Characterization and Detection of Cross-Router Covert Channels

Oren Shvartzman, Adar Ovadya, Kfir Zvi, Omer Schwartz, Rom Ogen, Yakov Mallah, Niv Gilboa, and Yossi Oren

Ben-Gurion University of the Negev, P.O.B. 653, Beer-Sheva, 8410501 Israel.

Abstract

In covert channel attacks, an adversary seeks various means to influence a tangible characteristic of a system, and then makes the systems leak information by measuring this characteristic. Covert channels are, by nature, very elusive. This makes it very difficult to identify them and defend against attacks that use these channels to leak sensitive information. Thus, they are a serious threat to the security of many systems.

In this paper, we present two network timing covert channel attacks, and a defense mechanism against them. The purpose of the proposed attacks is to leak sensitive information between two logically separated (or isolated) networks that are hosted by a single router – one that is connected to the Internet, and another that is isolated and contains sensitive information. The attacks build on the fact that the response time of the router for a specific type of packet sent from a device that is connected to it is usually predictable, given the network topology. By interacting with the single shared router in a specific manner, an attacker can increase the router's packet response time. The interaction is determined based on the information to be leaked, so the receiver (a computer located on the Internet-connected network) can measure the delayed packet response times and decode the sender's signals (which operates from the isolated network) to receive classified information. The two classes of attacks presented in this work differ in the way that the delay is caused. In the cross-router covert channel (CRCC) attack, the sender overloads the router with control-plane packets; in the Wi-Fi micro-jamming attack, the sender uses a pre-installed implant to transmit single-tone signals in the 2.4GHz frequency range, disturbing the router's packet transmissions. We showed that both attacks can influence a wide range of router brands and Wi-Fi capable devices and evaluated the optimal settings for both attacks when using different types of packets, transmission power, and data transmission rates.

Our proposed defense mechanism is based on semi-supervised machine learning and deep learning algorithms, which are both used for novelty detection in network traffic. By detecting unusual traffic, we can identify the disturbances needed to leak the information. The system can then respond by blocking the suspicious devices that are involved in the attack. We evaluated the attacks in noisy and noise-free environments, successfully detecting both attacks in both environments. All the data and code, which includes the implementation of the attacks and the defense mechanism, is published as a benefit to the research community.

Keywords: Covert Channels, Network Intrusion, Novelty Detection, Machine Learning, Deep Learning.

1. Introduction

Network separation and isolation are important components of many organizations' security policies. The goal of such policies is to prevent network intrusion and information leakage by separating sensitive network segments from other segments of the organizational network and from the Internet. Sensitive traffic (or data) sent over such network segments may include mission-critical business documents, control data for industrial systems, and private health records. Less sensitive data may include multimedia streams, environmental sensor readings, and data related to smart home devices.

The above-mentioned network isolation can also be extended to the networked devices themselves. While some devices are protected from security risks by their owners and manufacturers, either through careful administration or by using automatic updates, other networked devices, such as Internet of Things (IoT) nodes [1] and medical devices [2], are difficult or impossible to patch, and are at greater risk of malware infection. It is particularly important to isolate such less-secure networked devices from other devices on the network.

A common network isolation approach is to separate one physical network into multiple logical networks. Many routers provide this functionality by splitting the network into a host network and a guest network. The router discards any traffic traveling between one network and the

Email address: `orenshva@post.bgu.ac.il, adarov@post.bgu.ac.il, zvikf@post.bgu.ac.il, omershv@post.bgu.ac.il romog@post.bgu.ac.il, mallah@post.bgu.ac.il, gilboan@bgu.ac.il, yos@.bgu.ac.il` (Oren Shvartzman, Adar Ovadya, Kfir Zvi, Omer Schwartz, Rom Ogen, Yakov Mallah, Niv Gilboa, and Yossi Oren)

other, enforcing separation as long as nodes on the two networks do not connect to a common node on the Internet.

Logical isolation is not only common in practice, but it is actively recommended as a security measure. For example, the U.S. National Institute of Standards and Technology (NIST) [3] recommends isolating industrial control system components, which typically have monolithic software installations that are difficult to upgrade and maintain, into dedicated network segments, isolated from the main corporate information technology (IT) network. Based on this recommendation, the U.S. Department of Veterans' Affairs (VA) created the medical device isolation architecture (MDIA) [4], which mandates the use of software-based mechanisms to isolate medical devices and prevent their traffic from entering the hospital's networks of Veterans' Affairs hospitals.

In this work, we present two attacks that use network timing covert channels to break logical network separation and exfiltrate data in a setup that includes two logically isolated networks supported by a single router. One of the networks is not connected to the Internet and handles sensitive information, while the other network is a public Internet-connected network. In the proposed attacks, we assume that the attacker has successfully installed malware on at least one machine in each network, and that the malware has already obtained the desired sensitive information from the isolated network (see Section 4.1 for more details). The malware in the isolated network will be referred as the *sender*, and the malware in the public network will be referred as the *receiver*. In both attacks, the sender uses network timing covert channels as a means of leaking the sensitive information to the receiver. The attacks differ in the way that the information bits are signaled by the sender to the receiver. In the first attack, called the cross router covert channel (CRCC) attack [5], the sender overloads the router with control-plane packets according to the leaked information bits. In the second attack, called the Wi-Fi micro-jamming attack [6], the sender uses an implant capable of transmitting single-tone signals in the 2.4GHz frequency range to signal the leaked information bits.

To exemplify the general process of the attacks, Figure 1 presents the above-mentioned scenario.

As shown in the figure, the sender, which operates from the isolated guest network, has collected some sensitive data (for example, a personal health-related sensor reading) and would like to leak this data. Overt communication between the isolated and public networks is blocked, so the data is leaked through a covert channel. The data is decoded in an infected computer in the public host network (the receiver), and from there the data is leaked to the Internet.

We evaluated the impact of our attacks by performing them on various routers and Wi-Fi enabled devices. Also, for the CRCC attack, evaluation included performing the CRCC attack with different packet types; And for the Wi-Fi micro-jamming attack, the evaluation included measuring

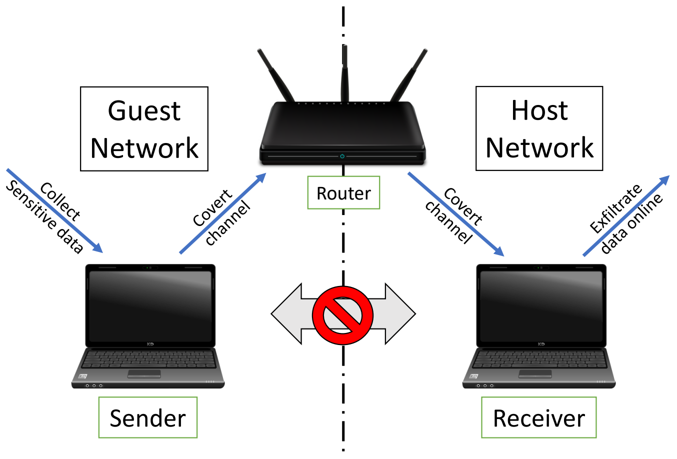


Figure 1: A covert channel between a host and guest network: overt traffic is blocked, but the covert channel is not.

the optimal configuration for the attack (like the implant's transmission power, the micro-jamming frequency, and more) and the maximal transmission rate.

On the defensive side, we present a machine-learning (ML) and deep-learning (DL) based defense mechanism to detect the attacks. We use semi-supervised models to perform novelty detection. This kind of model assumes that the training data is benign and tries to predict whether the test data is indicative of malicious activity. Such models were selected, based on the assumption that it would be easy to obtain benign data that does not include this type of attack. The defense mechanism evaluation included performing the attacks in a noisy and noise-free environment. The noise, also referred in this paper as *cover traffic*, included packets that were not related to the attack but can often be found in a standard network environment (like ARP packets, DNS packets, TCP packets, and more). The evaluation metrics used for the models were F1-score and AUC.

As mentioned above, the attacks presented in this paper assume that malware is installed on at least one computer in each network. Also, the Wi-Fi micro-jamming attack relies on a hardware implant that can transmit unmodulated signals in the 2.4GHz frequency range. These limitations will make it hard for an inexperienced attacker or an attacker without access to the isolated network to perform this attack (see Section 4.1 for more details). But when considering highly skilled attackers, it is very important to be aware of the limitations of software-based network isolation, given both the adoption of inexpensive and relatively insecure IoT devices and organizations' increasing need for secure IT infrastructure.

In this paper, we make the following contributions:

- We characterize two network timing covert channels, which enable data leakage between a public host network and an isolated guest network. The router, which hosts both networks, is used as a shared medium that enables the covert channels.

- We implemented a defense mechanism that can be placed between a network's router and its clients. The defense uses ML/DL techniques to detect information leaks from the network timing covert channels. The ML-based defense mechanism can be enabled without training on data that contains malicious behavior.
- We publish our data and code as a benefit to the research community [7]. The code includes the implementation for both the attacks and the defense mechanism.

The rest of the paper is organized as follows. In Section 2 we describe the attacks' implementation, evaluation methods, and results. Next, in Section 3 we describe the defense mechanism's implementation, evaluation methods, and results. Finally, we conclude in Section 4 with a discussion that includes a comparison between the attacks, the performance of our proposed defense mechanism, attacks limitations, defense limitations, related work and directions for future research.

1.1. Network Covert Timing Channels

Covert channels, first defined in 1973 by Lampson in [8], are communication channels that exist between two parties, a sender and a receiver, and can be used when overt communication between the parties is prohibited due to privilege separation, sandboxing, or other architectural boundaries. Zander et al. [9] defined two main types of covert channels: direct and indirect. A direct covert channel describes the case in which the two parties operate an innocuous-looking overt communication channel that contains a hidden covert channel. An indirect covert channel describes the case in which an overt communication channel between the parties does not exist. In this case, the two parties establish a covert channel using shared hardware.

Timing-based covert channels are a type of indirect covert channel. In this channel, communication is performed by measuring patterns in time, and more specifically delay patterns caused by interfering with the system resources. Maurice et al. have successfully established an indirect covert channel between virtual machines running on different cores [10]. They exploited the inclusive feature of caches, which is a shared resource among the machines, allowing a core to evict lines in the private first level cache of another core. By measuring access times to the cache, the receiver notices which bit was received.

In a network traffic context, network timing covert channels transfer information using packet arrival patterns and not by the actual content of the packet. Tian et al. [11] described two construction technologies of network timing covert channels: communication content covert channels and transmission network covert channels. Communication content covert channels encode the information into time behavior of the network traffic, which is basically an information steganography technique. Transmission network covert channels change the structure of the transmission network (for example, adding proxy servers to the network) to send information. Tian et al. also determines three metrices of a covert channel quality: message and identity concealment, channel robustness, and transmission efficiency (e.g., transmission rate).

An example of a communication content covert channel can be found in Maurice et al. [12] work that showed a timing channel that relies on measuring packet arrival times or inter-packet delays. In this channel, the sender sends a packet to send the bit '1' and sends nothing to send the bit '0'. The packets are sent at constant intervals so the receiver can synchronize with the sender's transmissions and be prepared to receive (or not receive) a packet at predetermined intervals to decode the secret information. The timing channel they presented uses a direct channel to convey information since it is performed on an overt communication channel. El-Atawy et al. [13] presented another content communication direct covert channel, a sorting channel. In this channel, the sender sends a group of packets in a specific order to the receiver. Each packet permutation is equivalent to a unique piece of information. Using a pre-shared code, the receiver decodes the secret information by observing the packets' permutation.

Wustrow et al. [14] presented a transmission network covert channel which used the end-to-middle proxy technique. This technique is meant to overcome IP blocking. In an end-to-end proxy, users access remote servers through an accessible proxy server. This makes the proxy server a single point of failure which can be configured to block certain servers' IP addresses to censor their content. Using the end-to-middle proxy technique, a user can communicate with a friendly ISP that will transfer all of the user's traffic to an unblocked server unless the traffic includes a special stenographic tag. From the censor's point of view, the user is still unable to access the blocked site.

In this research, we examine indirect timing covert channels, which are achieved by having the sender selectively exhaust the finite hardware resources available on the router. Then, the receiver measures the effect of this varying resource consumption on the router's performance, specifically the router's response time to the receiver's requests (e.g., communication content covert channels). Blocking this form of covert channel, which exhausts hardware resources, is more difficult than network timing covert channels, since it may require architectural changes to the router.

1.2. Network Covert Timing Channels Detection

As Tian et al. [11] shows, there are several methods to detect network covert timing channels. Overall, the methods can be considered as anomaly detection methods. Anomalies are rare data instances in which one or more of the features has an abnormal value compared to most data instances. Anomaly detection is aimed at identifying anomalies within the data.

Network covert channel detection heavily relies on constant monitoring of the network. The monitoring process parameterizes the network's characteristics, which are then analyzed by a network intrusion system. This helps the system create a model of the network's normal behavior which can be used to predict anomalies when they occur. Over time, the system obtains new information which is used to train the model and improves its ability to detect such anomalies.

For communication content channels, one can use statistical methods to capture the shape, regularity and randomness of the network traffic data or use ML/DL methods that specialize in anomaly detection. To perform statistical-based anomaly detection, the network traffic is recorded and a profile representing its statistical behavior is created. The profile includes parameters like traffic rate, number of different IP addresses, connection rate, and more. To detect an anomaly, one must build a profile of regular network traffic, which is a recording of prior network traffic that the user trusts; this profile is then compared to the current real-time network traffic profile. This comparison enables alerts to be raised when the examined network traffic profile contains an abnormal network event. To perform ML/DL-based anomaly detection, the network traffic is recorded, and features are extracted. The features are later used to train a ML/DL model, which will be used for anomaly detection. Clustering and outlier detection are combined in one method which uses ML/DL models to identify anomalies. In this approach, unsupervised ML/DL algorithms gather similar data instances together to create clusters. The algorithm determines the similarity between data instances using a distance measure in the feature space. Data instances that do not belong to any cluster are called outliers, and such outliers are the anomalies we wish to detect.

For transmission network channels, the detector can embed a watermark to each packet in the network traffic for easier tracing of the traffic's path. Another method involves unique identification of the traffic's data using regular expressions. This can allow easy identification of packets that include a special tag like in the middle-to-end proxy technique. Finally, ML/DL can be used to identify communication coming from the proxy server.

In this research, we use semi-supervised ML/DL models to perform novelty detection. As Miljković [15] explained, novelty detection is an anomaly detection process where the ML model learns the system's normal behavior. Later, based on this knowledge, the model should be able to predict any abnormalities. This is considered a semi-supervised model because the training data is guaranteed not to include any abnormalities. This contrasts with the unsupervised case mentioned above, in which the training data may contain anomalies.

2. Covert Channel Attacks

As Yi et al. [16] describes, the two main elements in the router responsible for packet delivery are the software-based routing plane and the hardware-based forwarding plane. When the router receives a packet, the software-based routing plane calculates the forwarding table (FIB). Then, the packets are delivered through the forwarding plane, following the rules of the FIB. The forwarding plane is usually implemented in hardware operates at the nanosecond time scale. The router's routing plane is implemented in software executed by a traditional CPU and operates at the millisecond or second timescales.

Guided by this explanation, we attempt to create covert channels that are not operating on the faster forwarding plane but rather on the slower routing plane. We do so by generating traffic which the router does not simply forward, but rather must respond to in software. While devices on the host network may not have a wide variety of ways of interacting with the router's control plane, we claim that even the most secure router must expose a minimal set of router control plane functions to the guest network to function properly, notably the Dynamic Host Configuration Protocol (DHCP), Address Resolution Protocol (ARP), and Domain Name System (DNS).

2.1. Attacks Prerequisites

As explained in Section 1, the attacks presented in this paper aim to create a covert channel between an isolated ("guest") network and an Internet-connected ("host") network, which are hosted by the same router. The computer in the guest network which sends the sensitive data (the "sender") causes delays in the router's response times according to the leaked data bits. The computer in the host network (the "receiver") records and analyzes the router's response times to decode the data.

These attacks assume that both computers, in the guest and host networks, are infected by malware that operates the attacks. Moreover, in the Wi-Fi micro-jamming attack, it's also assumed a malicious implant is installed in the sender's machine. These requirements are further discussed in Section 4.1.

Since the attacks rely on deliberately delaying the router's response time, they should have the same transmission window length. We'll explain the meaning of this term with an example. If the sender wishes to send the bits '101' to the receiver, it must delay the router's responses for some limited time, then stop the disruptions, and after that continue the disruption. To do that, the sender can disturb the router for one second, stop for one second, and finally continue the disruption for another one second. The receiver should know that the sender works in one-second intervals to decode the message correctly. Those intervals are the transmission window length. Note that the transmission window length determines the leaked bitrate. In the example, the sender uses a one second transmission window, so the leaked bit rate is one bit per second.

2.2. Cross-Router Covert Channel

The cross-router covert channel (CRCC) attack exploits the fact that routers have limited resources. Given this limitation, a router that supports different logically separated networks, could be overflowed with routing plane packets from one network and therefore perform poorly (with delayed responses) for another network. The delays in the response time are measurable, and therefore can be used as a network covert timing channel.

2.2.1. Attack theoretical basis

A denial-of-service (DoS) attack is performed by malicious actors that wish to degrade the performance of network services. The attack is done by transmitting a large amount of data to the network server to exhaust its limited bandwidth or to interrupt the router's ability to forward packets efficiently. According to Murillo Piedrahita et al. [17], the data sent to the server/router can be random data or carefully crafted data which resemble legitimate traffic in large volume.

In our case, the CRCC attack uses a DoS attack which overflows the network's router to make it underperform. We assume that the router supports two logically separated networks: a public host network (connected to the Internet) and an isolated guest network. The attacker wants to leak sensitive information from the guest network to the host network (see Figure 1). By inflicting intentional delays in the router's responses, the sender of the CRCC attack can signal information to the receiver, thus creating a network covert timing channel.

2.2.2. Attack process

The attack is performed as follows. First, as mentioned earlier, we assume that the attacker installed malware in the sender machine (that is part of the isolated network) which collected the sensitive information prior to the attack and can leak the information using the CRCC attack. It is also assumed that the receiver (that is part of the public network) is infected with malware that can receive information from a CRCC sender. The attack will start at a predetermined time, which is known to both the sender and the receiver. Before the attack starts, the receiver sends a few packets to the router (without loss of generality, we assume the packets are ARP requests), measures the response times, and stores the average. This measurement will be referred to as the normal response time. Then, the attack begins. Take, for example, a case in which the sender sends DHCP discover packets, and the leaked information transmission rate is one bit per second (e.g., the transmission window length is one second). To leak sensitive information, the sender will operate as follows. To send the '0' bit, it does not transmit anything for a period of one second. To send the '1' bit, it transmits a large amount of DHCP discover packets to the router at a high transmission rate. During this time, the receiver continuously sends ARP requests and measures the router's response time.

Assuming the sender and receiver shared the same transmission window length, the receiver can collect the measured response times and divide them by one second intervals. For each interval, if the average response time is close to the normal response time, the slot represents the '0' bit. If the average response time is much higher than the normal response time, the slot represents the '1' bit.

2.2.3. Methodology

Initial results about the impact of the CRCC attack we proposed were presented at the 2019 WOOT workshop by Ovadya et al. [5]. We attempted to attack as many router models from multiple vendors and at price points as possible. The goal was to test which of the routers are vulnerable to the CRCC attack. Table 1 shows the tested router models.

To prepare each router for experimentation, we first inspected its online documentation, both on the official vendor website and on the OpenWRT website, which contains hardware information for many router models. Next, we performed a factory reset on the router and updated it with the most recent firmware version we were able to find on the vendor's website. Then, we used the router's web-based management interface to enable the router's host and guest isolation feature and connected two different computers respectively to the router's host and guest networks. Finally, we checked that the isolation feature worked in principle by verifying that naive direct connections between the two computers were blocked by the router.

To determine whether a specific router is vulnerable to the CRCC attack, we applied a Student's Independent two-sample t-test on the outputs of the receiver; if the test can differentiate between two sets of 1,000 timings, obtained either with or without the sender, with a significance of $p < 0.05$, the router under test is vulnerable to the CRCC attack. Many cross-router covert channels, which differed in terms of the type of packets used, were tested on each router. The tested packet types included ARP, SSH, CSRF, ICMP, and DHCP packets.

In addition to observing the attack's effect on different kinds of routers, we addressed the covert channels based on three criteria. The first and most significant criterion is the channel's pervasiveness: how widespread is the channel among the various types of hardware and how difficult would it be to block this channel using a simple software upgrade. The next criterion is the channel's rate: how much data can be transferred over this channel per unit of time within a reasonable data transmission rate. The last criterion is the channel's degree of covertness: how similar is traffic sent using this channel to regular traffic exchanged by the router and how hard is this channel to detect using forensic tools which examine log files and other external artifacts. These criteria match Tian et al.'s [11] covert channel quality metrices.

The experimental setup for the CRCC attack consisted of two Raspberry PIs (RPIs) which acted as the sender and receiver, a router, and a test harness computer, as shown in Figure 2. The two RPIs (model: RPI 3 Model B, revision 1.2) were connected to the router via Wi-Fi in different logically separated networks. Both RPIs were

Identifier	Vendor	Model	CPU type	Core count	CPU speed	Year introduced	Price
TP1	TP-Link	Archer C3200	Broadcom BCM4709A0	2	1 GHz	2015	$218
TP2	TP-Link	Archer C2	MediaTek MT7620A	1	580 MHz	2017	$63
DL1	D-Link	DIR-882	MediaTek MT7621A	1	880 MHz	2017	$154
DL2	D-Link	DIR-825AC	Realtek RTL8197DN	1	660 MHz	2015	$50
ED1	Edimax	RG21S	MediaTek MT7621AT	2	880 MHz	2017	$209
ED2	Edimax	BR-6208AC	Realtek RTL8881AQ	1	520 MHz	2014	$47
LS1	Linksys	EA7500-eu	Qualcomm IPQ8064	2	1.4 GHz	2016	$185

Table 1: Evaluated Routers in the CRCC attack [5]

controlled remotely by the test harness computer, which was connected to them with an Ethernet cable. Using the test harness computer, we logged onto the RPIs using an SSH session and operated the Python attack scripts (receiver and sender). Both scripts use the Scapy library for packet manipulation and tcpdump for network traffic recording.

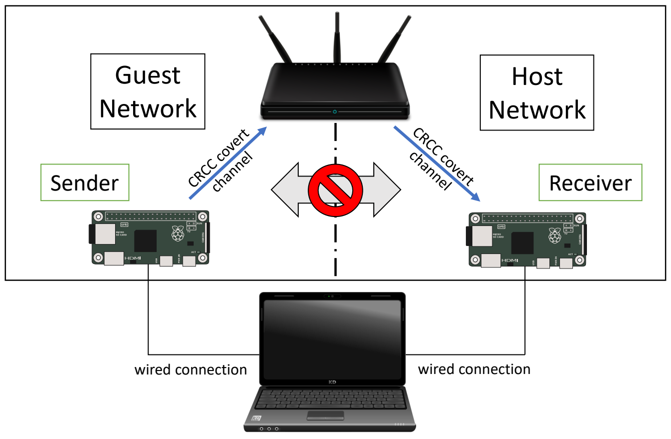


Figure 2: Cross-Router Covert-Channel experimental setup.

The receiver script could send packets at a specified slow rate, record the network traffic, and analyze the router's response times. The sender script could send packets at a high transmission rate in a structured manner to leak the sensitive information to the receiver. Both scripts needed to be started manually to enable their simultaneous execution. After both scripts were done transmitting, the receiver script decoded the leaked information and printed the leaked information and the bit error rate (BER).

2.2.4. Attack Results

We found that all the routers examined were vulnerable to the CRCC attack, but not all routers are vulnerable to every covert channel. Table 2 summarizes the assessed covert channels. The ARP-SSH is the worst covert channel compared to other types of channels since it only allowed one-way communication on two routers (out of seven). The best covert channel is the DHCP-ARP channel that successfully operates on all the routers and allows two-way communication in five out of the seven tested routers. Also, we tested the covert channels' pervasiveness, rate, and covertness. Table 3 summarizes the results.

The SSH-ARP channel is one of the most covert of the timing-based channels we identified, as it generates no log-file entries since the SSH connection establishment never concludes. We still consider it less pervasive than the other timing-based covert channels due to the limited number of routers with default support for the SSH protocol. The two ARP-based channels, ARP-CSRF and ARP-ARP, are part of the most pervasive channels in our opinion, since virtually all routers expose some sort of web server on their host network side, and all routers support the ARP protocol. The CSRF-ARP channel is slightly less stealthy since the thousands of web requests per second may constitute an irregular access pattern which can be detected by external intrusion detection systems. Both ARP-based channels are limited in their rate because ARP packets are easily handled by the router's CPU and generate only a minimal resource footprint. The ICMP-ICMP channel is both more covert and stealthier than the ARP-ARP channel, but its pervasiveness is limited by the fact that not all routers expose ICMP on the guest network side. Finally, the DHCP-ARP channel is very pervasive and can be operated in high rate but is not so covert since lots of DHCP requests can be considered irregular network pattern which can be detected by a network intrusion system (similarly to the ARP-CSRF channel).

The vulnerability reports for the various covert channels and router models were granted the following CVE IDs: CVE-2019-13263, CVE-2019-13264, CVE-2019-13265, CVE-2019-13266, CVE-2019-13267, CVE-2019-13268, CVE-2019-13269, CVE-2019-13270, and CVE-2019-13271. We note that the workshop paper also examined direct covert channels, which are not within the scope of this article.

2.3. Wi-Fi Micro-jamming

The Wi-Fi micro-jamming attack shares the CRCC attack's concepts of degrading the router's performance and leaking data through response time measurement, but it influences the router in a different way. The disturbance to the router is caused by transmitting a weak single-tone signal in the Wi-Fi frequency band (2.4GHz). This signal lets the attacker exploit the CSMA/CA mechanism

Channel	**Type**	**TP1**	**TP2**	**DL1**	**DL2**	**ED1**	**ED2**	**LS1**
ARP-SSH	Timing	$G \Rightarrow H$	$G \Rightarrow H$	–	–	–	–	–
ARP-ARP	Timing	$G \Leftrightarrow H$	$G \Leftrightarrow H$	–	$G \Leftrightarrow H$	$G \Leftarrow H$	$G \Leftrightarrow H$	$G \Leftrightarrow H$
ARP-CSRF	Timing	$G \Leftrightarrow H$	$G \Rightarrow H$	$G \Leftrightarrow H$	$G \Rightarrow H$	$G \Leftrightarrow H$	$G \Rightarrow H$	$G \Rightarrow H$
ICMP-ICMP	Timing	–	–	–	–	$G \Leftrightarrow H$	–	$G \Leftrightarrow H$
DHCP-ARP	Timing	$G \Leftrightarrow H$	$G \Leftrightarrow H$	$G \Leftrightarrow H$	$G \Leftrightarrow H$	$G \Rightarrow H$	$G \Leftarrow H$	$G \Leftrightarrow H$

Table 2: Covert channels supported by different routers, taken from [5]. The arrow direction describes the possible flow of the data between the guest (G) and host (H) networks, while the column headings discuss different router models.

Channel	**Pervasiveness**	**Rate**	**Covertness**
ARP-SSH	++	++	+
ARP-ARP	+++	+	+++
ARP-CSRF	+++	+	++
ICMP-ICMP	++	++	++
DHCP-ARP	+++	++	+

Table 3: Quality of different covert channels [5]

employed by the 802.11 protocol, to increase the amount of time it takes the router to respond.

2.3.1. CSMA protocols and Wi-Fi

As defined by Kurose and Ross in [18], carrier-sense multiple access (CSMA) protocols employ random algorithms in order to access the data link layer when a number of devices share the same medium to communicate. Common examples of CSMA protocols are the wired IEEE 802.3 Ethernet protocol and the wireless IEEE 802.11 Wi-Fi protocol.

According to the collision avoidance (CA) variant of the CSMA protocol specification, before a station transmits, it first goes through a carrier sense phase, where it senses the status of the medium. When the medium is sensed as busy, the station will wait for a certain amount of time (which is called distributed coordination function inter-frame space, or DIFS, in the Wi-Fi protocol) until the channel becomes free again before transmitting. This precaution by itself does not guarantee collision-free access to the medium due to propagation delays and the hidden terminal problem [18]. To ensure that a transmitted Wi-Fi packet was received without errors, the receiving station sends an acknowledgment frame (ACK) every time a packet is received without errors. If the transmitting station does not receive an ACK for a transmitted frame, it will attempt to retransmit it several times, until it gets acknowledged by the receiving station. If, after several attempts, the sending physical layers do not receive an ACK, it will discard the packet and alert the higher layer protocols[1].

The Wi-Fi protocol provides two different methods of detecting whether a channel is busy or clear. As specified in Subsection 17.3.10.6 of the 2016 version of the Wi-Fi standard [19], the Wi-Fi station performs both a carrier sense-based clear channel assessment (CS/CCA), in which it looks for Wi-Fi traffic on the channel, and a generic energy-detection-based assessment (CCA-ED), in which it looks for any kind of energy on the Wi-Fi band.

2.3.2. Attack theoretical basis

Jamming is a denial-of-service (DoS) attack that takes advantage of a shared medium in electronic communication. Jamming is performed by transmitting signals that interfere with the ability to communicate on the medium.

Stations communicating using the 802.11 protocol will wait until the medium is free for the DIFS time period to transmit their pending frames. Therefore, if a transmitter continuously transmits on a channel in one of the 802.11 frequency bands, none of the devices communicating on the channel will be able to transmit. Such jamming will result in a denial of service to any wireless networks using that channel, usually causing the devices communicating on the channel to indicate an error condition and ultimately abandon the channel altogether.

Our approach, initially presented at 2018 WOOT workshop by Ogen et al. [6], is to use micro-jamming to cause transmission delays in the frames transmitted on a wireless network. In contrast to traditional jamming techniques, which result in various degrees of denial of service, micro-jamming does not aim to perform a DoS attack. Instead, in a micro-jamming approach, jamming is started and stopped frequently. Because communication is not entirely blocked, transmitted frames may be delayed or re-transmitted, but they are rarely discarded. As a result, this method's impact on the usability and throughput of the overall channel is minimized.

An example of micro-jamming is presented in Figure 3, which presents actual recordings of network traffic between a laptop and a wireless router captured using a Tektronix RSA604 real-time signal analyzer. A simple DNS transaction carried out over Wi-Fi is presented at the top of Figure 3. The transaction begins when a higher layer of the protocol stack creates a packet (in this case, a DNS query) and asks the Wi-Fi physical layer to transmit it. The Wi-Fi controller on the laptop performs a CCA check, which is immediately successful in this case, and transmits the packet over the air, as indicated by the first energy band on the left. Very shortly after this event, the Wi-Fi physical layer on the router successfully receives the packet

[1]An optional RTS/CTS access mode also exists in Wi-Fi protocol, but it is not commonly deployed and is outside of the scope of this paper.

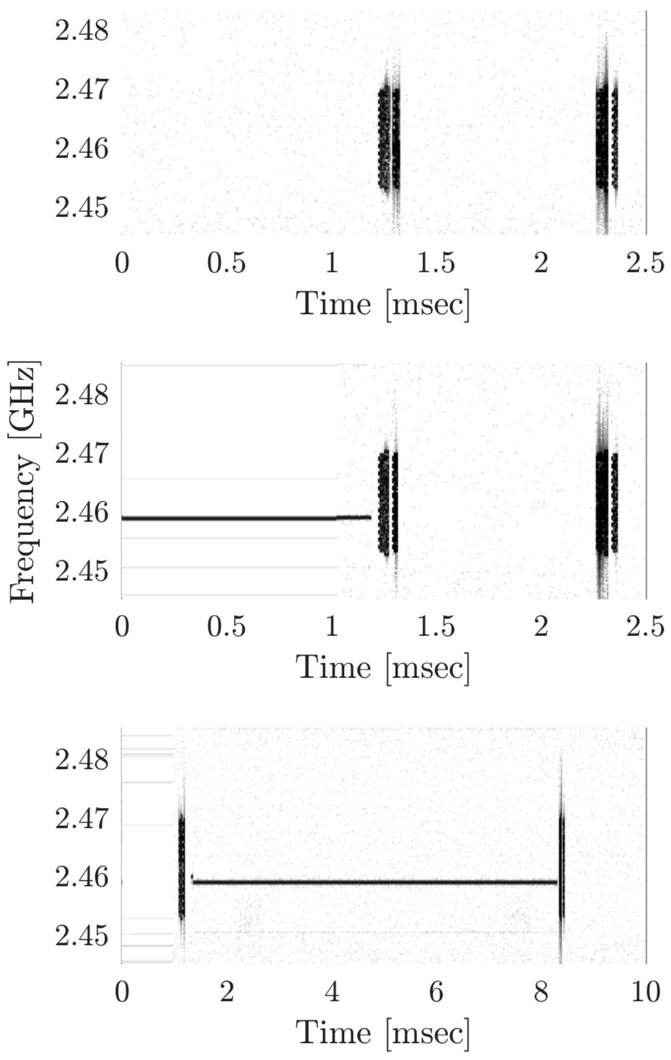


Figure 3: Wi-Fi micro-jamming effect [6]. Top: A DNS query and response without jamming. Middle: A DNS query delayed by micro-jamming. Bottom: A DNS response delayed by micro-jamming.

and immediately sends an ACK packet back to the laptop. In the figure this can be seen as a more powerful energy band which immediately follows the laptop's packet. Then, the router performs the operations required to satisfy the DNS query by the upper protocol level, either by handling it locally or by sending it to another machine. After the DNS response is ready, it is now the router's turn to perform CCA, after which it sends a DNS response back to the laptop. This DNS response is indicated in the figure as a higher-energy band. Finally, the laptop receives this DNS response and immediately acknowledges it.

The middle part of Figure 3 presents an actual micro-jamming scenario in which the higher-level protocol layers request the Wi-Fi stack to transmit a DNS query, while the shared medium is experiencing the effect of micro-jamming (this is seen in the figure as a thin solid line). As soon as the micro-jamming stops, the laptop's physical layer waits for an additional short period of time (DIFS) and then immediately transmits the DNS query it stored; it then behaves normally, like the upper part of Figure 3 which was explained above. It is important to note that in this case the application layer on the laptop will detect a longer round-trip delay before receiving the DNS response, since the DNS query was not immediately sent to the shared medium.

The bottom part of Figure 3 shows a third scenario in which the laptop was able to transmit the DNS query to the router as soon as it arrived from the higher protocol layers, but the router experienced micro-jamming as it was preparing to send its answer to the laptop. As seen in the figure, the router delays sending its DNS response until the jamming stops. This can be seen in the figure as the long solid line separating the DNS request (and associated ACK) from the DNS response (again with associated ACK).

2.3.3. Attack process

As shown in Figure 4, the Wi-Fi micro-jamming attack model consists of a malicious implant, which serves as the data sender; an attacker, which serves as the receiver in a remote Internet-connected location; and a victim device, which browses a website with some attacker-controlled content using a Wi-Fi access point.

Figure 5 presents a high-level diagram of the implant. The implant contains two components: the analog modulation component and the radio frequency (RF) transmission component. In the analog modulation component, the data to be exfiltrated is first modulated using pulse amplitude modulation or PAM at frequency f_J to create short breaks in the jamming signal which allow some traffic to go through. Then, in the RF transmission component, the waveform is then used to modulate a sine wave at a central frequency of one of the Wi-Fi channels that is ready for transmission. Before the transmission, a duty cycle D_J is determined which is the proportion of time in each of the sine wave's cycles in which jamming is performed. Finally, the resulted waveform is amplified and transmitted over the air.

On the receiver side, the target device is induced to load attacker-controlled web content which includes a small amount of JavaScript code. This code causes the target device to issue DNS requests to non-existent domains. The code then measures the time it takes for an error response to arrive from the DNS server. In practice, this is done by creating an HTML image element and setting its source to an image hosted on an invalid domain. We chose this method of probing the network, since DNS requests require only one UDP packet per request and one packet per error response, in contrast to common web traffic which is carried over TCP and includes an additional round trip for connection setup.

Similarly to a CRCC attack, when the malicious implant wants to send a logical '1' to the victim device, it activates micro-jamming on the Wi-Fi channel, causing an increase in the round-trip time (RTT) that is measurable in JavaScript.

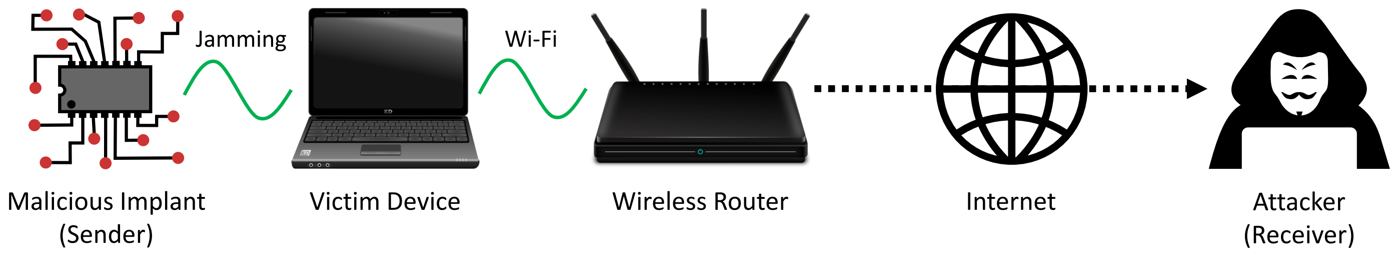


Figure 4: The general Wi-Fi Micro-jamming attack model.

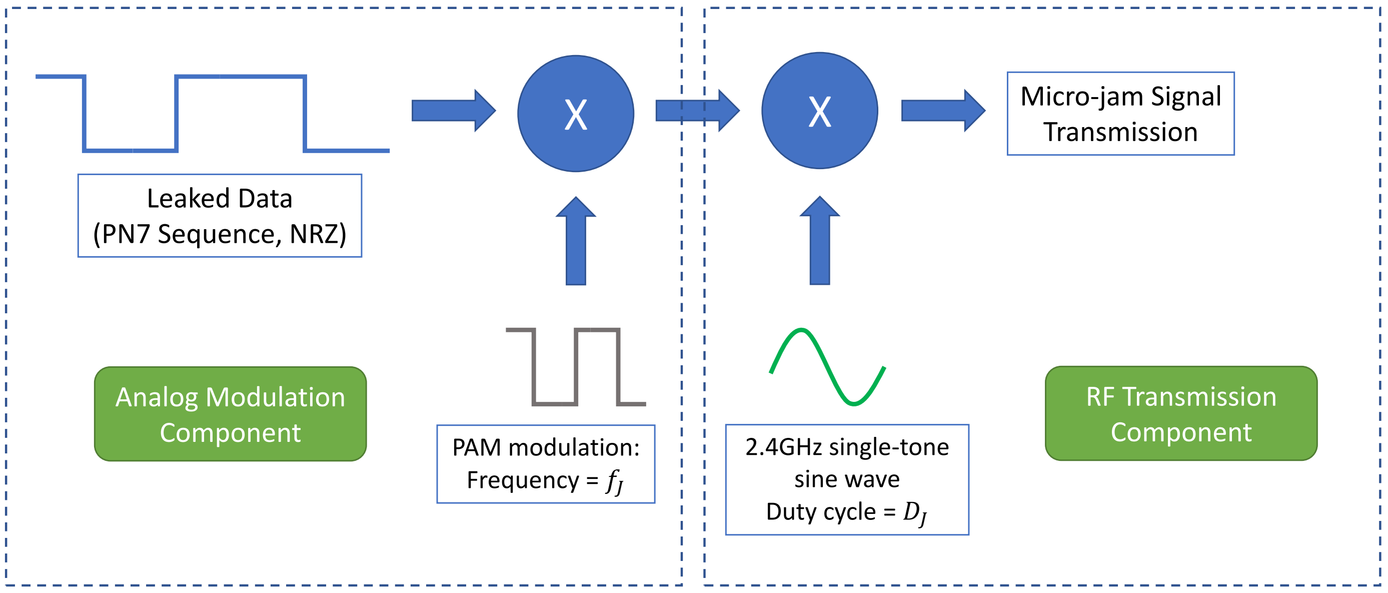

Leaked Data (PN7 Sequence, NRZ)
X
X
Micro-jam Signal Transmission
Analog Modulation Component
PAM modulation: Frequency = f_J
2.4GHz single-tone sine wave Duty cycle = D_J
RF Transmission Component

Figure 5: Wi-Fi Micro-jamming implant.

2.3.4. Methodology

Our initial results, presented by Ogen et al. [6] in the WOOT workshop, demonstrated the significant impact of this covert channel on different devices. We examined whether the Wi-Fi micro-jamming attack causes response time delays that can be measured and used to reliably leak data. We also performed the attack with different values for its parameters (such as the data rate, micro-jamming frequency, duty cycle, range, and transmission power) to find the attack's optimal configuration settings.

Figure 6 describes the experimental setup used. An Atmel ATMEGA256RFR2 Xplained Pro board and a Keysight 33622A 120 MHz waveform generator were used as the attack's malicious implant [20]. The waveform generator can create various signals with different modulations and configurations. The ATMEGA256RFR2 microcontroller can generate raw unmodulated signals in the 2.4 GHz frequency band at a variety of frequencies and output levels.

We used the waveform generator's pre-configured PN7 pseudo-random sequence as the data to be leaked using the Wi-Fi micro-jamming covert channel. This data was modulated by PAM modulation with frequency f_J and was partially transmitted to the ATMEGA256RFR2 microcontroller's external I/O pins according to the configured duty cycle D_J. A simple program compiled for the Atmel microcontroller allowed us to turn the jamming signal on and off according to the data coming from the waveform generator. In this setup the waveform generator acted as the analog modulation component and the Atmel microcontroller acted as the RF transmission component.

For the receiver, a wide variety of different devices were used, including phones, laptops, tablets, and even an IoT node. A TP-Link WR940N 450Mbps wireless N router served as the victim Wi-Fi access point, and a local server functioned as the local DNS server of the LAN managed by the TP-Link router. The server hosted a local webpage which contained malicious JavaScript code that periodically sent DNS requests to non-existent URLs and measured the RTT of the DNS message. When the channel was interfered with, the RTT was comparatively higher than when there was no interference.

2.3.5. Attack Results

We successfully performed the Wi-Fi micro-jamming attack with several different victim devices, thus concluding that all Wi-Fi capable devices are susceptible to this kind of attack. Moreover, we showed that the power requirements are very low, with the active transmitter consuming less than 20 microwatts for transmission - power levels which are low enough to be utilized by miniature battery-operated

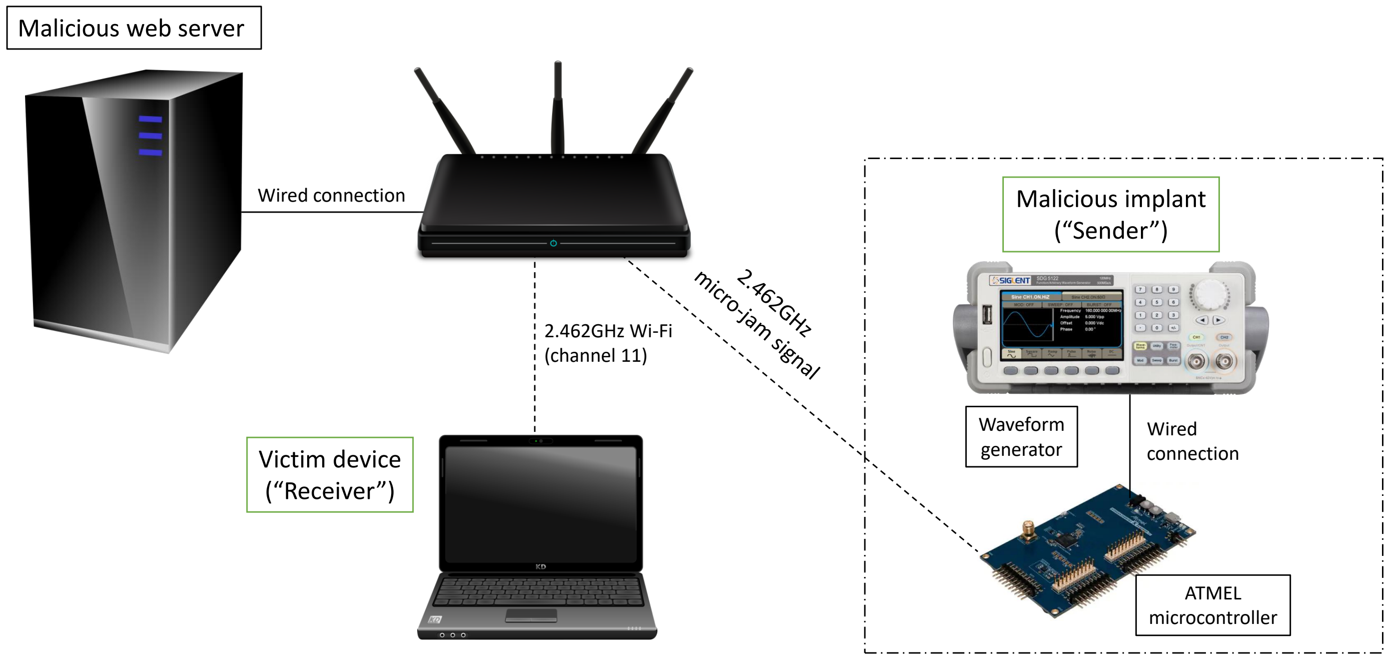

Malicious web server
Wired connection
2.462GHz Wi-Fi (channel 11)
2.462GHz micro-jam signal
Malicious implant ("Sender")
Waveform generator
Wired connection
ATMEL microcontroller
Victim device ("Receiver")

Figure 6: Wi-Fi micro-jamming experimental setup.

implants. Also, the analysis of the BER of the leaked information as a function of the micro-jamming frequency and duty cycle is shown in Figure 7.

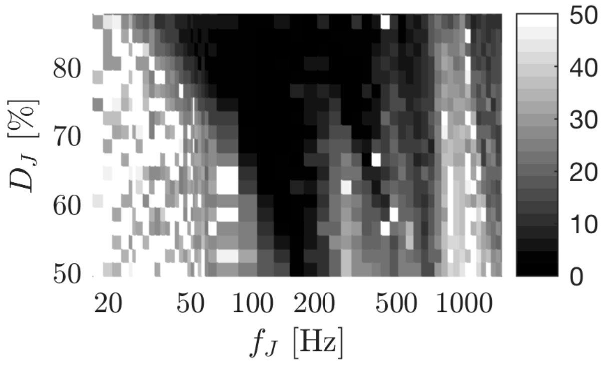


Figure 7: Bit error rate (BER) as a function of micro-jamming frequency and duty cycle (DC).

Figure 7 shows that in terms of the BER, the optimal duty cycle is in the range of 80-88% and in terms of the micro-jamming frequency, the optimal frequency is in the range of 100-200Hz. We've also measured the Wi-Fi's data throughput as a function of duty cycle. The results can be seen in Figure 8. It can be seen in the figure, that 80% duty cycle decreases the Wi-Fi's throughput from 50Mbps to 30Mbps. Thus, the attacker should consider that operating the Wi-Fi micro-jamming attack with its optimal values can be noticeable.

Finally, we found that the receiver can be located over 15 meters away from the sender and can receive data at a rate of 40 bits per second (bps). The workshop paper also examined another Wi-Fi micro-jamming method that relies on the backscatter phenomenon; however, this method is not within the scope of the current paper.

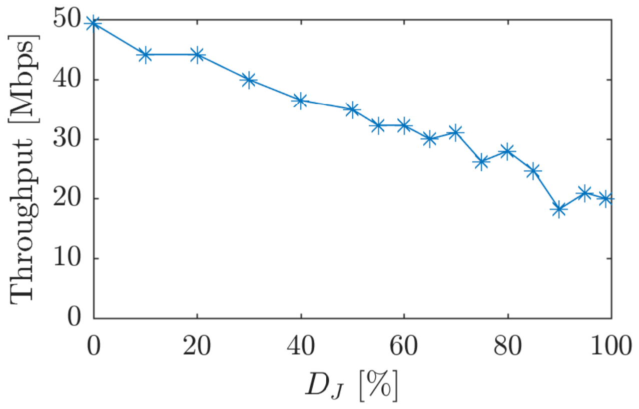


Figure 8: Wi-Fi throughput as a function of micro-jamming duty cycle (DC).

3. Covert Channel Defense

This section describes the approach we took to detect the attacks described in Section 2. As explained earlier, both attacks can leak information between two logically separated networks. Based on this, we assumed that the defense mechanism should be able to see all the network traffic, as the router sees it, to identify cross-network leakage. We also assumed that the attacks represent a low percentage of the total network traffic, and therefore we considered novelty detection models. This type of models tries to identify anomalies by learning the normal behavior of the system, and this provides a good fit for the problem at hand.

3.1. Defense Motivation

The attacks presented in this paper break logical network isolation. This isolation is crucial to organizations' efforts to avoid the leakage of sensitive information. Both attacks are difficult to detect for two main reasons. First, in contrast to malware or fuzzing attacks, the attacks presented here involve sending legitimate packets which are used for basic network management and are not uniquely associated with an attack scenario. Second, the attacks, by their very nature, change the way the network behaves. Therefore, dissecting individual packets will not help in attack detection.

We were able to overcome these difficulties by relying on an ML/DL-based defense mechanism capable of novelty detection. Relying on novelty detection algorithms makes this defense tool more easily integrable, because the user does not have to train the tool with data containing the actual attacks. In the event of an anomaly, an alert will be raised in real time, and the machines associated with the anomalies can be examined by the network administrator.

3.2. Defense theoretical basis

As discussed in Section 1.2, Tian et al. [11] noted that network covert timing channel which are communication content channels can be detected using ML/DL models that use network traffic characteristics to learn the normal behavior of the network. Those models can later identify anomalies in the network traffic, which can indicate the presence of a covert channel. Assuming that recordings of the covert channel communication are non-existent, the chosen models should be ones that don't rely on labeled malicious data to make predictions. That's why it is preferable to use unsupervised or semi-supervised models that use clustering techniques to identify outliers. The general process of developing such detection tools involves data collection, feature extraction, model training, and model testing.

In our work, the detector was developed following the steps described in Figure 9. As seen in the figure, we implemented the attacks to create a data collection platform. After that we extracted features from the data. Then, we trained our models with semi-supervised novelty detection models. Finally, we tested the models in both noisy and noise-free environments. The noise-free environment was free from any traffic unrelated to the computers performing the attack. The noisy environment included a great deal of cover traffic. *Cover traffic* is network traffic that is unrelated to the attack, and its sole purpose is to add noise in the form of common packets sent in computer networks to simulate standard network activity. Each step described here is further elaborated in the following sections.

3.3. Data Collection

To develop an ML-based anomaly detector, we needed to collect data, and to accomplish this, we created setups similar to those described in Sections 2.2.3 and 2.3.4.

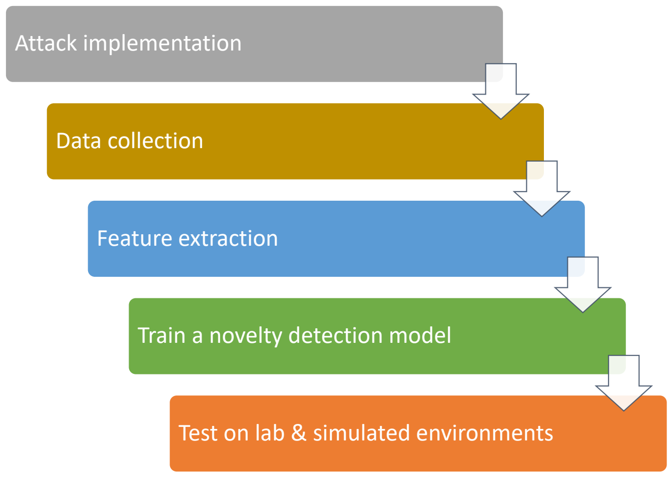

Attack implementation
Data collection
Feature extraction
Train a novelty detection model
Test on lab & simulated environments

Figure 9: The general defense process employed to develop the defense mechanism.

The setup for the CRCC attack consisted of one specific router (model: D-Link DIR-825ACG1), and the setup for the micro-jamming attack consisted of one specific victim device (Dell Inspiron 5559 laptop). The attacks were performed while the network traffic was recorded. We collected two sets of data: data without cover traffic and data with cover traffic.

The cover traffic was generated by two computers which were connected via the virtual network computing (VNC) protocol. This protocol lets the computer that hosts the VNC server see and interact with the desktop window of the computer that serves as the VNC client. To produce cover traffic, the VNC client executed a script that opened and closed a calculator or notepad program, started and then quit the Firefox browser, and transferred a large file to the VNC client using the file transfer protocol (FTP). While all the above was taking place, the VNC client played a short video in an infinite loop. These actions generated a heavy stream of TCP communication due to the rapid changes in the VNC client's screen (associated with the video and the various opening and closing programs) and the file transfer. It is important to note that the sender's packets in the CRCC attack were vastly outnumbered by the cover traffic packets, making them relatively rare samples in the general network traffic, as it would probably be in a real scenario.

For each attack and dataset, we collected 90 recordings of benign traffic (without attacks) and 10 recordings of malicious traffic, each of which was several minutes long. For the CRCC attack recordings, we randomly chose the duration (60-300 seconds), type of packets to be sent (ARP/DHCP), and the transmission rate (150-600 packets per second). The CRCC attack's leaked data bit rate was 1bps. For the Wi-Fi micro-jamming attack, we used $f_j = 500Hz$, $D_j = 88\%$, DNS transmission rate of 120 packets per second, and a leaked data bit rate of 10bps. These are the optimal configuration values for error-free transmission, as identified by Ogen et al. [6]. We did not randomize the Wi-Fi micro-jamming configurations, as described in more detail in Section 4.

3.4. Feature Extraction

The next step was feature extraction. First, we note that both attacks are similar in nature: both are network timing covert channel attacks that use delayed response times to leak sensitive data from an isolated network to a public network (which is connected to the Internet). Because of this, we extracted the same features for both attacks.

As mentioned earlier, to observe the attacks, the network's behavior must be observed rather than the packets themselves. Therefore, we decided to extract features from a flow of packets rather than from individual packets. A *flow*, according to Davis and Clark [21], is an "unidirectional sequence of packets sharing a common key such as the same source address and port, and destination address and port". The flow features we extracted can be categorized into two types: general flow features and response time features. We consider general flow features as common characteristics that are usually measured in a packet context, such as the flow duration and number of packets. Response time features are various statistical measurements of the router's response time to each of its clients separately (i.e., the router's response time measurement for each flow); these features are important for the detection of the proposed attacks, because the attacks rely on careful analysis of the delay in response times. The response time features, which took the response time of every client packet into account, required a relatively high processing time to extract, but were critical to the success of our detection mechanism. Table 4 summarizes the flow features.

General Flow Features	**Response Time Features**
Flow duration	Response time median
Number of packets	Response time variance
Number of bytes	Response time average
Transmission rate (median and variance)	Min response time
Packet length (median and variance)	Max response time
Flow to router \broadcast (bool)	

Table 4: The full list of features used in CRCC and Wi-Fi microjamming detection.

As described in Section 3.3, we collected 90 benign recordings and 10 malicious recordings of each attack; each recording lasted a couple of minutes. As said earlier, both attacks use delayed response times to send the sensitive data. The data transmission rate was 1bps for CRCC attack and 10bps for the Wi-Fi micro-jamming attack. This means that the response time changes that encode the data for the CRCC and Wi-Fi micro-jamming attack can be best observed when analyzed in one second intervals and 0.1 second intervals, respectively. Therefore, we divided the recording files of the CRCC attack into one-second files and of the Wi-Fi micro-jamming attack into 0.1 second files before extracting the features.

We define *mini-flow* as a small portion of the original flow. In other words, each flow of a one-second recording file is a mini-flow. Concatenating all of the mini-flows chronologically will result in the original flow. Note that a mini-flow is not necessarily a one second or 0.1 second flow. It is a small portion of the flow that was created because we divided the recording files into smaller files.

To extract features from the mini-flows, we took each of the original recordings (before dividing them to one second recordings) and assigned a unique ID to each flow found in the recording. Then, after dividing the flows, we extracted features from each mini-flow while the mini-flows kept their original flow ID (this was crucial for identifying a malicious flow, see Section 3.5).

3.5. Novelty Detection

We examined several novelty detection algorithms to detect the attacks: deep autoencoder, local outlier factor (LOF), and one-class SVM (OCSVM). After comparing them, we selected for each attack the method which yielded the best results in terms of the F1-score and AUC score. A deep autoencoder model was chosen to detect CRCC attacks, and a local outlier factor (LOF) model was chosen to detect Wi-Fi micro-jamming attacks. All of the results are presented in Section 3.7.

3.5.1. Deep Autoencoder (AE)

A deep autoencoder (AE) is a neural network which copies its input to its output after processing it through several smaller hidden layers. The network is trained to reduce the reconstruction error caused by transferring the input through these internal layers. Amarbayasgalan et al. [22] used this model to perform novelty detection using the following method (shown in Figure 10). First, the model is trained on benign data. Then, the model is tested on a dataset containing both benign and anomalous instances. A reconstruction error threshold is set to determine whether an instance is anomalous. Data instances with a reconstruction error above the threshold are considered anomalous (in our study, 2-norm was used to calculate the distance between the reconstructed data instances). The model can effectively detect anomalies since it was trained on just benign data. It expects the reconstruction errors of a normally behaving network to be small; any abnormality in the network's behavior will result in a data instance with a higher reconstruction error, because the model was not trained to reconstruct this kind of data instance properly.

For the evaluation of both attacks, we used a min-max scaler to scale the features to be in the [0-1] range. Then, we used a deep autoencoder with five hidden layers. The input layer consisted of 13 nodes, which was then reduced to nine, six, and four nodes. After that, the hidden layers

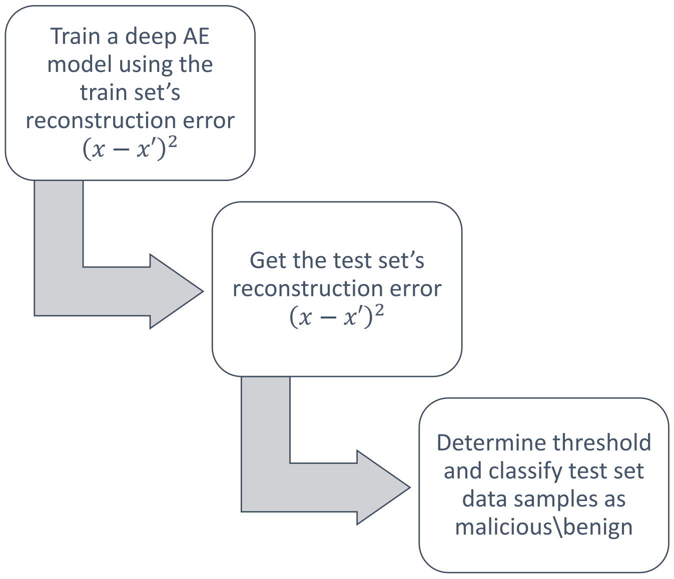

Train a deep AE model using the train set's reconstruction error $(x - x')^2$
Get the test set's reconstruction error $(x - x')^2$
Determine threshold and classify test set data samples as malicious\benign

Figure 10: Process of using deep autoencoder for novelty detection.

expanded to six and nine, and finally expanded to 13 nodes, which were used as the output layer (see Figure 11). In all the hidden layers, we used the tanh activation function as it works well on normalized data. Finally, in the output layer, we used the sigmoid activation function to make the features' values be in the [0-1] range, so it can be comparable to the features in the input layer.

3.5.2. Local Outlier Factor (LOF)

As described by Breunig et al. [23], the local outlier factor (LOF) model is an unsupervised clustering model used for anomaly and novelty detection. The LOF model computes the density around each data instance in the feature space and uses it to determine which instances are anomalous. The density is defined as the mass per unit of volume. The LOF model relies on the fact that anomalies will be positioned in low density areas in the feature space, i.e., isolated from the rest of the data instances. It calculates the local density of each data instance compared to k of its neighbors. Once it discovers an instance with substantially lower density than its neighbors, it deems the instance an anomaly.

For the evaluation of both attacks, we used a min-max scaler to scale the features to be in the [0-1] range. Then, we used the LOF model as a semi-supervised novelty detection model with $k = 1000$. This value of k let the model focus on global anomalies instead of local anomalies.

3.5.3. One-Class Support Vector Machine (OCSVM)

The one-class support vector machine (OCSVM) model is an unsupervised model that can be used for anomaly detection tasks in high-dimensional data. According to Erfani et al. [24], OCSVM maps the input data to a higher dimension space in which it's easier to distinguish between normal and anomalous data points. The mapping is determined by a kernel function that if chosen correctly, can theoretically model any non-linear pattern of normal behavior. OCSVM works best on small but feature-rich datasets and suffers from the curse of dimensionality with bigger datasets.

For the evaluation of both attacks, we used a min-max scaler to scale the features to be in the [0-1] range. Then, we used the OCSVM model as a semi-supervised novelty detection model with $nu = 0.1$. nu is the upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors relative to the total number of training examples.

3.6. Novelty Detection Process

The novelty detection process was done as follows. The same process was applied when training on noise-free data and noisy data. First, the models were trained on benign mini-flows (as done in novelty detection). Then, we used the models to detect malicious mini-flows in the test set (which included both benign and malicious mini-flows). To determine whether a specific flow was malicious, we gathered all its mini-flows (using the unique ID of their original flow) and calculated the percentage of malicious mini-flows. Values above a certain threshold, the *mini-flow threshold*, indicated that the flow should be deemed malicious. We trained the models without cover traffic, to make sure that the attacks are detectable in a noise-free environment, and then moved on to training the models with cover traffic.

Figure 12 presents the setup for the CRCC attack with cover traffic. First, the router was set up with two logically isolated networks: the public host network and the isolated guest network (above and below the dotted line, respectively). The host network consisted of the CRCC receiver and the router. The guest network consisted of the CRCC sender, the router and two additional computers responsible for generating the cover traffic (in the figure, "Cover 1" and "Cover 2"). The guest network also consisted of an access point (AP) and a man-in-the-middle (MITM) computer which served as the defense component, analyzing the traffic to and from the router to detect abnormal activity. Some of the clients were connected to the AP, so the MITM could record the traffic as it goes to and from the router[2].

Figure 13 describes the setup for the Wi-Fi micro-jamming attack with cover traffic. Like the CRCC setup, the router hosts two logically isolated networks: the host and guest (below and above the dotted line, respectively) networks which consists of all the required devices to reproduce the Wi-Fi micro-jamming attack with added cover traffic (micro-jamming sender/receiver, a router, an AP, cover traffic generators, and a MITM computer). Note that the micro-jamming receiver in the presented setup

[2]A very small number of the packets from the guest network were also received by the MITM computer and filtered out before the feature extraction step. Even though those packets were received, it is very unlikely that they had any effect due to their small amount.

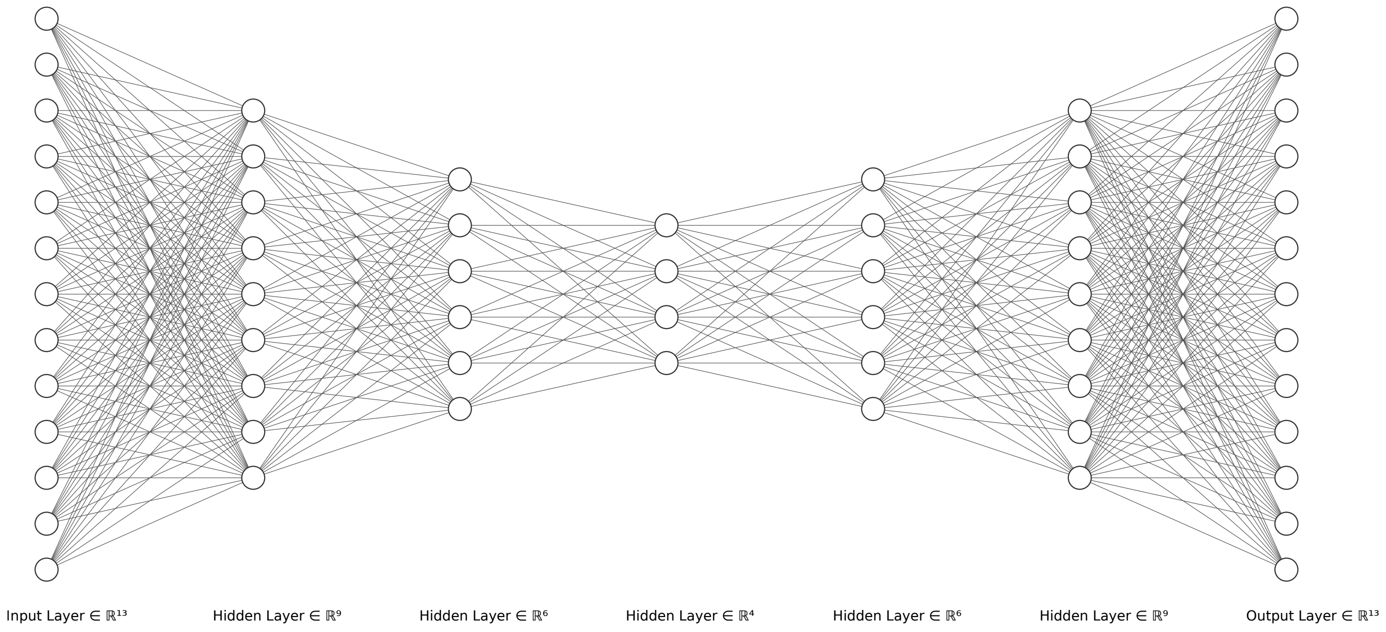


Figure 11: Deep autoencoder structure.

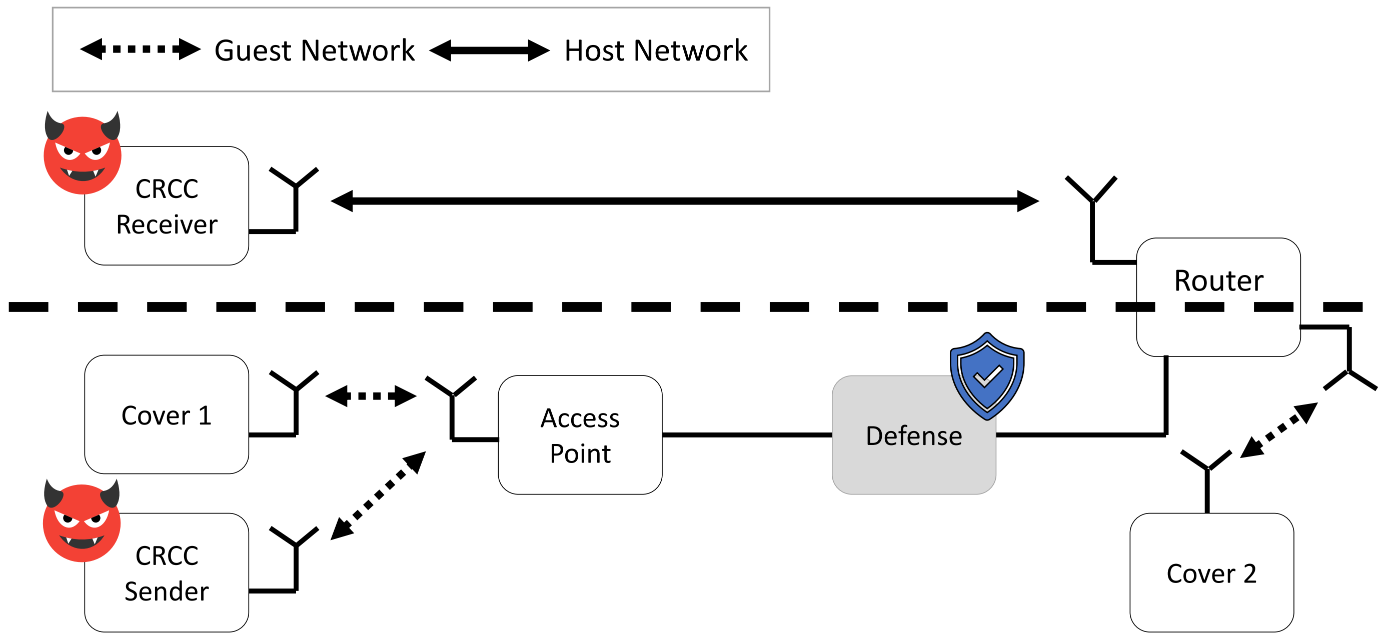


Figure 12: The CRCC attack setup with cover traffic.

is the victim device that connects to a malicious website (in Section 2.3, the receiver was described as the attacker itself). The malicious website is hosted by a local DNS server, which is not seen in the image but was part of the setup.

3.7. Defense Results

In this section, we present the results of our evaluation of the proposed defense system's performance. The models' performance was measured using three metrics: the ROC curve (with AUC score), F1-score of mini-flows predictions and F1-score of flow prediction. Note that since the dataset is imbalanced, the F1-score metric is more indicative of the model's performance than the AUC score.

As explained in Section 3.5, all of the models were responsible for predicting whether a mini-flow is malicious or not. The determination of whether a flow is malicious is based on the percentage of malicious mini-flows associated with it; if the percentage is above the mini-flow threshold, the flow is considered malicious.

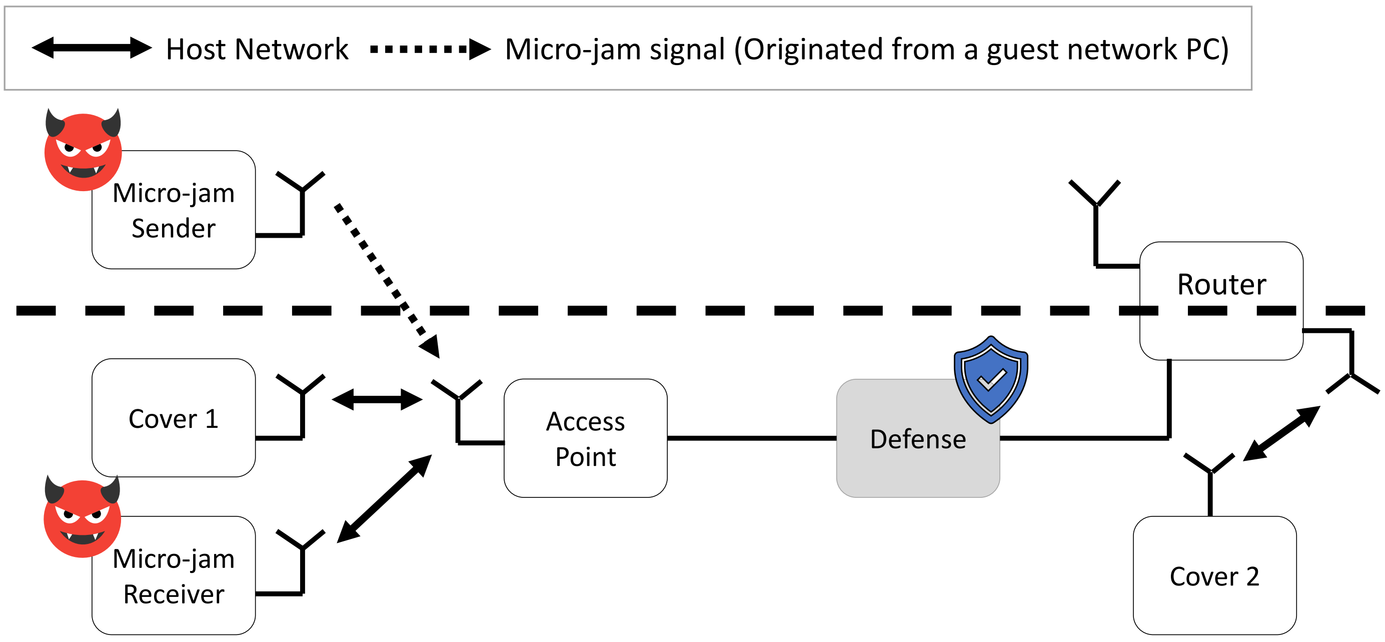


Figure 13: The Wi-Fi micro-jamming setup with cover traffic.

3.7.1. Cross-Router Covert Channel

Table 5 summarizes the results of CRCC detection with/without cover traffic by three different models: LOF, Deep AE, and OCSVM. To obtain these results, we used the models with different parameters, as described in Table 6. In the deep AE model, the AE threshold is the reconstruction error threshold.

As seen in Table 5, the deep autoencoder model outperformed the other two models in all of the evaluation metrices: AUC score, F1-score for mini-flow detection, and F1-score for flow identification. To get these results, we set the mini-flow threshold to be 2%; in other words, if 2% of the mini-flows of a specific flow are malicious, we classify the whole flow as malicious. We set the reconstruction error threshold at one (i.e., the 2-norm result between two data instances should be greater than one). The model's parameter settings were determined using trial and error and were found to yield the best results for CRCC detection with/without cover traffic. The ROC curves of the deep AE model used for CRCC detection with and without cover traffic are presented in Figures 14b and 14a respectively.

The AUC and F1-score results show that the mini-flow detection algorithm is effective in detecting malicious flows, both with and without cover traffic. In both cases the performance based on F1-score of the identification of malicious flows is good, and the F1-score of mini-flow detection improved when cover traffic was present, as discussed in greater detail in Section 4. We believe that the performance of this algorithm can be improved further by applying domain knowledge relevant to the specific network being protected.

3.7.2. Wi-Fi Micro-Jamming

Table 7 summarizes the results of Wi-Fi micro-jamming detection with/without cover traffic by three different models: LOF, Deep AE, and OCSVM. To obtain these results, we used the models with different parameters, as described in Table 8. In the Deep AE model, the AE threshold is the reconstruction error threshold.

As seen in Table 7, the LOF model outperformed the other two models in all of the evaluation metrices: AUC score, F1-score for mini-flow detection, and F1-score for flow identification. To get these results, we set the mini-flow threshold to be 30% when detecting the attack without cover traffic, and 60% when detecting the attack with cover traffic present; in other words, for example, if 30% of the mini-flows of a specific flow are malicious when detecting the attack without cover traffic, we classify the whole flow as malicious. Also, we set $k = 1000$. The model's parameter settings were determined using trial and error and were found to yield the best results for Wi-Fi micro-jamming detection with/without cover traffic. The ROC curves of the LOF model used for Wi-Fi micro-jamming detection with and without cover traffic are presented in Figures 14d and 14c respectively.

The AUC and F1-score results show that the malicious flow detection performance is satisfactory with cover traffic and excellent without cover traffic. These results are discussed in greater detail in Section 4. We believe that performance of this algorithm can be improved further by applying domain knowledge relevant to the specific network being protected.

	LOF			Deep AE			OCSVM		
	AUC	F1-s (MF)	F1-s (F)	AUC	F1-s (MF)	F1-s (F)	AUC	F1-s (MF)	F1-s (F)
No CT	0.88	0.674	0.722	0.823	0.367	0.922	0.719	0.689	0.837
With CT	0.771	0.712	0.702	0.791	0.553	0.889	0.716	0.494	0.551

Table 5: Summarized results of CRCC detection. Note that we abbreviated "mini-flow" to "MF", "flow" to "F", "cover traffic" to "CT", and "F1-score" to "F1-s".

	LOF	Deep AE	OCSVM
No CT	MF_TH = 0.3, k = 1000	MF_TH = 0.02, AE_TH = 1	MF_TH = 0.3, NU = 0.1
With CT	MF_TH = 0.6, k = 1000	MF_TH = 0.02, AE_TH = 2	MF_TH = 0.3, NU = 0.1

Table 6: Model parameters used in CRCC detection. Note that we abbreviated "mini-flow" to "MF", "threshold" to "TH", and "cover traffic" to "CT".

4. Discussion

In this paper, we presented two timing covert channel attacks that can leak sensitive information from a logically isolated network to a public network (connected to the Internet). Both attacks rely on deliberately delaying the router's response time, which is done to signal the receiver the sensitive information. Both attacks are difficult to detect, because they disturb the behavior of the network traffic rather than sending specific packets or making irregular changes in packets' fields.

We also presented an ML/DL methods to detect the covert channels. In the CRCC attack case, the results indicate high AUC scores, low mini-flow F1-scores, and high flow F1-scores. Also, the mini-flow threshold was very low - only two percent. We've seen that the model rarely makes a prediction which is false positive but suffers from lots of false negative. For this reason, the F1-score of the mini-flows was very low (the predictions suffered from low recall but high precision). But, since the number of false positives was very low, we could use a low mini-flow threshold which was very precise in determining which flow is malicious.

Overall, the CRCC attack detection was considerably good, even when it was performed in a noisy environment with random configurations. We observed that adding cover traffic improved the detection capabilities. We assume that this occurs because the traffic generated by the CRCC attack is substantially different from regular network traffic, making it easier to identify. In a noise-free environment, the only traffic that exists is that of the CRCC attack, so there is less difference between the training and test set.

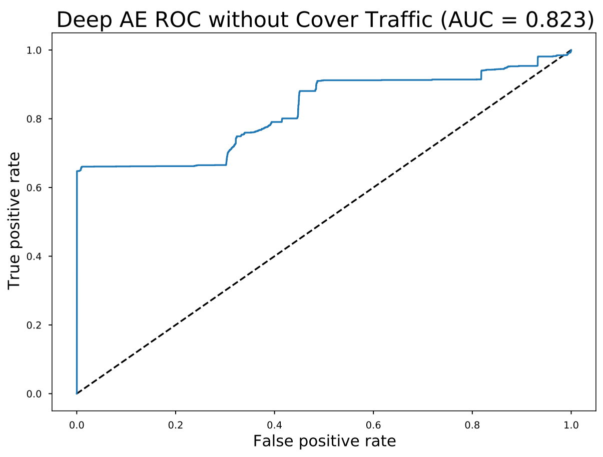
In the Wi-Fi micro-jamming attack case, the results indicate good values in all the evaluation metrices when the detection was performed without cover traffic, but when cover traffic was added, they became much worse. Also, the mini-flow threshold was very high - thirty percent without cover traffic and sixty percent with it. During the testing process, we saw the model suffers from a lot of false positives, which made us put a very high mini-flow threshold. Even with that, the large number of false positive mini-flows worsened the model's performance so it could barely detect the attack when cover traffic was present.

Overall, we could detect the attack in both the noisy and noise-free environments. Contrary to the CRCC attack, the model's performance declined when cover traffic was added. The reason for that is that the Wi-Fi micro-jamming attack, as opposed to CRCC, uses fewer packets, and transmits at a lower rate. Essentially, the attack's traffic is not distinct as the traffic of the CRCC attack. For this reason, the cover traffic masked the attack's traffic and made the model suffer from many false positive predictions.

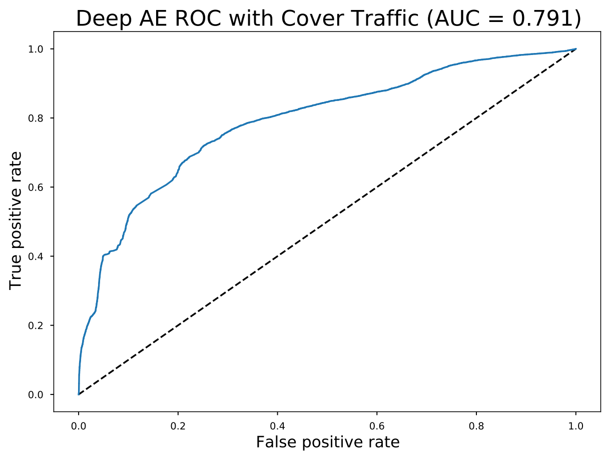
Also, as shown in Section 3.3, we configured the Wi-Fi micro-jamming attack with the optimal values for the attack's success and didn't change them while collecting data. This was done because when we experimented with different configuration values, it was difficult to transmit a message with no errors. So, we wanted to be sure that the test data only contains recordings of successful leak attacks. Even though the configuration did not change and the attack with this configuration caused relatively large delays in the router's response time (in comparison with other configuration values), our model could barely detect the attack with added cover traffic. We assume that the reason for that is that our model does not have a packet type feature; this was an intentional decision on our part, since the defense mechanism would not have knowledge of the type of packets used by the attacker (by focusing on a specific packet type response, an attacker can decode the data).

4.1. Attack Limitations

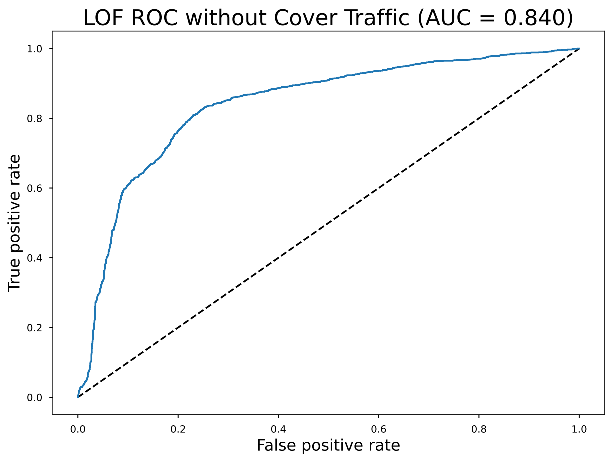
Both attacks discussed in this work require additional setup on the side of the attacker. As explained in Section 2.1, the attacks can only work if the attacker is able to craft custom traffic on the sender computer, send the data through the covert channel, decode the leaked information in the receiver computer and leak it online, without being noticed by any intrusion detection system. If the attacker has sufficient skills to install malware on the receiver's machine (which is connected to the internet), the attack's success is still dependent on the attacker being able to access the isolated system (the sender side). Moreover, in the Wi-Fi micro-jamming case, the attack requires an installation of a physical implant in the sender's network, which means the attacker must be able to access at least



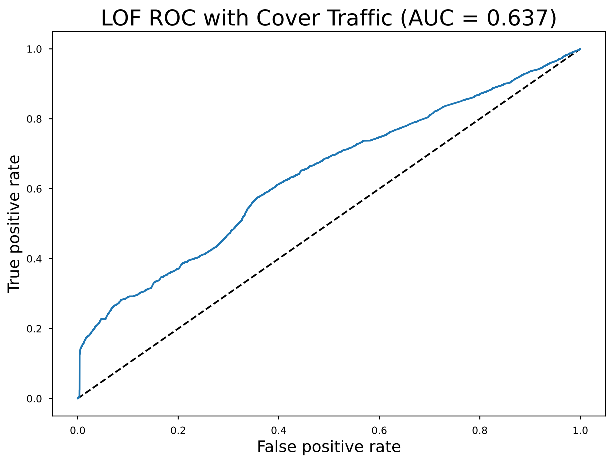
(a) ROC curve of the performance of the Deep autoencoder detecting the CRCC attack without cover traffic.



(b) ROC curve of the performance of the Deep autoencoder detecting the CRCC attack with cover traffic.



(c) ROC curve of the performance of the LOF model detecting the Wi-Fi micro-jamming attack without cover traffic.



(d) ROC curve of the performance of the LOF model detecting the Wi-Fi micro-jamming attack with cover traffic.

Figure 14: ROC curves of the performance of CRCC and Wi-Fi Micro-jamming detectors with and without cover traffic.

the vicinity of the isolated network to perform this attack. The reason for this is that the attack relies on a radio frequency (RF) signal to cause the delays, and the power of an RF signal decreases with distance. In the CRCC case, it is not limited by physical distance, but the method of delaying the response time (overloading the router with packets) is very noticeable.

All of this makes the attacks very difficult to perform. Only attackers the likes of nation-state actors, which could access the isolated system through other means, are probably able to perform the attacks. Nevertheless, if an attacker can overcome the mentioned difficulties, the isolated network's confidentiality can be compromised.

Also, it is important to note that using a single router as the host of both public and private networks is not compatible with network security's best practices. Still, the contribution of this work is showing that this kind of attack is theoretically possible and should be considered by security officers.

4.2. Defense Limitations

The solution that we propose to deal with the attacks described in this paper is based on ML/DL models that learn the network's "normal" behavior and perform novelty detection to detect the attacks. This defense system has some limitations. First, we assume that it is possible to integrate a man-in-the-middle device that can record all the network's traffic and process it to learn the network's behavior. Since it records all the network's traffic, including sensitive network data, this defense mechanism can become a prime target for hackers that wish to steal confidential information from the network. Moreover, to detect the attacks in real time, the defense mechanism should be able to record a large amount of network traffic and process it in real-time to detect the attack. This poses a big challenge in both software and hardware domains when developing the defense mechanism. Finally, network traffic tends to change over time for various reasons like clients' constant connection and disconnection, different network services usage and more. So, the defense mechanism should be constantly trained, and since it's simultaneously trying to

	LOF			Deep AE			OCSVM		
	AUC	F1-s (MF)	F1-s (F)	AUC	F1-s (MF)	F1-s (F)	AUC	F1-s (MF)	F1-s (F)
No CT	0.840	0.695	0.967	0.601	0.004	0.649	0.804	0.672	0.974
With CT	0.637	0.425	0.641	0.594	0.485	0.647	0.591	0.296	0.552

Table 7: Summarized results of Wi-Fi micro-jamming detection. Note that we abbreviated "mini-flow" to "MF", "flow" to "F", "cover traffic" to "CT", and "F1-score" to "F1-s".

	LOF	Deep AE	OCSVM
No CT	MF_TH = 0.6, k = 1000	MF_TH = 0.97, AE_TH = 0.45	MF_TH = 0.5, NU = 0.2
With CT	MF_TH = 0.3, k = 1000	MF_TH = 0.1, AE_TH = 0.9	MF_TH = 0.15, NU = 0.1

Table 8: Model parameters used in Wi-Fi micro-jamming detection. Note that we abbreviated "mini-flow" to "MF", "threshold" to "TH", and "cover traffic" to "CT".

detect the attacks, it'll probably suffer from many false-positive detections.

4.3. Related Work

This work presents two network timing covert channels and an ML/DL defense mechanism that can detect them.

In the timing channel implemented by Cabuk et al. [25], the covert information was divided into small, fixed-size timing windows. The sender's and receiver's times were synchronized. If the sender wanted to send the bit '1', it would send a packet in the current timing window. For '0', it would send nothing in the current timing window. Another work that presented a network timing covert channel that relies on packet arrival patterns was of Brodley and Spafford [26]. In their work, the receiver record packet arrival times sent by the sender and sorts them into two sets: S_0 and S_1. When the sender wants to send the bit '1', it replays a packet randomly chosen from the S_1 set. For the bit '0', it replays a random packet from S_0.

As for timing covert channel detection, Goher et al. [27] mentioned that timing covert channels are difficult to detect, because different applications on the network have different communication patterns that mask the irregular pattern of the covert channel. The authors presented a method for the detection of the timing covert channel which measures the interarrival time of packets and calculates attributes like the variance and difference between adjacent interarrival times. It is assumed that a covert channel that operates in a focused manner to leak information will have different statistical attributes than the seemingly random network communication of different applications. In our work, we take a similar approach, but instead of relying on statistical analysis for detection, we use machine and deep learning algorithms.

4.4. Future Work

In future work we plan to focus on finding the CRCC attack's optimal configuration and improving the defense mechanism by making it operate in real-time and finding methods to better identify the Wi-Fi micro-jamming attack with cover traffic.

Regarding the CRCC attack, in this work we assumed that the leaked data transmission rate was one bit per second. This may not be the max throughput possible for this attack. Implementing the attack in a more efficient programming language than Python like the C programming language, can improve the attack's performance and enable further exploration of the attack's optimal configuration. Also, both attacks may be improved by implementing an error correction/detection mechanism that will allow to increase the leaked data rate while keeping the data transmission reliable.

Regarding the defense mechanism, it is currently only capable of detecting attacks offline. To improve it, it should be implemented as a real-time detector which will be able to identify abnormal communication patterns and alert the network's administrator. We believe that the mechanism should be integrated in organizations' central routers, so it can analyze all a network's communication. The deployment should include an extensive training period in which the mechanism can learn the network's normal behavior; then it could record one-second segments of the network traffic and predict whether an anomaly is present in a segment.

Also, since the detection of the Wi-Fi micro-jamming attack with cover traffic is not sufficient for a real-time detector, and the fact that the network behavior constantly changes due to clients connecting and disconnecting, we propose exploring unsupervised/semi-supervised online learning models. These models are better suited for training on an ever-changing stream of data and might output better results than the models suggested in this work.

Finally, we also suggest that instead of determining the maliciousness of a flow based on the percentage of its malicious mini-flows, the classification of a malicious flow should be determined by a more dynamic "malicious score meter." In this case, each flow will receive a malicious score which will be initialized to zero. Each time a malicious mini-flow is detected, the score will increase by a small amount (for example, +0.1), and each time a benign mini-flow is detected, the score will decrease by a larger amount (for example, -0.5). Then, if a flow's malicious score exceeds a predefined threshold, it will be identified as malicious. This method is better fitted to dynamic detection and will lower the false positive detection of flows due to the small amount that is added to the score when identifying a malicious mini-flow.

4.5. Conclusion

In this work, we present two network timing covert channel attacks which can be used to break logical network isolation. By using specially crafted network traffic to create delays in the router's response time, we were able to leak sensitive information between two logically separated networks. We also proposed a defense mechanism based on ML and DL algorithms. We showed that the information leakage of both attacks can be detected by the defense mechanism in both a noisy and noise-free environment. Since the Wi-Fi micro-jamming attack uses low power RF to inflict delays in the router's responses rather than overloading the router with control plane packets, and the fact that we could barely detect the Wi-Fi micro-jamming attack with our model when cover traffic was present, we conclude that the Wi-Fi micro-jamming attack is stealthier than the CRCC attack. Network operators should evaluate their logically isolated networks and determine whether they match the attack model presented in this work. If so, they should consider applying the defense we presented here to detect and block cross-router attacks.

Acknowledgments

This work was supported by Israel Science Foundation grants 702/16 and 703/16. We thank Eyal Ronen for providing the inspiration and equipment for initiating our Wi-Fi micro-jamming research. We also thank Clémentine Maurice for providing the motivation for our cross-router covert channel research, and our WOOT shepherd, Paul Pearce, for helping us improve the initial workshop paper. Finally, we would like to thank Yagel Netanel for designing and implementing large parts of the CRCC attack system used to collect data for this research.

References

[1] Zhi-Kai Zhang, Michael Cheng Yi Cho, Chia-Wei Wang, Chia-Wei Hsu, Chong Kuan Chen, and Shiuhpyng Shieh. Iot security: Ongoing challenges and research opportunities. In *7th IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2014, Matsue, Japan, November 17-19, 2014*, pages 230–234. IEEE Computer Society, 2014.

[2] Johannes Sametinger, Jerzy W. Rozenblit, Roman L. Lysecky, and Peter Ott. Security challenges for medical devices. *Commun. ACM*, 58(4):74–82, 2015.

[3] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16, 2011.

[4] Medical device security, VA enterprise design patterns privacy and security, January 2017.

[5] Adar Ovadya, Rom Ogen, Yakov Mallah, Niv Gilboa, and Yossi Oren. Cross-router covert channels. In Alex Gantman and Clémentine Maurice, editors, *13th USENIX Workshop on Offensive Technologies, WOOT 2019, Santa Clara, CA, USA, August 12-13, 2019*. USENIX Association, 2019.

[6] Rom Ogen, Kfir Zvi, Omer Shwartz, and Yossi Oren. Sensorless, permissionless information exfiltration with wi-fi micro-jamming. In Christian Rossow and Yves Younan, editors, *12th USENIX Workshop on Offensive Technologies, WOOT 2018, Baltimore, MD, USA, August 13-14, 2018*. USENIX Association, 2018.

[7] Oren Shvartzman and Yagel Netanel. Characterization and detection of cross-router covert channels github repository. `https://github.com/orenshva/Characterization-and-Detection-of-Cross-Router-Covert-Channels`, 2022.

[8] Butler W. Lampson. A note on the confinement problem. *Commun. ACM*, 16(10):613–615, 1973.

[9] Sebastian Zander, Grenville J. Armitage, and Philip Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys and Tutorials*, 9(1-4):44–57, 2007.

[10] Clémentine Maurice, Christoph Neumann, Olivier Heen, and Aurélien Francillon. C5: Cross-cores cache covert channel. In Magnus Almgren, Vincenzo Gulisano, and Federico Maggi, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 46–64, Cham, 2015. Springer International Publishing.

[11] Jing Tian, Gang Xiong, Zhen Li, and Gaopeng Gou. A survey of key technologies for constructing network covert channel. *Secur. Commun. Networks*, 2020:8892896:1–8892896:20, 2020.

[12] Clémentine Maurice, Manuel Weber, Michael Schwarz, Lukas Giner, Daniel Gruss, Carlo Alberto Boano, Stefan Mangard, and Kay Römer. Hello from the other side: SSH over robust cache covert channels in the cloud. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society, 2017.

[13] Adel El-Atawy, Qi Duan, and Ehab Al-Shaer. A novel class of robust covert channels using out-of-order packets. *IEEE Trans. Dependable Secur. Comput.*, 14(2):116–129, 2017.

[14] Eric Wustrow, Colleen Swanson, and J. Alex Halderman. Tapdance: End-to-middle anticensorship without flow blocking. In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, pages 159–174. USENIX Association, 2014.

[15] Dubravko Miljković. Review of novelty detection methods. In *The 33rd International Convention MIPRO*, pages 593–598. IEEE, 2010.

[16] Cheng Yi, Alexander Afanasyev, Ilya Moiseenko, Lan Wang, Beichuan Zhang, and Lixia Zhang. A case for stateful forwarding plane. *Comput. Commun.*, 36(7):779–791, 2013.

[17] Andrés Felipe Murillo-Piedrahita, Sandra Julieta Rueda, Diogo M. F. Mattos, and Otto Carlos M. B. Duarte. Flowfence: a denial of service defense system for software defined networking. In *2015 Global Information Infrastructure and Networking Symposium, GIIS 2015, Guadalajara, Mexico, October 28-30, 2015*, pages 1–6. IEEE, 2015.

[18] James F. Kurose and Keith W. Ross. *Computer networking - a top-down approach featuring the internet.* Addison-Wesley-Longman, 2001.

[19] IEEE 802.11 Working Group. Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, Dec 2016.

[20] Microchip Technology Inc. Atmega256rfr2 xplained pro evaluation kit. `http://www.microchip.com/DevelopmentTools/ProductDetails.aspx?PartNO=atmega256rfr2-xpro`.

[21] Jonathan J. Davis and Andrew J. Clark. Data preprocessing for anomaly based network intrusion detection: A review. *Comput. Secur.*, 30(6-7):353–375, 2011.

[22] Tsatsral Amarbayasgalan, Bilguun Jargalsaikhan, and Keun Ho Ryu. Unsupervised novelty detection using deep autoencoders with density based clustering. *Applied Sciences*, 8(9):1468, 2018.

[23] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and

Jörg Sander. LOF: identifying density-based local outliers. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 93–104. ACM, 2000.

[24] Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognit.*, 58:121–134, 2016.

[25] Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP covert timing channels: design and detection. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick D. McDaniel, editors, *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004*, pages 178–187. ACM, 2004.

[26] CE Brodley, EH Spafford, and S Cabuk. Network covert channels: Design, analysis, detection, and elimination. *Dissertations & Theses, Purdue University*, 2006.

[27] S Zerafshan Goher, Barkha Javed, and Nazar Abbas Saqib. Covert channel detection: A survey based analysis. In *High Capacity Optical Networks and Emerging/Enabling Technologies*, pages 057–065. IEEE, 2012.