional fields within the PDUs could transmit secret data without altering their normal appearance or function. However, different routing protocols have different fields and use these fields in various ways, thus requiring deeper research to understand each protocol. This understanding is crucial to determine how their communications could be leveraged by a covert storage communication channel.

RIP

RIP (Routing Information Protocol) is an interior gateway protocol used for exchanging routing information within small to medium-sized networks. Mainly used versions of RIP today are RIPv2 and RIPng for IPv6 networks. This version supports enhanced functionalities over its predecessor, including the ability to specify version 2 explicitly for sending and receiving updates, authentication mechanisms (e.g., MD5), and the management of route summarization and split horizon. [9]

In a network employing RIP protocol, communication starts when routers broadcast Request packets to discover routing information from their neighbors. Upon startup, or when operational, routers send out RIP Response messages, which contain their routing tables, at regular intervals, typically every 30 seconds. These messages are sent to each router's immediate neighbors, listing the routes they know and the distance in hops to each destination. Every field in RIPv2 table update packet was analyzed and a set of possible to alter fields for covert information transfer is shown in 1 table. The type of hiding pattern chosen is according to S.Wendzel taxonomy of non-payload steganography storage patterns. [37]

In scenarios where direct routing is used and the next hop is not specified (next hop is 0.0.0.0) or if the network topology is static, these fields might be manipulated for covert purposes. However,

1 table. *RIPv2 Potential Fields for Covert Information Hiding*

Field	**Size (Bits)**	**Hiding pattern**
Unused	16	Reserved/ Unused
Route Tag	16	Reserved/ Unused
Subnet Mask	32	Add redundancy
Metric	32	Reserved/ Unused

this requires careful planning to avoid routing disruptions and thus might not be suitable for a stable covert channel.

Route Tag header field, although used to distinguish between internal and external routes, creative use of this field or portions of it could be employed for covert communication, especially if the network does not rely on this distinction.

Unused field is reserved and typically set to zero, it can be a perfect place to hide information without affecting the routing functionality. In a scenario of default RIP settings transmits 16 bits with every RIP packet, sent every 30 seconds, ensures consistent performance. By embedding covert data into each RIP message, the system achieves a transmission rate of 0.53 bit/s, or 1.92 kbit/h. Therefore, a 24-character password could be covertly sent in just 6 minutes.

By leveraging optional settings, such as optional fields or subtly manipulating timers for faster information transfer, it's possible to encode and transmit hidden information across a network in a manner not originally intended by the protocol's design.

OSPF

OSPF (Open Shortest Path First) is a widely adopted interior gateway protocol designed for the efficient exchange of routing information within large and diverse networks. The primary versions of OSPF in use today are OSPFv2 for IPv4 environments and OSPFv3, which supports IPv6 networks. This protocol offers advanced capabilities beyond its predecessors, including the utilization of a link-state routing algorithm which allows for faster route calculation and convergence times.

Within an OSPF network, nodes initially establish a communication pathway with their adjacent nodes. Once their interfaces become active, they dispatch Hello packets every 10 seconds to each other. Then a period of database description (DD) exchange process, where the nodes first exchange and assess the information contained within their respective databases. In the last phase, the nodes actively solicit link state advertisements (LSAs) from one another to finalize the synchronization process.

The work of S. Michael, S. Daniel, and K. Jörg identifies possible packet fields of OSPF hello packet for implementation of hiding patterns and is visualized in 2 table.

2 table. *Applicable protocol fields and patterns.*

Protocol field	**Size**	**Pattern**	**IP-Ver.**
Sequence number	32 bit	Sequence	4/6
Authentication	64 bit	Add Redundancy	4
Address Prefix	n*32 bit	Add Redundancy	6
(TOS) Metric	16 bit	Reserved/Unused	4
Reserved	8/16 bit	Reserved/Unused	4/6
Options	24 bit	Reserved/Unused	6

Other OSPF protocol messages are rejected as possible covert communication media because of non-periodically sent messages or the lack of possibility to alter packet fields like the ones used for authentication in the OSPF authentication mechanism[32]. Because of periodically sent messages ”HELLO” packets are very suitable for covert channel communication. Packet fields like *Options* and *Reserved* offer up to 16 bits of space for a single transmission this translates to a covert storage channel bandwidth of up to 1.6 bit/s.

EIGRP

EIGRP (Enhanced Interior Gateway Routing Protocol) is a Cisco proprietary advanced distance vector routing protocol. EIGRP is an enhanced distance vector protocol, which relies on the Diffused Update Algorithm (DUAL) to calculate the shortest path to a destination within a network.

Within an EIGRP network, nodes initially establish communication with Neighbor Discovery. When a router receives a ”Hello” message from another router with matching EIGRP parameters (such as AS number and K-values), it acknowledges the message and a bidirectional communication is established, forming an EIGRP neighbor relationship. Initially, a router sends a Request message to new neighbors to solicit routing information. In response, neighbors send Update messages containing their routing table information. Each Update message received must be acknowledged by an ACK message, ensuring that both neighbors have synchronized routing tables. This process ensures data integrity and consistency across the network. Routers send Hello messages periodically on their configured interfaces to discover potential EIGRP neighbors.

Update messages are transferred only when there is a change in routing information. An increased volume of Update messages might alert a network monitoring tool and catch a covert communication happening. Query, and Reply messages could be also considered partially triggered events and detectable by a network traffic warden. Query messages are sent to ask neighboring routers if they have a feasible route to the lost destination and Reply is an answer to that query.

A distinctive feature of EIGRP is its use of periodic and triggered updates to maintain routing information. In a stable network, EIGRP minimizes overhead by only transmitting ”hello” packets to maintain neighbor relationships. These packets are sent every 5 seconds on high-speed interfaces and every 60 seconds on lower-speed links. EIGRP packets that are sent regularly are what concern most for the availability of a covert channel because they are not suspicious and provide guaranteed steganographic bandwidth.

In EIGRP hello packets several protocol header fields are commonly not used or are set to 0. They serve no purpose in a hello message so changing them keeps the communication stable thus being perfect fields for hiding information in a covert storage network communication. These fields and their sizes along with hiding patterns are shown in 3 table..

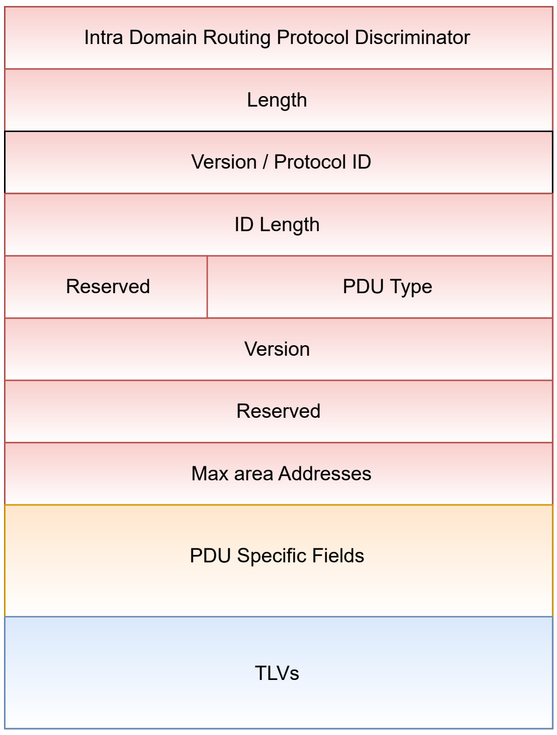
3 table. *EIGRP Potential Fields for Covert Information Hiding*

Field	Size (Bits)	Hiding pattern
Sequence Number	32	Reserved/ Unused
Acknowledgment Number	32	Reserved/ Unused

In a scenario of default settings, EIGRP transmits 32 bits of covert information with every ”Hello” packet, sent every 5 seconds. By embedding covert data into each EIGRP ”Hello” message, the system achieves a transmission rate of 6.4 bit/s or 23.04 kbit/h. Therefore, a 24-character password could be covertly sent in just 30 seconds. If you combine these two fields a total of 64 bits of information could be sent in one ”Hello” packet. That would mean a steganographic bandwidth of 12.8 bit/s. The same length password could be covertly sent in just 15 seconds.

IS-IS

IS-IS is a link-state interior gateway routing protocol that shares many similarities with OSPF. Based on the same SPF algorithm, the same basic flooding principle, and the same basic idea of splitting an autonomous system into areas [38]. IS-IS inserts its messages directly into Layer2 frames, and does not use IP in order to route the information. The PDU of IS-IS protocol is seen in 1 figure. Again for covert channel communication, it is important to look into protocols periodically sent messages to avoid detection. Link-state packets are responsible for sharing routing information among IS-IS nodes. IS-IS Hello packets, on the other hand, play a crucial role in setting up and sustaining neighbor relationships and are sent periodically just like EIGRP and OSPF. These packets are also known as IS-to-IS Hello PDUs (IIHs) seen in. On broadcast networks, Level-1 routers deploy Level-1 LAN IIHs, while Level-2 routers employ Level-2 LAN IIHs. For point-to-point networks, the P2P IIHs are used [24]. For regular routers, the default hello message interval is 10 seconds. However, for a designated intermediate system (DIS) on a multi-access link, hello packets are sent more frequently, every 3.3 seconds [25].



1 figure. *IS-IS PDU format*

IS-IS hello packets provide a few usable fields for covert storage communication outlined in table 4 table. The IS-IS protocol mandates that hello packets be padded to whichever is larger—the MTU or the LSP buffer size. This padding theoretically ensures that systems establishing adjacencies via a link can effectively receive and handle each other's LSPs and additional IS-IS packets. Given the fact that padding carries no functional information, a hello packet padding field could provide a huge amount of up to 11344 bits of hidden information size. Nevertheless, it is common for connecting routers to utilize the same MTU size on their interfaces, rendering the padding of hello packets generally unnecessary. Disabling this default padding behavior can lead to considerable bandwidth savings, particularly when short hello intervals are set and the MTU size is significantly large. Although hello packet padding is configured by default, some organizations may choose to disable it due to the bandwidth savings and other benefits offered by this configuration change. However to keep the connection stable leaving the default setting might be time-saving, also CISCO, a globally recognized leader in networking and cybersecurity solutions, recommends keeping the padding. Generally, you do not need to disable padding unless a link is experiencing slow performance. Thus before implementing this covert channel more information must be collected about IS-IS routing settings and it may vary from network to network. Other optional, authentication, or reserved fields are not that significant.

4 table. *IS-IS potential fields for Information Hiding*

Field	**Size (Bits)**	**Hiding pattern**
Reserved	8	Reserved/ Unused
Padding	11344	Reserved/ Unused

In a typical IS-IS setup, padding can provide a covert channel bandwidth of approximately 1418 bytes per Hello packet. Given the default interval of 10 seconds for sending Hello packets, this translates to a covert transmission rate of around 1.19 kilobits per second (kb/s). By embedding covert data into these Hello packets, a significant amount of information can be covertly sent within the network without detection.

IGP summary

As covert channels in routing protocols can achieve a covert bandwidth varying from 0.53 bit/s to 6.4 bit/s in contrast to covert channels that operate on the upper layers of the OSI model, IGPs have a narrower range of communication, confined to a single Autonomous System (AS) [32]. This limitation restricts IGP routing protocols to the internal exfiltration to the transmission of hidden information solely within a single AS. Therefore the application of such covert channel is limited to networks within a singular organization. Access is required to the nodes within the IGP domain, or additional measures must be taken to secure such access. Protocols with exterior communication possibilities are of interest in the concept of exfiltrating covert information between different AS.

BGP

BGP (Border Gateway Protocol) is the de facto inter-domain routing protocol of the Internet, enabling data and information to be routed between autonomous systems (ASes). By exchanging routing and reachability information among ASes, BGP ensures that data packets find an efficient and reliable path to their destination, making it instrumental in the backbone of the global Internet infrastructure.

Within a BGP (Border Gateway Protocol) network, routers (often referred to as BGP peers) initiate communication by establishing a TCP connection, typically using port 179. Following this, BGP Open messages are exchanged between the routers to negotiate session parameters, including BGP version, AS number, and hold time. If the parameters are acceptable to both sides, the connection is established, and the routers transition to an Established state, marking the formation of a BGP peering relationship. Initially, routers exchange full routing tables through BGP Update messages to share available routes. Subsequently, only incremental updates are sent to notify peers of routing changes, additions, or withdrawals. Keepalive messages are periodically transmitted between peers to maintain the connection and ensure its liveliness. In case of any errors or session issues, Notification messages are sent, which may lead to the termination of the BGP session. This intricate process of message exchange and negotiation ensures the stability, scalability, and resilience of internet routing by managing inter-domain routing decisions.

In the realm of Border Gateway Protocol (BGP), Update messages serve as the cornerstone for disseminating routing information across the internet, ensuring that data packets find the most efficient paths to their destinations. An unusual surge in BGP Update messages can signal to network monitoring systems the presence of anomalous or covert communications, as these messages typically indicate changes in network topology, route availability, or route preference. Similarly, BGP employs Notification messages, which are used to signal errors or terminate a BGP session between routers, and could be indicative of operational issues or malicious activity when observed in high volumes.

BGP also makes use of Keepalive messages, sent at regular intervals to maintain the connection between BGP peers. With the default configuration, these messages are sent every 60 seconds. Structure of a BPK Keepalive messages is shown in 5 table. While less conspicuous than Update messages, a disruption in the pattern of Keepalive messages could alert network administrators to potential link failures, configuration errors, or attempts to disrupt the BGP session. In essence, BGP's reliance on these types of messages for maintaining and optimizing internet routing infrastructure makes them focal points for network surveillance and anomaly detection efforts.

In the context of covert information transfer, Keepalive messages are the most of interest because they are sent regularly and don't require any triggers therefore are less likely to be detected.

5 table. *BGP KEEPALIVE message fields*

Field	**Size (Bits)**	**Hiding pattern**
Marker	128	Add Redundancy
Length	16	Add Redundancy
Type	8	Add Redundancy

Router protocol summary

Research shows positive results in achieving covert communication within a single OSPF area. Despite showing positive results in achieving covert communication within a single OSPF area, the research indicates a significant limitation: the inability to effectively establish covert channels between routers in different OSPF areas [32]. OSPF routers require matching Area IDs in their Hello packets to form adjacencies, which restricts their direct inter-area communication capabilities without special configurations like virtual links. Conversely, IS-IS inherently supports inter-area communications through its use of Level 2 Hello packets, which are designed to maintain connections and exchange information across different areas. This protocol structure allows IS-IS to facilitate more direct and impactful covert communication across areas. Therefore, while the OSPF-based research might successfully demonstrate intra-area covert channels, it would be inherently less capable compared to IS-IS in scenarios involving inter-area data transmission, where IS-IS's Level 2 communications provide a broader, more integrated platform for covert data exchange.

3 Covert channel development

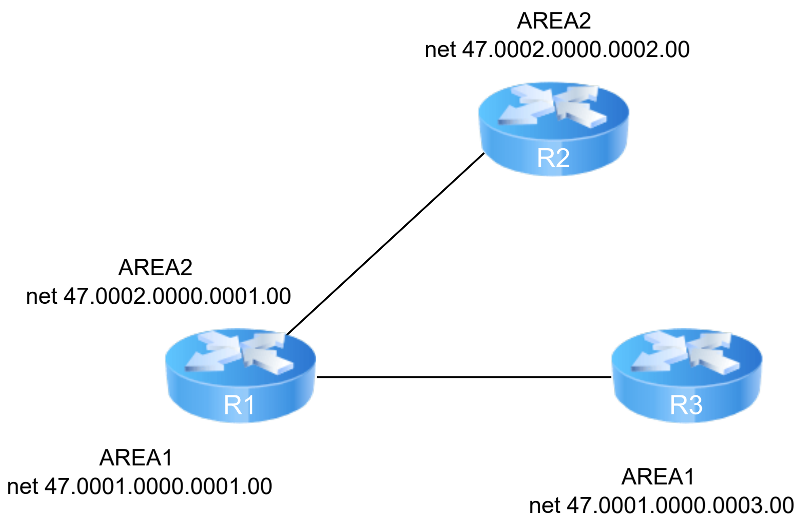
3.1 Data description

This research utilizes data from three virtual routers configured to run the IS-IS (Intermediate System to Intermediate System) protocol. Traffic generation is carried out using Hello packets, which are essential for establishing and maintaining neighbor relationships in IS-IS. These Hello packets are padded to a default size to ensure consistent communication. In this covert channel implementation, covert information will be embedded within the padded sections of the Hello packets to test the feasibility and effectiveness of covert communication through these channels. The communication's stability and performance will be evaluated to determine the success of the covert information transfer.

3.2 Setup

The lab setup involves which is seen in 2 figure. using VMware Workstation PRO to create three virtual machines (VMs) that will function as virtual routers. These routers will use the FRRouting (FRR) protocol suite to implement the IS-IS protocol. The FRR project was developed by multiple contributors using the C programming language. FRRouting was selected because it allows the use of the ISIS protocol without incurring additional costs. In contrast, virtual routers like MikroTik and open-source network simulators such as GNS3 require licenses for using the ISIS protocol. Cisco hardware was also considered; however, it similarly demands a license for the ISIS routing protocol. The virtual routers will be configured to operate in different IS-IS Areas to simulate a more complex network environment. In IS-IS routing protocol, a NET address (Network Entity Title) uniquely identifies a router and consists of an Area ID, a System ID, and a Network Selector (NSEL). These NET addresses are used when establishing point-to-point routing adjacencies within the IS-IS network.

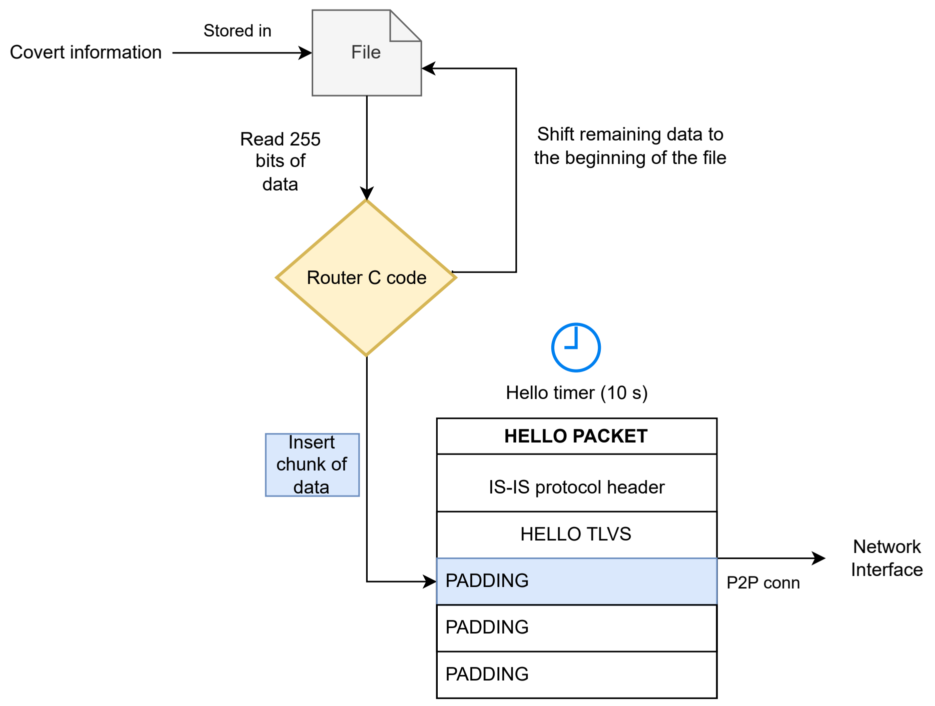
This setup allows for a controlled environment to test the embedding and extraction of covert information within IS-IS Hello packets, ensuring the integrity and stability of the communication between the routers and the effectiveness of the covert channel.



2 figure. *Lab setup diagram*

3.3 Covert channel design

The design of the covert channel is based on the manipulation of specific protocol fields within the Intermediate System to Intermediate System (ISIS) protocol, specifically within the Hello packets. Unlike the OSPF protocol, which required character-by-character transmission due to limited field sizes, the ISIS protocol allows for the transmission of a string of up to 1439 bits at once, thanks to its more generous padding and field sizes. Hidden information is first stored in a file on the file system of the routing node. Access to this file system is required to utilize the covert channel. When the covert channel is initialized, it attempts to read a chunk of data from this file, with each chunk being up to 255 bytes in size—the maximum capacity of a single padding section in an ISIS HELLO packet. The system opens the covert data file in binary mode and determines the total file size. If the file size exceeds the maximum chunk size (255 bytes), only a portion of the file (up to 255 bytes) is read into memory. The remaining data is shifted to the beginning of the file, effectively managing the file content like a queue. This ensures that subsequent reads will continue from where the last chunk ended. If the file is smaller than or equal to the chunk size, the entire file is read into memory, and the file is deleted after reading.



3 figure. *IS-IS hello packet Covert storage channel architecture*

Once the covert data chunk is loaded, it is embedded into the padding fields of the ISIS Hello packets during transmission. For each byte of available padding, the system writes a corresponding byte of the covert data to the padding field. If there is no more covert data to transmit, it fills the remaining padding with a default value which is NULL. This mechanism that checks if there is any covert information to transfer left not transferred and embeds it to a HELLO packet happens every 10 seconds (Default HELLO packet sending interval for ISIS protocol). 3 figure. figure is a visualization of how the covert channel works.

3.4 Proof of concept

The FRRouting (FRR) project code, written in C language, was edited to implement covert communication within IS-IS Hello packets. The IS-IS HELLO packet handling code was modified to include additional data in the padded section of the HELLO packets.

This involved editing the relevant C files to append covert information from a file named ”/tmp/covert_data” on the operating system. The HELLO interval triggers three essential functions in the implementation of the covert channel:

1. read_covert_data_chunk: This function reads a portion of the covert data from the ”/tmp/covert_data” file. It handles file operations, including reading, truncating, and deleting the file when all data has been read.

2. add_padding: This function is called when creating ISIS HELLO messages. It reads covert data using read_covert_data_chunk and incorporates this data into the padding of the message. If no covert data is available, it uses standard padding (0x00 bytes).

3. The ISIS protocol's existing functions, such as stream_putc, are used to construct the message with the modified padding.

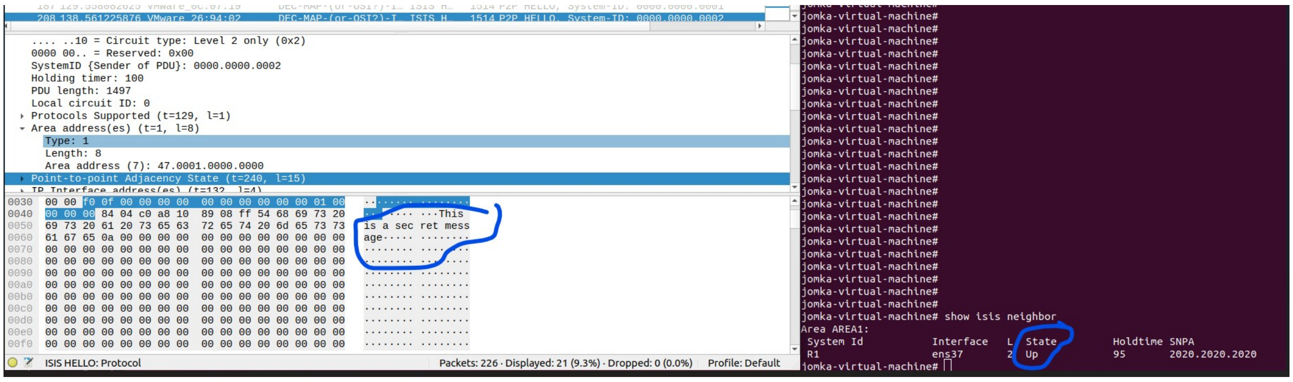
Another extra function for verification:

log_padding: This function logs the padding data to ”/tmp/padding_log” for debugging and verification purposes.

The modified Hello packets containing covert information were sent from Router 1 to Router 2. These packets maintained their structural integrity, ensuring that the routers continued to establish and maintain their IS-IS adjacency.

3.5 Evaluation

To demonstrate the functionality of the covert channel, the covert channel architecture shown in 3 figure. was used. Figure 4 figure. illustrates the practical application of this architecture, where a secret message was embedded within the reserved field of the ISIS protocol PDU. Wireshark was used to capture and analyze the network traffic. This allowed to confirm that the covert data was successfully embedded in the ISIS HELLO messages and transmitted over the network. Additionally, to ensure that the ISIS neighbors were functioning correctly running the ”show ISIS neighbor” command on the FRR router confirmed that the ISIS adjacencies were established and maintained, demonstrating that covert channel implementation did not disrupt the normal operation of the ISIS protocol. As per the protocol design, the router ignored the extra data in the padded section of the Hello packets since this padding is typically used to maintain packet size and does not affect routing functions. This allowed the covert data to be transmitted seamlessly without detection by the routers themselves.



```
208 138.561225876 VMware_26:94:02     DEC-MAP-(or-OSI?)-I… ISIS H…   1514 P2P HELLO, System-ID: 0000.0000.0002

    .... ..10 = Circuit type: Level 2 only (0x2)
    0000 00.. = Reserved: 0x00
    SystemID {Sender of PDU}: 0000.0000.0002
    Holding timer: 100
    PDU length: 1497
    Local circuit ID: 0
  › Protocols Supported (t=129, l=1)
  ⌄ Area address(es) (t=1, l=8)
      Type: 1
      Length: 8
      Area address (7): 47.0001.0000.0000
  › Point-to-point Adjacency State (t=240, l=15)

0030  00 00 f0 0f 00 00 00 00  00 00 00 00 00 00 01 00   ........ ........
0040  00 00 00 84 04 c0 a8 10  89 08 ff 54 68 69 73 20   ........ ...This 
0050  69 73 20 61 20 73 65 63  72 65 74 20 6d 65 73 73   is a sec ret mess
0060  61 67 65 0a 00 00 00 00  00 00 00 00 00 00 00 00   age..... ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

ISIS HELLO: Protocol          Packets: 226 · Displayed: 21 (9.3%) · Dropped: 0 (0.0%)    Profile: Default
```

```
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine#
jomka-virtual-machine# show isis neighbor
Area AREA1:
 System Id           Interface   L  State         Holdtime SNPA
 R1                  ens37       2  Up            95       2020.2020.2020
jomka-virtual-machine# 
```

4 figure. *Covert channel works and ISIS remains functionality*

A comprehensive evaluation of a covert channel is outlined in other papers

[26]. To conduct such evaluation it is essential to assess its undetectability, bandwidth, and robustness.

Detectability

The neighbor discovery and for that matter all operation of ISIS protocol remains unaffected by the transmission of the covert channel. As a result, it is unlikely that routers will detect any anomalies.

Defense designed for other network storage covert channels that employ similar hiding techniques could potentially make this covert channel ineffective. For instance, a traffic normalizer could Zero the padding bytes[17]. However, this will only occur if the normalizer is even used in infrastructure and it's known that ISIS protocol might be the carrier of a network covert channel.

Bandwith

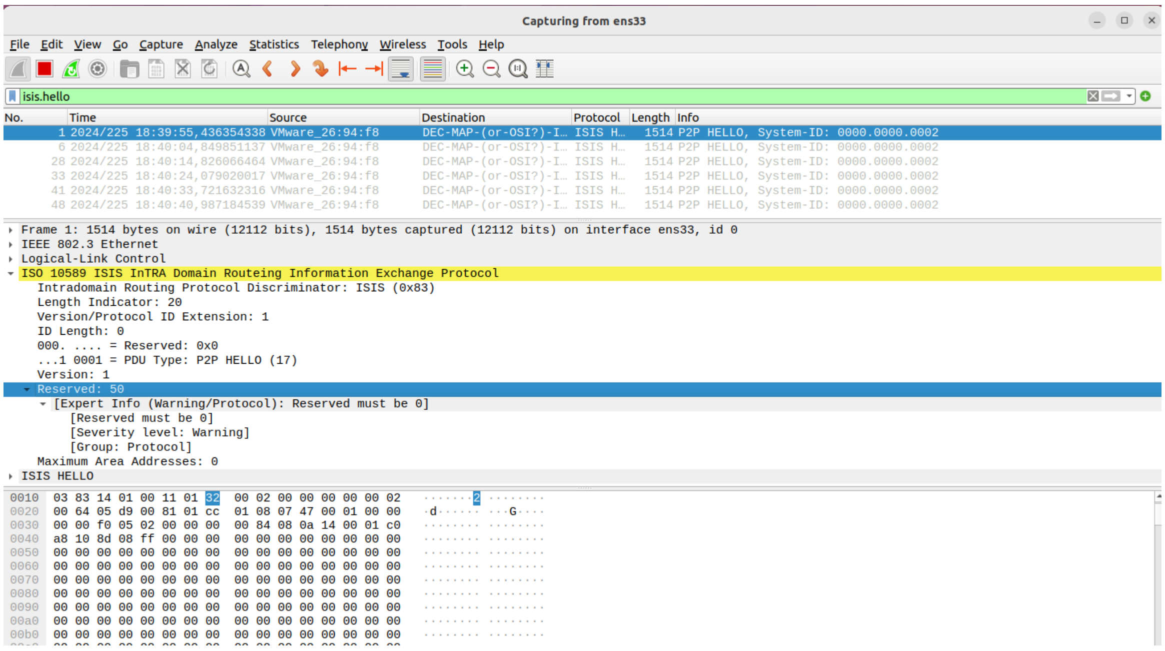
This covert channel architecture allows to transfer covert information with every ISIS hello packet which means that every 10 seconds a chunk of information is transferred. When utilizing only one of the padding fields a total of 255 bytes of covert data could be sent in one HELLO message. That gives around 204 bits/s or 734.4 kbits/h.

Robustness

Robustness was not tested because of the lab that was set up. It's a fairly small infrastructure and shares the same, however, we can imagine because IS-IS works in a single Autonomous system it can be safely assumed that this covert channel is fairly robust.

3.6 Extended Evaluation on additional PDU fields

The Proof of Concept (POC) was extended to evaluate the feasibility of embedding covert data into the reserved field located in the header of the ISIS protocol PDU. This experiment involved injecting non-zero values into the reserved bit and monitoring the resulting network traffic. Wireshark was used to inspect the traffic and identify any anomalies. Upon analysis, although routers were able to become ISIS neighbors Wireshark was able to detect that the reserved bit was not set to its usual value of 0, indicating the presence of altered data. This observation can be seen in Figure 5 figure.. Given that the communication occurs within the network of an autonomous system and within an organization's infrastructure it is unlikely that a tool for countermeasures would specifically inspect ISIS HELLO packets. Therefore, while covert communication could be detected, it is not guaranteed that it will be.

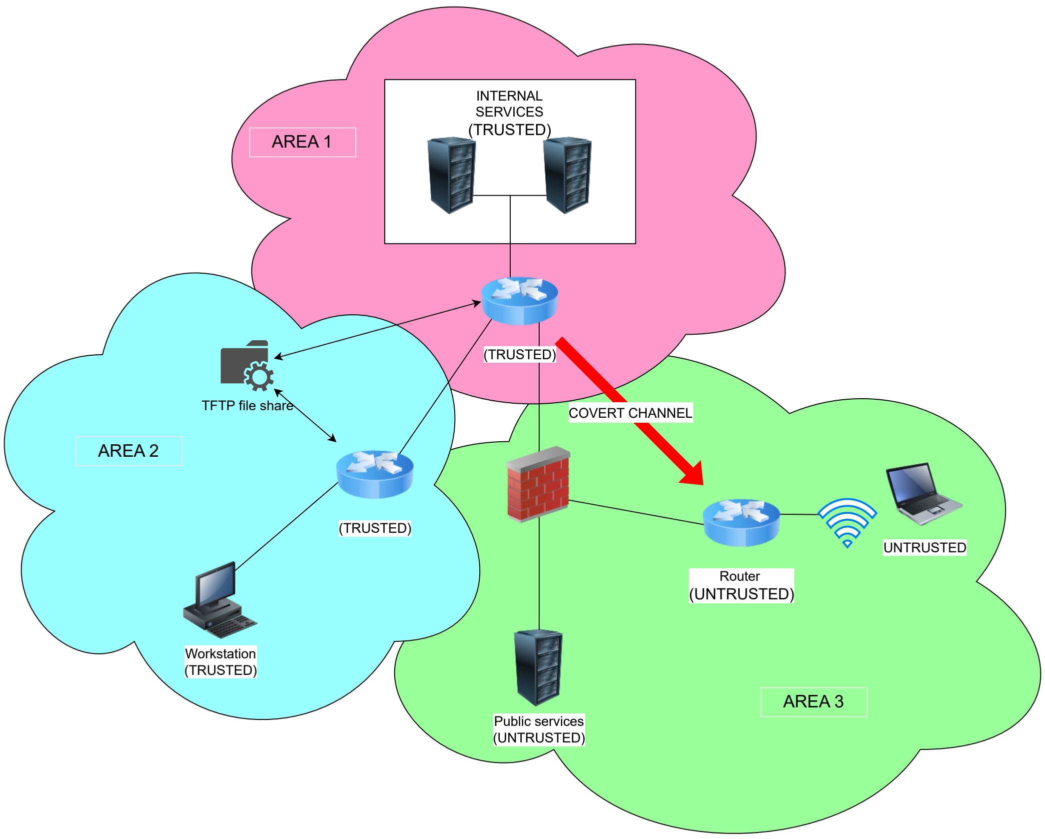

Capturing from ens33
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
isis.hello

```
No.   Time                            Source              Destination             Protocol  Length Info
    1 2024/225 18:39:55,436354338 VMware_26:94:f8   DEC-MAP-(or-OSI?)-I…  ISIS H…   1514 P2P HELLO, System-ID: 0000.0000.0002
    6 2024/225 18:40:04,849851137 VMware_26:94:f8   DEC-MAP-(or-OSI?)-I…  ISIS H…   1514 P2P HELLO, System-ID: 0000.0000.0002
   28 2024/225 18:40:14,826066464 VMware_26:94:f8   DEC-MAP-(or-OSI?)-I…  ISIS H…   1514 P2P HELLO, System-ID: 0000.0000.0002
   33 2024/225 18:40:24,079020017 VMware_26:94:f8   DEC-MAP-(or-OSI?)-I…  ISIS H…   1514 P2P HELLO, System-ID: 0000.0000.0002
   41 2024/225 18:40:33,721632316 VMware_26:94:f8   DEC-MAP-(or-OSI?)-I…  ISIS H…   1514 P2P HELLO, System-ID: 0000.0000.0002
   48 2024/225 18:40:40,987184539 VMware_26:94:f8   DEC-MAP-(or-OSI?)-I…  ISIS H…   1514 P2P HELLO, System-ID: 0000.0000.0002

Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface ens33, id 0
IEEE 802.3 Ethernet
Logical-Link Control
ISO 10589 ISIS InTRA Domain Routeing Information Exchange Protocol
    Intradomain Routing Protocol Discriminator: ISIS (0x83)
    Length Indicator: 20
    Version/Protocol ID Extension: 1
    ID Length: 0
    000. .... = Reserved: 0x0
    ...1 0001 = PDU Type: P2P HELLO (17)
    Version: 1
    Reserved: 50
      [Expert Info (Warning/Protocol): Reserved must be 0]
          [Reserved must be 0]
          [Severity level: Warning]
          [Group: Protocol]
    Maximum Area Addresses: 0
ISIS HELLO

0010  03 83 14 01 00 11 01 32  00 02 00 00 00 00 00 02   ·······2 ········
0020  00 64 05 d9 00 81 01 cc  01 08 07 47 00 01 00 00   ·d······ ···G····
0030  00 00 f0 05 02 00 00 00  00 84 08 0a 14 00 01 c0   ········ ········
0040  a8 10 8d 08 ff 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
00b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
```

5 figure. *Reserved field alteration in ISIS header*

3.7 Use case

The practical application of this experiment revolves around the potential for an insider attack via a supply chain compromise, where the integrity of centralized firmware or IOS updates distribution to routers is jeopardized and Layer 2 packet padding is not zeroed out. This scenario becomes particularly relevant in environments where routers are centrally managed and updated through shared file systems, such as FTP or TFTP servers. In such cases, an adversary could maliciously alter or replace updated files with compromised versions, potentially affecting the entire network infrastructure. The attack could then exploit the router permissions to access the file system, leveraging non-zeroed padding bits in Layer 2 communications to exfiltrate data from a secure environment to a less protected one specifically as shown in figure 6 figure., using HELLO packets to covertly transfer sensitive information, exploiting the routers legitimate access rights to the shared file system.



6 figure. *Covert channel in a multiple area infrastructure*

A significant limitation in prior work about OSPF [32] is the assumption that covert information transfer is confined to scenarios where all routers operate within a single OSPF area. This assumption does not hold in real-world networks, which often employ multiple areas to enhance security, particularly in environments with both trusted and untrusted zones. According to Cisco documentation [8] it is standard and recommended practice to deploy area border routers (ABRs) to segregate different security zones, ensuring that sensitive network areas remain out of reach. However, this research demonstrates that the IS-IS protocol offers a more robust architecture for covert channel communications. Unlike OSPF, which terminates HELLO messages at the first ABR due to the absence of multi-hop capabilities for such packets, IS-IS can transmit HELLO packets across different areas using Layer 2 of TCP/IP model communication, bypassing the limitations imposed by ABRs. This advantage positions IS-IS as a superior choice for covert communications in complex network environments. There are some IS-IS architectures that have backbone areas like OSPF. Future work could further explore the development of multi-hop covert channels across multiple routers, extending the capabilities of such attacks beyond the constraints observed in OSPF-based systems.

3.8 Detection and counter measures

Research on covert channels operating exclusively at Layer 2 has not been extensively documented, with most detection methods focusing on Network, Transport, or Application layers [2, 13]. For the IS-IS Hello packet covert channel discussed in this research, a simple countermeasure would be to disable packet padding, which would prevent the covert channel from embedding data. However, Cisco documentation advises maintaining padding, so detecting this type of covert communication is relatively straightforward using basic network traffic analysis tools.

While many covert channels exist, developing normalization rules for each can be resource-intensive. The Adaptive Warden, as proposed by Wendzel, offers a more efficient solution. Wendzel's research explains that adversaries often test various covert channels to find one that works within a specific network infrastructure before launching an attack [27]. To counter this, one could either enable all detection rules in a network normalizer, which would be resource-intensive and could disrupt normal communication, or take a more strategic approach by interrupting covert communication at key moments. This forces the adversary to restart data exfiltration, making the process more time-consuming and less efficient.

The Adaptive Warden achieves this by dynamically changing normalization rules, and disrupting the covert channel at various stages. Given the capacity of the IS-IS Hello packet covert channel to transfer significant amounts of data in a single packet, it would be wise for the adaptive algorithm to include the ISIS Hello message rule more frequently to effectively counteract this specific threat.

4 Modeling detection

The detection of network covert channels (NCCs) is a critical challenge in cybersecurity, as the techniques to covertly transmit information are always evolving and violating system security policies. As novel cybersecurity threats emerge some traditional detection or prevention systems like signature-based or anomaly-based approaches might struggle to identify sophisticated covert channels. Machine learning (ML) is an advanced and rapidly evolving technology, continuously improving in its ability to analyze vast amounts of data, identify patterns, and make predictions with increasing accuracy and efficiency. In the literature, machine learning (ML) techniques are frequently employed to enhance the detection and classification of NCCs. By leveraging algorithms like Support Vector Machines (SVM), k-nearest Neighbors (k-NN), and Deep Neural Networks (DNN), studies demonstrate the potential of ML to detect covert channels with high precision and low false positive rates[3, 13, 28]. Although findings in these studies underscore the potential of ML to enhance cybersecurity defenses they also highlight the importance of using diverse datasets. All of the studies use some kind of training and testing data sets and show great results. However, they failed to introduce the trained models to unseen covert channel data or new network domains. In the next sections of this work, an experiment will be carried out which will try to find the best ML technique for a given data set with a selected covert channel implemented as well as test this technique by introducing it with the same covert channel only on different communication flow.

4.1 Detection methodology

pcapStego tool

The pcapstego tool is a network steganography utility that allows users to embed secret data into network traffic captured in PCAP files [41]. It was written by M. Zuppelli and L. Caviglione mainly in Python. It manipulates specific fields within IP packets, such as TOS, TTL, ID, or even timing between packets, to covertly insert information without altering the overall structure of the network traffic. This makes it possible to hide data within ordinary network communication, which can then be extracted later, making it a useful tool for studying covert communication techniques in network security

When utilizing the pcapstego tool, several issues were encountered, primarily related to handling large PCAP files and the proper identification of TCP and UDP flows. Initially, the tool hung during the group by operation due to memory constraints when processing large datasets. This was resolved by breaking down the processing into smaller, manageable chunks and implementing a manual counting mechanism instead of relying on groupby. Additionally, the tool expected specific column names like tcp.srcport and tcp.dstport, but the actual data contained different column names (srcport and dstport). This mismatch was addressed by renaming the columns to match the expected format, ensuring compatibility with the existing code. As a result of these changes, the tool now processes large PCAP files efficiently, accurately identifies flows, and injects the payload without any issues, leading to a more robust and reliable steganography tool.

4.2 Data setup

The experiment was meticulously structured, beginning with data collection from the kaggle.com https://www.kaggle.com/datasets/ymirsky/network-attack-dataset-kitsune. Then the injection of covert data using a modified packet injector tool. A few considerations were taken as to how to prepare the traffic packets for model testing:

1. A few traffic flows will be chosen to inject covert information to mimic a real-life scenario where some endpoints are acting maliciously. A set of those flows would be prepared as normal communications (before malicious activity) and a portion would be injected with covert information and labeled accordingly. This is where other researchers fail to mention whether they injected random packets with covert information or not.

2. The data will consist of 80% overt packets and 20% of injected packets.

3. 10 % of the prepared data will be taken as validation for testing the model on unseen data.

The pcapStego tool supports an injection of IPv4 Identification Number (16bit/pkt) which is the chosen covert channel to test. For machine learning model testing Orange Data Miner tool was used. It is a powerful open-source data analysis and visualization tool that is widely used for machine learning and data mining tasks. It provides an intuitive, drag-and-drop interface that allows users to build and test various machine learning models without requiring extensive programming knowledge. The tool's visualization capabilities also aided in interpreting the results and fine-tuning the models for optimal performance.

4.3 Evaluation

To evaluate the performance of the network covert channel detection models, several key metrics were calculated, including accuracy, false positives, detection rate, and precision. These metrics were derived using a confusion matrix, which categorizes the outcomes of the classification process. Specifically, the confusion matrix includes False Negatives (FN), representing instances where a covert channel was incorrectly classified as normal; True Negatives (TN), where normal traffic was correctly identified; False Positives (FP), where normal traffic was incorrectly flagged as a covert channel; and True Positives (TP), where covert channels were accurately detected.

4.4 Experimental results

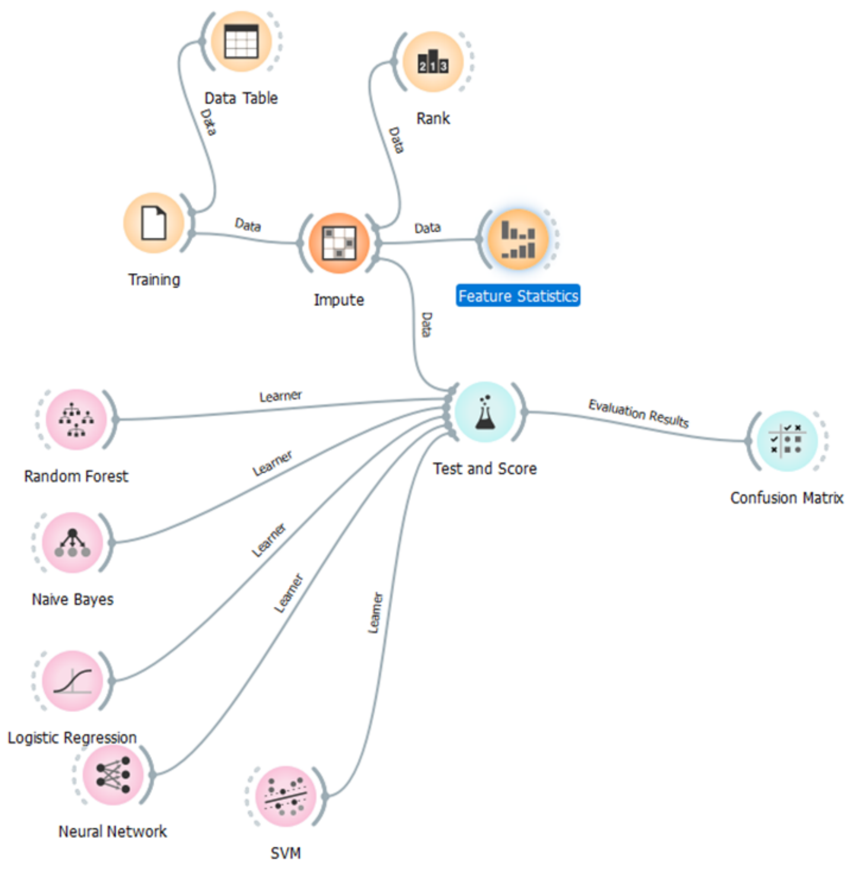
Firstly, using Wireshark, packets were prepared for injection, along with overt communications. This was accomplished by analyzing the packet count in Wireshark's communication function. The selected packets were filtered and exported for injection. After injecting covert information into the packets, they were merged into one dataset using Excel spreadsheets and labeled accordingly. An additional column was inserted in the spreadsheet to extract the identification number in decimal form. After these preparations, there were approximately 205k overt packets and 47k injected packets. The distribution of normal and covert network packets is summarized in Table 6 table.

6 table. *Sample distributions in the Dataset*

Type of Packets	Number of Packets	Sample Percentage
Injected packets	47271	19.05%
Normal packets	200777	80.95%

To generate the covert information, a script was written to produce random UTF-8 symbols. Then, using the pcapstego tool, the identification number field of TCP packets was injected with the covert information.

To find the best performing model a 3 fold Cross Validation setup was designed in Orange. The setup can be seen in figure 7 figure. Cross Validation divides the data into folds and tests the model. This gives a general view of which model might perform best when training and testing on full data. For this test results are seen in 7 figure.



7 figure. *Cross validation Orange data miner setup*

7 table. *Performance metrics of different models in cross validation*

Model	**AUC**	**CA**	**F1**	**Prec**	**Recall**	**MCC**
Random Forest	**0.998**	**0.998**	**0.998**	**0.998**	**0.998**	**0.994**
Naive Bayes	0.995	0.886	0.895	0.928	0.886	0.733
Logistic Regression	0.998	0.997	0.997	0.997	0.997	0.990
Neural Network	**1.000**	**0.998**	**0.998**	**0.998**	**0.998**	**0.995**
SVM	0.885	0.810	0.829	0.903	0.810	0.615

At first glance at Table 7 table. Random Forest and Neural Network show the best results however scores are nearly perfect which alerts that overfitting of the models might be happening. To avoid overfitting better feature selection must be done. To achieve the goal of the models performing best on unseen data Source and Destination IPs were skipped as features. As these changes did not help avoid overfiting more features were skipped. Orange's "Rank" widget shows feature correlation to the target feature and helped decide which ones to skip. Length, Source, and Destination ports as well as Protocol features were skipped which resulted in models not overfit anymore.

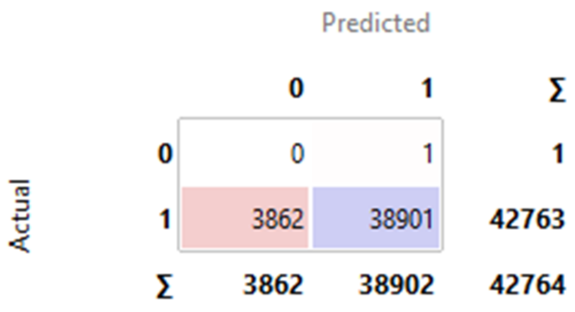
Model	**AUC**	**CA**	**F1**	**Prec**	**Recall**	**MCC**
Random Forest	**0.984**	**0.975**	**0.976**	**0.977**	**0.975**	**0.923**
Neural Network	**0.974**	**0.944**	**0.946**	**0.953**	**0.944**	**0.839**
Logistic Regression	0.935	0.895	0.897	0.900	0.895	0.673
SVM	0.326	0.384	0.386	0.855	0.384	0.238
Naive Bayes	0.947	0.899	0.903	0.913	0.899	0.709

8 table. *Performance metrics of different models*

Next, the data was divided into training and testing sets using Orange's Data Sampler widget, allowing for further evaluation of the models. Since Random Forest and Neural Network performed the best in cross-validation, these models were selected for testing on the training and test data, which produced similar results. To identify the best-performing model, several additional experiments were conducted:

1. Changing various model parameters to achieve the best results
2. Testing the trained model on unseen data,
3. Testing the trained model on data consisting only of injected packets
4. Testing the trained model on data consisting only of normal packets

For Neural Networks changing the hidden layer neuron number showed some improvement however not significant. The best configuration for Neural networks 128 in the first layer and 64 in the second was found to show best results though the time it took for the model to predict took longer. For Random Forest Number of trees of 100 also showed the best results although also not significant.


Predicted

Actual	0	1	Σ
0	0	1	1
1	3862	38901	42763
Σ	3862	38902	42764

H]

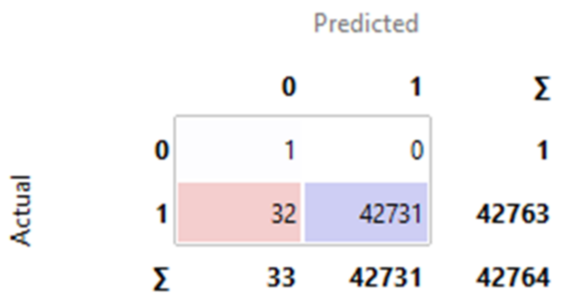
9 figure. *Neural Network confusion matrix on data consisting only one normal packet*

Testing the trained model on unseen data revealed that Random Forest still remains a better model with higher scores overall when detecting injected packets. These results are seen in table 9 table.

9 table. *Performance metrics on unseen data*

Model	AUC	CA	F1	Prec	Recall	MCC
Random Forest	0.983	0.976	0.920	0.866	0.982	0.909
Neural Network	0.977	0.947	0.838	0.721	1.000	0.823

Introducing the trained models with datasets that consist only of injected or normal packets showed that Random forest still holds as the best performing model as it made fewer errors than then Neural Network model. A confusion matrix where 0 stands for normal packet and 1 is injected is presented in Figure 8 figure. where Random Forest was able to identify the one normal packet. And observing the Neural Network confusion matrix which is presented in Figure 9 figure. it is visible that the Neural Network model made more mistakes than Random Forest.


Predicted

Actual	0	1	Σ
0	1	0	1
1	32	42731	42763
Σ	33	42731	42764

8 figure. *Random Forest confusion matrix on data consisting only one normal packet*

5 Results overview

To understand why a machine learning model produces such good results, it is essential to examine the structure of the IP header and how network equipment or operating systems assign values to these fields.

In the experiment, secret information was injected into the IP identification field by reading data from a text file and directly assigning it to this field. While modifying this field does not disrupt network functionality (if there are no packet transmission errors), it is easily detectable because different operating systems follow specific patterns when generating values for this field. A detailed analysis of these patterns can be found in Klein A.'s work [22]. For instance, Windows uses a unique counter for each source-destination tuple to initialize the IP identification field. These counters are stored in objects called ”Path,” which are organized in a hash table named ”PathSet.” When a new packet is sent between a specific source and destination address for the first time, a new ”Path” object is created, and its identification counter is initialized with a random value. For subsequent packets, the counter is incremented until the connection ends or a certain limit is reached.

Due to this logic, any analysis tool that monitors full traffic flows between devices could detect deviations from these patterns, flagging them as potentially illegal activity.

When testing machine learning models in this research data consisted of separate packets, so the patterns might not be a factor. However, observing the logic of the Random Forest machine learning model in classifying legitimate versus illegitimate packets, it became apparent that the model relied heavily on the magnitude of the identification field values. It can be inferred that the good results were due to poor choices in the method for injecting secret data into the IP identification field. In some cases, the identification field values were unnaturally large or small, making them easily detectable.

Changes in the TTL (Time-to-Live) field of the IP header also do not affect normal packet transmission. Zander [5] described a method for using this field to transmit secret information. The TTL field, essential for preventing routing loops, decreases by one at each hop and is typically initialized with a value between 0 and 255. Secret data can be encoded by using a high TTL value to represent a 1 and a low value to represent a 0, minimizing the impact of changes from hops. However, this method risks detection since most systems use fixed TTL values, and deviations from these are easily noticeable. The authors recommended using two adjacent TTL values and changing them infrequently to blend with normal traffic. Nonetheless, such a covert communication method is likely easy to detect due to the fixed nature of TTL values, and any deviation from normal ranges can be identified and neutralized by normalizing TTL values before forwarding packets.

In practice, exploiting individual IP packet fields for covert data transmission is nearly impossible due to the high likelihood of data loss and the obvious deviations from normal values [6]. However, combining the use of multiple fields with confirmation of successfully received data can increase the amount of transmitted information and help avoid significant deviations from normal traffic. This approach can circumvent traffic analysis and normalization tools. This concept was extensively detailed and experimentally validated in the paper ”A TCP-based covert channel with integrity check and retransmission” [6].

6 Neural networks in network security process

Preparing the data for the proposed ML solution is time-consuming, such as feature engineering, and selecting the right feature format instead of raw bytes. Analyzed methods have significantly contributed to improving detection accuracy yet require expert knowledge and involvement to process the extensive data and more experiments showed that these models are prone to overfit. These observations are backed up by many other types of research on Machine Learning intrusion or malware detection [36]. Thus, more research has been done on other methods that would require less feature engineering, simpler data formats, and faster computation times.

Neural networks have become an innovative tool in network security, offering advanced capabilities for detecting and mitigating a wide range of cyber threats. Leveraging their ability to analyze large datasets and learn intricate patterns, neural networks excel in tasks such as intrusion detection, anomaly detection, and malicious traffic classification.[4]

Deep convolutional neural networks (CNNs), a machine learning method rooted in processing image data, have gained significant and promising results in computer vision due to their remarkable performance [30]. By excelling in automatically extracting spatial features from images CNNs overcome a fundamental limitation of traditional text- and numeric-based machine learning methods: their inability to effectively represent and process complex relationships within data. This limitation becomes particularly apparent when network traffic payload data is used as the feature vector, as such data often involves intricate patterns and high dimensionality. [15] Furthermore, CNNs eliminate much of the need for manual feature engineering by autonomously identifying relevant features from raw image data [33], a task that is labor-intensive in text- and numeric-based approaches. As a result, converting network data into image form and leveraging CNNs offers a promising solution for network intrusion detection systems (NIDS), especially when dealing with the vast complexity of network payloads.

7 Network traffic anomaly detection using CNN

The paper ”Squeeze-and-Excitation Networks” introduces a novel architectural unit called the Squeeze-and-Excitation (SE) block, enhancing convolutional neural networks (CNNs) for tasks like network traffic anomaly detection.[19] While primarily designed for image recognition, SE blocks recalibrate channel-wise feature responses to improve representational power, which can extend to analyzing traffic patterns in cybersecurity. Building upon the principles of SENet, the SeNet-I model extends these capabilities specifically for detecting network anomalies and intrusions[15]. SeNet-I transforms raw network traffic—both packet-based and flow-based—into serialized RGB images, enabling deep concatenated CNNs to effectively capture high-level spatial and temporal patterns in the data. By doing so, SeNet-I overcomes the challenges of high-dimensional payload data and evolving cyberattack strategies, achieving enhanced anomaly detection performance without the need for manual feature engineering. SeNet-I demonstrated its superiority. Serialized RGB images achieved exceptional accuracy and F1 scores, reaching up to 96% for flow-based data and 83% for packet-based data.

In addition to the advanced neural network techniques, more research has further demonstrated the efficiency of CNN in analyzing network traffic to detect anomalies and covert channels. These models are based on the hierarchical structure of network data, converting raw packets and flows into feature-rich formats for processing. For instance, recent research has introduced a novel CNN architecture designed specifically for IPv6 networks [3]. By taking advantage of the protocol’s extended features like its 128-bit address space and new header fields imitated anomalies, DDoS attacks, and even covert channels. Later the study proposed a three-layer CNN architecture capable of classifying IPv6 traffic into normal or malicious categories with exceptional accuracy. By transforming network packets into high-dimensional feature vectors, the model extracts meaningful patterns that distinguish between benign and anomalous flows. Notably, the approach excelled in handling IPv6 traffic covert channel classification, such as flow labels and ICMPv6 header fields.

More research has been done exploring the application of Convolutional Neural Networks and Long Short-Term Memory networks in detecting network traffic anomalies. Hybrid models of CNN-LSTM demonstrated exceptional performance in identifying covert timing channels in Sharouq Al-Eid’s paper [12]. In the research, a hybrid model was used to detect covert timing channels (CTCs), which exploit packet timing intervals to transmit hidden data undetected by conventional methods. Using real inter-arrival time datasets with varying stream lengths, the hybrid model achieved a detection accuracy of 97.5%.

These advancements show how powerful CNNs can be for cybersecurity. They make it easier to classify and process complex network traffic accurately and efficiently. Models like this one are a great fit for dealing with the challenges of modern network anomaly detection. They offer a reliable and flexible way to build intrusion detection systems (IDS).

8 CNN for storage and timing covert channel detection

To advance this research on covert channel detection methods, the following experiments are proposed:

1. **Evaluation of Convolutional Neural Network (CNN) models against advanced covert storage channels:** This experiment focuses on sophisticated covert channels that manipulate multiple altered fields within a packet. Such channels represent a significant challenge compared to simple injections into a single IP header field.

2. **Evaluation of CNN models against covert timing channels:** This experiment aims to address the subtle and difficult-to-detect nature of timing-based covert channels. Although various studies have explored methods for identifying such channels, limited research has incorporated CNN-based approaches, highlighting the need for further investigation in this area.

9 Multiple altered fields in a packet

TCP protocol packets are also susceptible to covert channel attacks. Several works explored using TCP's Initial Sequence Number (ISN) field for covert channels. The ISN, crucial for tracking transmitted bytes and distinguishing connections, can theoretically hold any value, making it challenging for a traffic analyzer to alter without disrupting traffic. Neutralizing such channels would require proxying all outgoing TCP connections, which may degrade performance [18]. Multiple research proposed hiding data by encoding ASCII characters in the ISN's most significant bits, randomizing the rest for realism, achieving a bandwidth of 8 bits per TCP connection.

The smallest amount of bandwidth in covert channels that utilize packets IP fields should be considered as good bandwidth because packet fields that are in the lower layers of OSI can't hold big amounts of data. Covert data is sent in small chunks and this provides the most important aspect of a covert channel - stealthiness. So the proposed technique has great bandwidth for a covert channel while keeping it stealthy enough. This technique was later improved by other researchers to address its Robustness - a third important aspect of a covert channel.

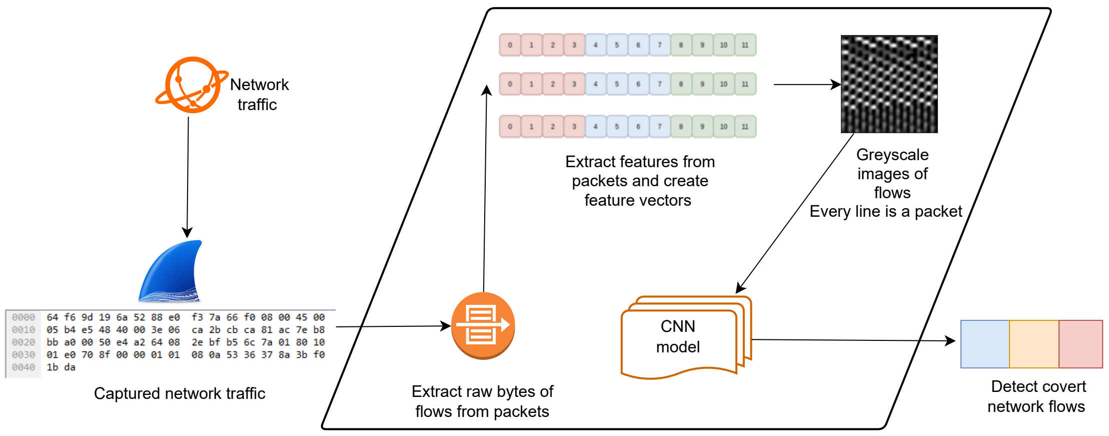
To improve the covert channel to the point where the covert stream to be cannot be interrupted because of some data being lost, researchers utilized the functionality of TCP's three-way handshake to check the integrity of data and retransmit it[6]. This method allows for covert data injected into the first TCP packet of a new connection to be read by the covert receiver. The receiver reads the ISN field of the packet and answers with a packet consisting of yet another forged TCP packet. This packet's RST and ACK flags are set to 1 to pretend the receiver is simply trying to reject the Sender's attempt to establish a three-way handshake. The sender checks if the received packet is as expected meaning that the integrity of the packet checks out and then continues sending another chunk of secret information. This method is further developed by simultaneously utilizing the TCP header's ISN field and the IP header's Identification field. Secret information is first encrypted and then embedded in the ISN field, while the Identification field is used to transmit the decryption key needed by the Covert Receiver to decode the hidden data. A large amount of consistent failed handshakes might raise suspicion of a network analyzer. With some modifications involving choosing time frames when to initialize the handshakes and waiting for different time intervals after sending each packet the authors managed to lower the score of an open-source software designed to analyze network traffic to an non alarming level.

This technique proved to be successful in maintaining the integrity and reliability of covert data transmission while significantly reducing the likelihood of detection by traditional network analyzers. By carefully timing the initiation of handshakes and introducing varied intervals between packets, the covert channel blended more effectively into normal traffic patterns, lowering detection scores to non-alarming levels in open-source network monitoring tools. However, this success raises an important question: could modern techniques, such as neural networks, be trained to analyze similar traffic flows and detect covert channels with altered fields?

10 Detection Model

10.1 Architecture

The proposed model is designed to process network traffic by transforming raw packets into a selected feature vector, which is subsequently converted into an image and used as input for a deep concatenated CNN model. The whole process is shown in figure 10 figure.The feature vector is carefully constructed from specific IPv4 fields known to be susceptible to covert information injection. Depending on the protocol, additional relevant features are appended to the vector. The novelty of the model is that each image generated is of a fixed size, representing a network flow, where each line corresponds to a packet and its selected attributes. To enhance computational efficiency and model performance, feature selection is a critical step. It not only ensures the dimensionality of the data remains manageable for faster processing but also focuses on fields that contribute directly to covert information injection. Fields like ports or source and destination IPs, which are irrelevant for covert channel formation, are excluded, thereby simplifying the training process and reducing unnecessary complexity in the model.


Network traffic
Extract features from packets and create feature vectors
Greyscale images of flows
Every line is a packet
CNN model
Captured network traffic
Extract raw bytes of flows from packets
Detect covert network flows

```
0000  64 f6 9d 19 6a 52 88 e0  f3 7a 66 f0 08 00 45 00
0010  05 b4 e5 48 40 00 3e 06  ca 2b cb ca 81 ac 7e b8
0020  bb a0 00 50 e4 a2 64 08  2e bf b5 6c 7a 01 80 10
0030  01 e0 70 8f 00 00 01 01  08 0a 53 36 37 8a 3b f0
0040  1b da
```

10 figure. *Proposed CNN model for Benign and covert traffic flow classification*

10.2 Data

The data set was created using the before mentioned pcapStego tool injecting the normal network traffic data from http://mawi.wide.ad.jp. After injecting the packets and creating images of flows the flow images are labeled and stored in a .csv file. The dataset from mawi.wide.ad.jp consist of various 24 hour network traffic.With this data set in the experiment phase of the research more then 100k images of flows were created.

10.3 Image generation and pattern

In this research on converting raw packet data into images, an approach outlined in a similar paper was considered[15]. In the paper authors utilized linear transformation and fixed dimensions (e.g., 64×32) for packet-based data. The authors of the paper chose linear transformation as it is a widely adopted and computationally efficient method for arranging one-dimensional data into two-dimensional arrays. The selected dimensions were powers of two, a choice made to balance computational efficiency, compatibility with deep learning architectures, and effective spatial representation of the data. These dimensions also ensured that sufficient information could be captured while maintaining uniformity across samples. However, dataset that was acquired for this research presented different challenges, as the network flows involved had a small packet count, and using the proposed 64×32 dimensions would have introduced excessive padding, potentially diminishing the clarity of data representation. Instead, 28×28 images were created, where each line represents a different packet. This approach minimizes unnecessary padding while leaving room for future exploration with additional protocols or varying packet structures. Furthermore, the chosen method allows for the identification of distinct packet patterns by matching similar outgoing calls across different packet sequences, enhancing the ability to detect recurring behaviors or anomalies in network traffic.

11 Covert storage channel experimental Data and Preparation

11.1 Data conversion to images approach

In this experiment, IPv4 flows were extracted from a large dataset of packet captures (PCAPs). The dataset was divided into distinct flows, each containing up to 28 packets, which aligns with the granularity needed for subsequent covert channel and image generation tasks. The following steps were taken to prepare the data for a Neural network learning model:

Step 1: Dividing Flows Using a flow key derived from source, destination, ports and protocol packets were grouped into flows. Each flow was capped at 28 packets. This preprocessing step ensured that all flows were uniform in size, a necessary condition for consistent image creation. Non-IP packets were filtered out during this process.

Step 2: Injecting Covert Data To simulate covert channels, a modified PCAP Stego Tool was used to inject secret data into 40% of the flows. The injection process employed a script to encode secret messages into the IP ID field and TCP Initial Sequence Number (ISN) and the process is outlined in algorithm 1 algorithm.

1. Adjusting the IP ID Field: The IP ID was modified to include covert signaling. Specifically, the 2nd and 3rd digits of the decimal representation of the ID were adjusted to be prime numbers. A helper function ensured the IDs remained within the valid 16-bit range (0-65535) and avoided collisions with previously used IDs.

2. Encoding in TCP ISN: ASCII characters of the secret message were encoded into the least significant 8 bits of the TCP ISN. The checksum was recalculated to maintain packet integrity.

1 algorithm Pseudocode for Injection

```
for each packet in packets do
    if packet contains IP layer then
        Adjust ID:
        while ID not meeting conditions do
            Adjust ID to have prime 2nd and 3rd digits
        end while
        Set packet[IP].id = adjusted ID
        Encode ISN:
        if ASCII data remains then
            char_code = ord(next_char)
            packet[TCP].seq = (packet[TCP].seq & 0xFFFFFF00) | char_code
            Recalculate checksum
        end if
    end if
end for
```

Step 3: Extracting Features Post-injection, the PCAPs were processed to extract features. Byte-level data of these features were stored in .bin files for later conversion into images. Each .bin file contained 784 bytes corresponding to a 28x28 image.

Step 4: Image Creation A python script converts .bin files into images. Each file was read as a sequence of 784 bytes and reshaped into a 28x28 array as shown in algorithm 2 algorithm. Using the Python Imaging Library (PIL), these arrays were saved as .png images.

2 algorithm Pseudocode for Image Creation

```
1: for each bin_file in bin_files do
2:     Read 784 bytes from bin_file
3:     Reshape bytes into a 28x28 numpy array
4:     Create a grayscale image from the array using PIL.Image
5:     Save the image to the output directory
6: end for
```

The outlined process demonstrates the workflow for preparing data from raw PCAPs to images suitable for training CNN models. The next section explains the motivation behind choosing this covert channel method and describes the logic for injecting packets in such a way.

11.2 Defining a covert channel method

Research done by Kartik Umesh Sharma describes a high-bandwidth mechanism for transmitting hidden information using the identification number field in the TCP/IP packet header [34]. This approach involves embedding covert data in the payload of TCP packets, where the presence of such data depends on specific conditions in the identification number field. For example, only packets with prime numbers in certain parts of the field are considered to contain covert data. This altered identification number in a packet works like a pointer for the receiver to look for hidden information in that packet exactly. As described in this paper Identification number follows specific rules so altering this field without caution could alert a traffic warden. However, there are some ways that this approach could work:

Method 1. Altering the identification number only slightly to not raise suspicion in a short flow.

Method 2. A covert message sender could flood the receiver with packets and when the identification number naturally reaches its value where conditions meet the requirements (e.g. second and third digit of identification field is prime) include covert information into that packet in a chosen way[35] .

Method 3. As described in Section 9 covert information could be injected in the first packet of every three-way handshake of a TCP connection and the identification field could be used as a decryption key for the encrypted covert message in an ISN field. Because the first connection uses a sort of random identification number the warden is not alerted.

In the following experiment of this work, a covert channel that uses a sort of mix of the first two methods will be used and given for the CNN model to detect. The third method was not considered because the focus of this work is to analyze flows of packets and resetting connections generates a new flow thus the images would be very small and a lot of test data would need to be artificially generated.

In this research, a covert channel is proposed where the identification fields of a .pcap are altered only and only slightly and if needed to meet conditions where the second and third digits of the identification number are prime imitating a legitimate flow as close as possible and resembling the Second method. This would indicate that the least bytes of a raw Sequence Number (ISN) contain a covert message.

Experiments will also be conducted to see how the model performs on single field alterations and will convince that single field alterations are easy to detect due to certain patterns that exist for them.

11.3 Experiments

11.3.1 Simple one field covert channel

A convolutional neural network (CNN) for classifying network traffic images as either benign or injected was tested. Specifically, the model processes 28×28 grayscale images—each representing a flow of network packets—through a series of Conv2D layers with ReLU activation and MaxPooling operations. After the final pooling layer, the resulting feature maps are flattened and fed into one or more fully connected Dense layers for deeper representation learning. A final softmax output layer then produces a binary classification. This straightforward CNN architecture is trained using the Adam optimizer and a sparse categorical cross-entropy loss function, aiming to capture spatial patterns in the transformed packet data that differentiate benign traffic flows from those containing covert injection. The model was trained for five epochs, achieving an accuracy of 94% and a loss of 0.1656, with additional epochs showing improvement and reached 99% . Next, to extend the experiment of a simple CNN model analyzing different protocols and various injection methods for covert channel classification was implemented.

11.3.2 Covert channel of multiple packet fields

The developed Convolutional Neural Network (CNN) model demonstrated exceptional performance in detecting covert communication channels embedded within network flows. The model, trained over 50 epochs on a dataset comprising 10,000 benign and 6,000 malicious images of network flows, achieved a test accuracy of 98.78%. These flows were processed from network packets with malicious data containing altered TCP Identification and ISN fields to serve as covert channels. The classification results include a high precision (0.98) and recall (0.99) for malicious flow detection, reflected in an F1-score of 0.98. The confusion matrix highlights only minor misclassifications, with 31 benign flows identified as malicious and 10 malicious flows misclassified as benign, out of 3,353 test instances.

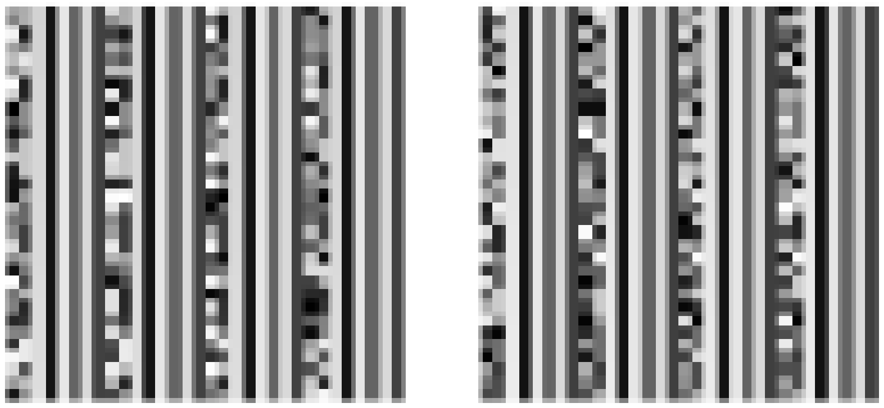
11.3.3 Storage covert channel detection conclusions

As expected CNN had no problem classifying network traffic images as benign or with covert data injected in IP and TCP packet fields. No major alterations were made to the simple CNN model to achieve these results. As reviewed earlier these header fields usually follow a concrete pattern and therefore any deviations are expected to be seen especially by advanced classification techniques like CNN and image processing. For future work extending these experiments for multiple class classification or real-time detection methods might be a potential direction to further improve storage covert channel detection.

12 Covert timing channel experimental Data and Preparation

12.1 Data conversion to images approach

The process of converting timing data to images is approached by raw timing data, such as packet arrival times extracted from PCAP files. This raw data is later transformed into grayscale images. Each PCAP file's timing data is preprocessed by padding incomplete flows and arranging them into fixed dimensions of 32x32. These arrays are then reshaped into 2D grids and saved as grayscale images, where pixel intensity encodes timing characteristics as shown in figure 11 figure. . This representation allows convolutional neural networks (CNNs) to leverage their strength in image recognition to detect patterns indicative of covert timing channels (CTCs) and distinguish between benign and malicious traffic.



11 figure. *Image data sample*

12.2 Defining a covert channel method

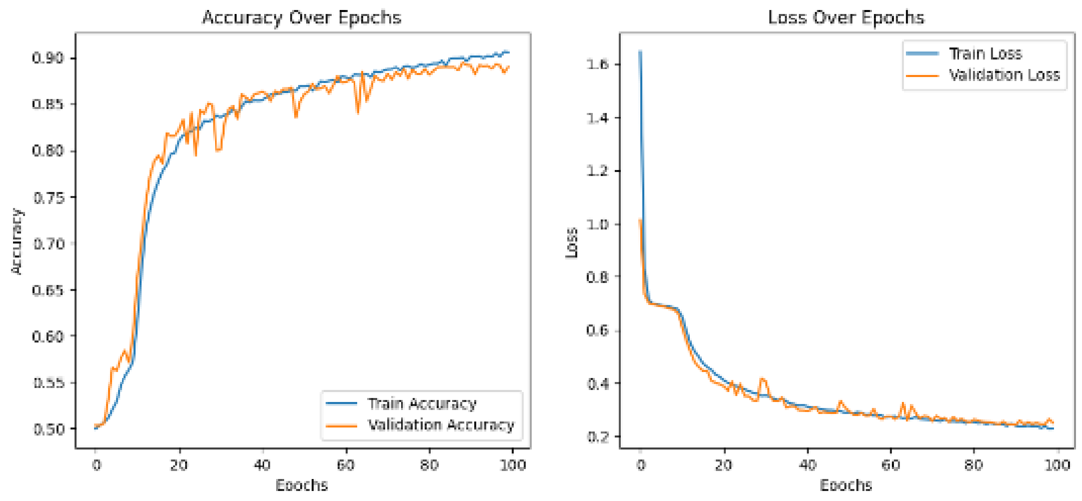
Cautious Covert Timing Channels (CCTC) attempt to partially imitate the delay times of overt traffic, making this type of covert channel harder to detect. CCTC represents a more advanced type of cyberattack that closely imitates overt traffic behavior, particularly in terms of packet inter-arrival times. To add complexity, a random packet within the flow is selected as the starting point for the covert channel, simulating real-world scenarios where covert transmissions do not necessarily begin with the first packet of a flow. In this setup, 8 bytes of data are injected into a 128-packet flow, with the packet delay set to half the mean inter-arrival time of overt traffic. This delay, when combined with the inherent delay introduced by the network, results in traffic patterns that exhibit minimal abnormalities. Such near-overt behavior blends covert transmissions seamlessly into legitimate communication, making them harder to detect by security measures. A pseudocode is presented in algorithm 3 algorithm

3 algorithm CCTC injection algorythm

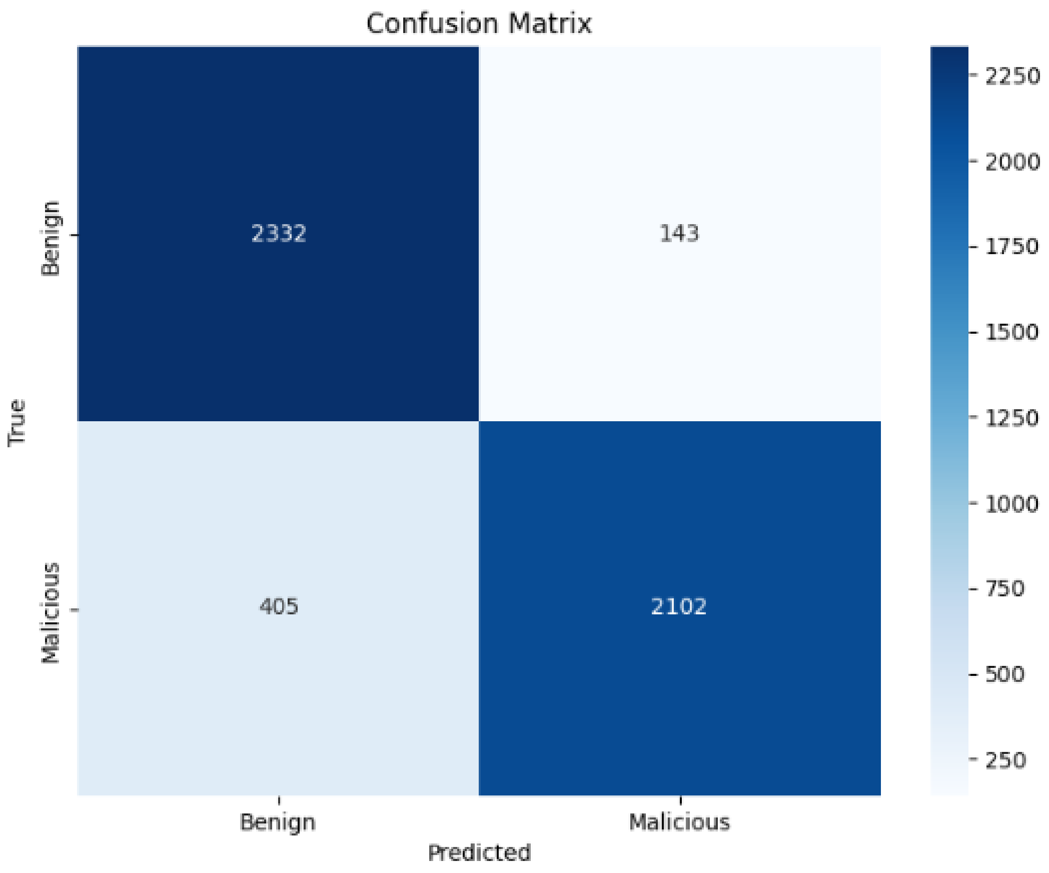
```
$delta\_min \leftarrow mean\_inter\_arrival$
$delta\_covert \leftarrow mean\_inter\_arrival/2$
for each packet $pkt$ in $pkts$ do
    if $pkt$ is in injection window(8 bytes = 64 pkts) then
        if $last\_time$ is None then
            Initialize $last\_time$ with $pkt.time$
        else
            $bit \leftarrow payload\_bits[start\_index$ mod length(payload_bits)$]$
            if $bit = 1$ then
                $pkt.time \leftarrow last\_time + delta\_covert$
            else
                $pkt.time \leftarrow last\_time + delta\_min$
            end if
            $last\_time \leftarrow pkt.time$
            $start\_index \leftarrow start\_index + 1$
        end if
    end if
end for
for each packet $pkt$ in $pkts$ do
    Recalculate checksum and length fields for $pkt$
end for
```

12.2.1 Covert timing channel detection

In this experiment, a convolutional neural network (CNN) was employed to classify grayscale images representing inter-arrival times of network packets, aimed at detecting covert timing channels (CTCs). The dataset consisted of 12k malicious flows and 12k benign flows, with each malicious image embedding 8 bytes of covert data within 128-packet flows. The delay time used to generate CTCs was set to half the mean inter-arrival time of legitimate traffic. This delay, when combined with the inherent delay enforced by the network, minimized abnormalities in the traffic pattern, making detection by conventional methods challenging. The CNN architecture featured three convolutional layers with increasing filter sizes (32, 64, and 128), each followed by max pooling and dropout layers to prevent overfitting. A fully connected layer with L2 regularization and dropout further enhanced the model's generalization ability. Hyperparameters included a batch size of 32, a learning rate of 0.0001, the Adam optimizer, and sparse categorical cross-entropy as the loss function. The model achieved a training accuracy of 95.07% and a testing accuracy of 89.44%, with validation curves showing steady convergence over 100 epochs as seen in figure 12 figure.. Despite the strong performance, confusion matrices seen in figure 13 figure. highlighted false negatives in the malicious class as a persistent challenge, suggesting room for improvement in detecting subtle covert flows.



12 figure. *Accuracy and Loss Over Epochs*



13 figure. *Test Matrix*

Results and conclusions

A novel covert channel was successfully designed and implemented using the IS-IS Hello packets, a method that has not been extensively documented in previous studies. This approach capitalizes on the unique characteristics of the IS-IS protocol, particularly its operation at Layer 2, allowing for the covert transmission of data within the padded sections of Hello packets to travel across different router areas. The experimental results demonstrated that this covert channel could effectively transmit hidden information without disrupting normal network operations or being easily detected by standard network monitoring tools. The use of padding fields in IS-IS Hello packets proved to be a highly effective medium for covert communication, offering significant bandwidth compared to other known methods. However, while this technique is innovative and effective under the controlled conditions of the experiment some simple configurations or an active warden could mitigate this threat only if the covert technique is known in the first place to which this research contributed.

The results from the machine learning experiments to detect covert channels indicate that while models like Random Forest and Neural Networks show promising performance, there are crucial areas for improvement before these models can be reliably applied in real-world scenarios. The primary challenge is that the models, despite their high accuracy in controlled environments, may not perform as well when exposed to more diverse and complex data sets. The risk of overfitting, where the models become too tailored to the training data and fail to generalize to new, unseen data, was highlighted in the experiments. To mitigate this, future work should involve the incorporation of more extensive and varied datasets that better represent real-world network conditions. This could include data from different network topologies, varying traffic patterns, and multiple types of covert channels, which would help in developing more robust detection models.

This research delves into the application of CNNs for the detection of covert timing channels (CTCs) in network communication. The experiments focused on advanced covert timing channels, presenting a robust analysis of their detection using convolutional neural networks. While the results did not outperform the benchmarks established in prior studies[12], this research proposed a more sophisticated and complex implementation of CTC, pushing the boundaries of what covert communication detection models can achieve.

The results demonstrate that the proposed model is effective and showcases significant potential, with performance metrics achieving a training accuracy of 95.07% and a testing accuracy of 89.44%, with validation curves showing steady convergence over 100 epochs. The model’s ability to process diverse covert timing channel configurations underscores its potential for scalability and application in real-world scenarios.

Overall, the research contributes valuable insights into the detection of covert channels, particularly in the use of neural networks. However, to translate these findings into practical applications, further development is necessary. Future work will explore several routes to enhance the detection capability of CNNs like: model optimization, diverse covert channels, integration with hybrid models, real-world data testing, and countermeasure design.

References and sources

[1] Y. F. Abdallah, H. H. O. Nasereddin, Y. F. Abdullah. *Proposed Data Hiding Technique-Text under Text Special Issue Proposed Data Hiding Technique-Text under Text*. Technical report 3. 2013. URL: `https://www.researchgate.net/publication/281118462`.

[2] M. A. Ayub, S. Smith, A. Siraj. "A protocol independent approach in network covert channel detection." In: *Proceedings - 22nd IEEE International Conference on Computational Science and Engineering and 17th IEEE International Conference on Embedded and Ubiquitous Computing, CSE/EUC 2019*. Institute of Electrical and Electronics Engineers Inc., 2019, pages 165–170. ISBN: 9781728116631. `https://doi.org/10.1109/CSE/EUC.2019.00040`.

[3] F. R. Alsenaid. "A DEEP LEARNING BASED APPROACH TO DETECT COVERT CHANNELS ATTACKS AND ANOMALY IN NEW GENERATION INTERNET PROTOCOL IPV6." In: (2020).

[4] L. Ashiku, C. Dagli. "Network Intrusion Detection System using Deep Learning." In: *Procedia Computer Science*. Volume 185. Elsevier B.V., 2021, pages 239–247. `https://doi.org/10.1016/j.procs.2021.05.025`.

[5] *ATNAC, Australian Telecommuncation Networks and Applications Conference, 2006 : proceedings, Melbourne 4-6 December 2006*. [Australian Telecommunication Networks & Applications Conference], 2007, page 455. ISBN: 0977586103.

[6] S. Bistarelli, A. Imparato, F. Santini. "A TCP-based covert channel with integrity check and retransmission." In: *International Journal of Information Security* (2024). ISSN: 16155270. `https://doi.org/10.1007/s10207-024-00879-z`.

[7] S. Cabuk. *Network Covert Channels: Design, Analysis, Detection, and Elimination*. Technical report.

[8] CISCO. *Understand Open Shortest Path First (OSPF) - Design Guide*. 2023. URL: `https://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/7039-1.html#toc-hId-1790473421`.

[9] Cisco. *IP Routing: RIP Configuration Guide Chapter: Configuring Routing Information Protocol*. 2014. URL: `https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_rip/configuration/15-mt/irr-15-mt-book/irr-cfg-info-prot.html`.

[10] CNSSI. *Committee on National Security Systems Committee on National Security Systems (CNSS) Glossary*. Technical report. 2022. URL: `https://www.cnss.gov.`.

[11] S. Al-Eidi, O. Darwish, Y. Chen, G. Husari. "SnapCatch: Automatic Detection of Covert Timing Channels Using Image Processing and Machine Learning." In: *IEEE Access* 9 (2021), pages 177–191. ISSN: 21693536. `https://doi.org/10.1109/ACCESS.2020.3046234`.

[12] S. Al-Eidi, O. Darwish, Y. Chen, M. Maabreh, Y. Tashtoush. "A deep learning approach for detecting covert timing channel attacks using sequential data." In: *Cluster Computing* 27.2 (2024), pages 1655–1665. ISSN: 15737543. `https://doi.org/10.1007/s10586-023-04035-5`.

[13] M. A. Elsadig, A. Gafar. *Covert Channel Detection: Machine Learning Approaches*. 2022. https://doi.org/10.1109/ACCESS.2022.3164392.

[14] M. A. Elsadig, A. Gafar. "An ensemble model to detect packet length covert channels." In: *International Journal of Electrical and Computer Engineering* 13.5 (2023), pages 5296–5304. ISSN: 20888708. https://doi.org/10.11591/ijece.v13i5.pp5296-5304.

[15] Y. A. Farrukh, S. Wali, I. Khan, N. D. Bastian. *SeNet-I: An approach for detecting network intrusions through serialized network traffic images*. Technical report.

[16] C. G. Girling. "Covert Channels in LAN's." In: *IEEE Transactions on Software Engineering* SE-13.2 (1987), pages 292–296. https://doi.org/10.1109/TSE.1987.233153.

[17] M. Handley, V. Paxson, C. Kreibich. *Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics*. Technical report. 2001. URL: https://www.researchgate.net/publication/221260544.

[18] D. Hintz. *Covert Channels in TCP and IP Headers*. Technical report. URL: http://guh.nu.

[19] J. Hu, L. Shen, S. Albanie, G. Sun, E. Wu. "Squeeze-and-Excitation Networks." In: (2017). URL: http://arxiv.org/abs/1709.01507.

[20] I. You, K. Yim. "Malware obfuscation techniques: A brief survey." In: *Proceedings - 2010 International Conference on Broadband, Wireless Computing Communication and Applications, BWCCA 2010*. 2010, pages 297–300. ISBN: 9780769542362. https://doi.org/10.1109/BWCCA.2010.85.

[21] iso.org. *Information technology-Telecommunications and information exchange between systems-Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)*. Technical report. 2002.

[22] A. Klein. "Subverting Stateful Firewalls with Protocol States (Extended Version)." In: (2021). URL: http://arxiv.org/abs/2112.09604.

[23] K. Man, Z. Qian, Z. Wang, X. Zheng, Y. Huang, H. Duan. "DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels." In: *Proceedings of the ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2020, pages 1337–1350. ISBN: 9781450370899. https://doi.org/10.1145/3372297.3417280.

[24] A. Martey, S. Sturgess. *IS-IS Network Design Solutions*. Technical report. 2002.

[25] A. Martey, S. Sturgess. *IS-IS Network Design Solutions*. Technical report. 2002.

[26] W. Mazurczyk, A. Houmansadr, K. Szczypiorski, S. Wendzel, S. Zander. *INFORMATION HIDING IN COMMUNICATION NETWORKS Fundamentals, Mechanisms, Applications, and Countermeasures*. Technical report.

[27] W. Mazurczyk, S. Wendzel, M. Chourib, J. Keller. "Countering adaptive network covert communication with dynamic wardens." In: *Future Generation Computer Systems* 94 (2019), pages 712–725. ISSN: 0167739X. https://doi.org/10.1016/j.future.2018.12.047.

[28] Mehdi Chourib. *Detecting Selected Network covert channels using machine learning*. IEEE, 2019. ISBN: 9781728144849.

[29] A. Ovadya, R. Ogen, Y. Mallah, N. Gilboa, Y. Oren. *Cross-Router Covert Channels*. Technical report.

[30] W. Rawat, Z. Wang. *Deep convolutional neural networks for image classification: A comprehensive review*. 2017. `https://doi.org/10.1162/NECO{\_}a{\_}00990`.

[31] T. Schmidbauer, S. Wendzel. "SoK: A Survey Of Indirect Network-level Covert Channels." In: *ASIA CCS 2022 - Proceedings of the 2022 ACM Asia Conference on Computer and Communications Security*. Association for Computing Machinery, Inc, 2022, pages 546–560. ISBN: 9781450391405. `https://doi.org/10.1145/3488932.3517418`.

[32] M. Schneider, D. Spiekermann, J. Keller. "Network Covert Channels in Routing Protocols." In: *ACM International Conference Proceeding Series*. Association for Computing Machinery, 2023. ISBN: 9798400707728. `https://doi.org/10.1145/3600160.3605021`.

[33] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich, T. Tsunoda. "DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture." In: *Scientific Reports* 9.1 (2019). ISSN: 20452322. `https://doi.org/10.1038/s41598-019-47765-6`.

[34] K. U. Sharma, A. U. Sharma. *High Bandwidth Covert Channel using TCP-IP Packet Header*. Technical report. 2016. URL: `https://www.researchgate.net/publication/312170838`.

[35] F. Tommasi, C. Catalano, A. Caniglia, I. Taurino. *COTIIP: a new covert channel based on incomplete IP packets*. Technical report.

[36] G. Wang. *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications : proceedings : 29 December 2020-1 January 2021, Guangzhou, China*. IEEE Computer Society, Conference Publishing Services, 2021. ISBN: 9781665403924.

[37] S. Wendzel, L. Caviglione, W. Mazurczyk, A. Mileva, et al. "A Generic Taxonomy for Steganography Methods." In: (). URL: `https://patterns.ztt.`.

[38] Y. Zhang, L. Wang, A. Afanasyev, L. Zhang. "Similar yet different: Protocol design choices in IS-IS and OSPF." In: *Asian Internet Engineering Conference, AINTEC 2017*. Association for Computing Machinery, Inc, 2017, pages 24–30. ISBN: 9781450355513. `https://doi.org/10.1145/3154970.3154974`.

[39] S. Zillien. *Reconnection-based Covert Channels in Wireless Networks*. Technical report. URL: `https://github.com/NIoSaT/WiFi_Reconnection_CovertChannel`.

[40] S. Zillien, S. Wendzel. "Weaknesses of Popular and Recent Covert Channel Detection Methods and a Remedy." In: *IEEE Transactions on Dependable and Secure Computing* 20.6 (2023), pages 5156–5167. ISSN: 19410018. `https://doi.org/10.1109/TDSC.2023.3241451`.

[41] M. Zuppelli, L. Caviglione. “PcapStego: A Tool for Generating Traffic Traces for Experimenting with Network Covert Channels.” In: *ACM International Conference Proceeding Series*. Association for Computing Machinery, 2021. ISBN: 9781450390514. `https://doi.org/10.1145/3465481.3470067`.