SoK: A Survey Of Indirect Network-level Covert Channels

Tobias Schmidbauer∗
tobias.schmidbauer@gmx-topmail.de
FernUniversität in Hagen
Hagen, Germany

Steffen Wendzel∗
wendzel@hs-worms.de
Worms University of Applied Sciences
Worms, Germany
FernUniversität in Hagen
Hagen, Germany

ABSTRACT
Within the last few years, indirect network-level covert channels have experienced a renaissance with new ideas and evolving concepts. Logical network separation may now be crossed and the sending and receiving activities can be performed with temporal distance between sending and receiving operations. Despite these new developments, all indirect network covert channels share certain basic principles that allow a categorization. So far, the concepts of indirect network-level covert channels have never been systematized. In this paper, we introduce a taxonomy containing indirect covert channel patterns that allow a differentiated analysis of all known indirect network-level covert channels. We introduce additional definitions to unify the understanding of the domain and further identify crucial features of indirect covert channels to make them comparable and describable. We further discuss application scenarios as well as potential and already evaluated countermeasures against indirect covert channels. Further, we discuss observable trends and anticipated future developments in the research area of indirect network-level covert channels.

CCS CONCEPTS
• Security and privacy → Network security; Distributed systems security; Information flow control; Pseudonymity, anonymity and untraceability; • Social and professional topics → Computer crime.

KEYWORDS
Steganography, Indirect Covert Channels, Information Hiding, Dead Drop, Proxy, Redirector, Hiding Patterns, Network Protocols

1 INTRODUCTION
Network-level Covert Channels (CCs) are used for steganographic data exchange between systems and are differentiated into direct and indirect CCs. An indirect CC is established when an intermediate node is exploited to store, represent or unintentionally redirect Covert Information (CI) between communicating parties.

Since 2019, indirect network-level CCs have experienced a renaissance, long after their first appearance in the 1990s. New ideas have emerged, for example the trend of storing CI in network protocol caches. Several techniques allow a temporal distance between the activities of a Covert Sender (CS) and Covert Receiver (CR), creating an asynchronous scenario [38]. The exploitation of shared physical devices for the exchange of CI, even if the CS’s and CR’s network are logically isolated [34], is another recent advancement. In addition to these new ideas, earlier implementations have been applied to additional protocols and in new environments. This fast development generates an urge for an appropriate systematization of available knowledge in the domain.

In this paper, we provide results of a literature analysis on indirect network-level CCs, covering publications from 1997 till 2021. We present a taxonomy that contains three so-called indirect CC patterns, which are used to group the discovered methods. We also introduce definitions and naming conventions for common terms and identify the crucial features of the indirect CC patterns. We analyze the utilization of so-called information hiding patterns [44] by indirect CCs to determine commonalities in their embedding and extraction methodology. Furthermore, we analyze the application scenarios for existing indirect network-level CCs and present potential countermeasures for every conceivable point between the CS and CR, at which a countermeasure can be located.

The remainder of this paper is structured as follows. In Sect. 2, we introduce the fundamentals of CCs, survey the known indirect network-level CCs and cover related work. Sect. 3 presents three indirect CC patterns for categorization as well as indirect CC features and naming conventions. In Sect. 4, we categorize existing indirect CCs based on our taxonomy, including their attributes, and discuss our observations. Furthermore, Sect. 5 summarizes already published application scenarios for indirect network-level CCs as well as potential and already evaluated countermeasures. We give an outlook on anticipated future developments in Sect. 6. Finally, Sect. 7 concludes.

2 FUNDAMENTALS
In the following subsections, we explain the fundamentals of CCs and survey published indirect network-level CC implementations. Finally, we cover related work.
2.1 Covert Channels

A CC is a communication channel between a CS and a CR to exchange CI in a stealthy or policy-breaking manner and was first introduced in [22]. CCs are based on exploitable carriers, the so-called overt channels or cover objects. Thus, a CC can be considered as a parasitic channel that nests into a legitimate communication. While there are also CCs defined for local systems (e.g., between local processes of an operating environment) all further mentions of CCs refer to network-level CCs. For network-level CCs, CS and CR exploit a communication protocol in such a way as it was never considered its purpose originally. The first network covert channels arose in the 1980s and exploited characteristics of local area networks [15, 47], several surveys (e.g., [31, 46, 50]) analyze and categorize network-level CCs. If a CS addresses more than one CR at the same time, this CC is also called a multicast or broadcast CC.

**Figure 1: Covert Channel Types**

Literature differentiates CCs into direct and indirect implementations [50]. For a direct CC, the CS and CR exchange information directly using a (modified) protocol, as shown in Fig. 1(a). There is no need for intermediary nodes to establish (or represent CI of) the CC, though the intermediaries are necessary to establish the overt channel. In contrast to this, an intermediate node has an essential role for an indirect CC, i.e., the intermediate node is involved stronger than a plain router. CI are not exchanged directly between the CS and the CR but are stored, represented or unintentionally redirected by the intermediate node (Fig. 1(b)), e.g., with the purpose of communicating over a barrier. Further, CCs were historically divided into two categories – timing and storage channels [6], depending on how the CI is encoded. This categorization has been revised with the introduction of a taxonomy of information hiding patterns for CCs [44]. Such hiding patterns are recurring solutions for similar problems and allow precise grouping based on a template. A distinction is made between embedding and representation patterns. Embedding-pattern describe how the CI is embedded into a carrier, while representation patterns describe how this information is rendered recognizable by a CR. We provide additional background information on hiding patterns in appx. A.

2.2 Survey of Indirect Network-level CCs

We conducted a careful literature search using several publication databases, such as Google Scholar. To the best of our knowledge, the implementations listed below are considered to be all currently known indirect network-level CCs. The channels are grouped according to their alphabetically sorted protocol abbreviation. In the remainder, they will be referred by this abbreviation, and enumerated w.r.t. their historical appearance from 1 to \(n\). Thus, they will be shortened as Protocol\(n\). If no protocol has been specified in the source, we will refer to it as NET in the sense of protocol-independence.

**802.11.** The IEEE 802.11 standard defines how information are exchanged in WiFi networks via the data link layer. In [54], an indirect CC has been described that exploits this standard (802.11.1). The CS forces \(n\) clients to reconnect within a specific time-frame, transmitting \(n\) bits per frame to a CR that observes the reconnect attempts. The clients are either directly controlled by the CS or indirectly forced to reconnect by manipulating 802.11 frames.

**ARP.** The ARP protocol maps the physical hardware address of a network device to its logical IP address. To reduce necessary discovery operations, the protocol stores information about known communication partners in a cache, the so-called ARP table. There exist four indirect CCs for ARP, two with and two without the exploitation of caches. In [38], the authors described how the ARP cache can be exploited for a steganographic data storage by modified ARP-requests. The extraction of the CI is made possible via SNMP, which is intended to query system status information. Linux-based systems can store up to 127 entries (ARP.1) and Windows-based systems up to 256 entries (ARP.2). There exist two more indirect CCs based upon ARP that have been described in [34]. ARP.3 exploits the fact that some routers forward ARP-requests if the target IP is set to a broadcast-address. Thus, the CS can address the CRs indirectly via an intermediate and transfer 8 bit of CI per ARP-Request. Further, ARP-requests affect the CPU load (ARP.4). A rate of 100 ARP-requests influences an intermediate node in such a way that the response time of other packets is delayed. A CR can measure this and extract CI from the intermediary.

**CoAP.** The Constrained Application Protocol (CoAP) is a lightweight communication protocol that has been designed for Internet of Things (IoT) and Industrial IoT devices and uses URLs and methods similar to HTTP to exchange information. In [18], the authors describe an indirect CC based on client resets (CoAP.1). Here, the CS forces the intermediate node to send RST-packets to several third-party nodes, where each node’s reset is linked to a different hidden symbol. The CR monitors the resets in the network to decode the CI. The CC can transfer \(\log_2 n\) bit per reconnect (\(n\) being the number of nodes which can be exploited for a reset).

**DHCP.** By using the DHCP protocol, a server can assign intended network configurations specifically and generally to clients. In [34], two indirect CCs based upon DHCP have been described. DHCP.1 exploits the fact that some routers with an activated DHCP-server reply to DHCP-Requests with faked broadcast- and multicast source IP addresses. The DHCP-ACK is then sent via broad- or multicast to the given source address and can cross network borders, by exploiting routers as intermediate nodes. Besides this exploitation, DHCP influences the CPU load and the file system activity due to logging (DHCP.2) [34]. This can be exploited by a CS to create delayed acknowledgments to transfer 1 bit of CI through an intermediate node per CR request.

**DNS.** The Domain Name System (DNS) consists of several pool-servers that resolve IP addresses to domain names (and vice versa) and refer available mail servers of a domain, among other features. A DNS-based indirect CC has been presented in [2] (DNS.1). The channel is based on the protocol’s functionality of negative caching. The CC participants first agree on an ordered list of non-existent domain names. If a non-existent domain name was requested from a
DNS server, an entry is added to the so-called NCACHE. To transmit CI, the CS sends a recursive request from one of these non-existent domains to a DNS server. One or more CRs can then extract the hidden message from this DNS server through a non-recursive query since the DNS server reports back an entry in the NCACHE. It must be noted that each DNS cache poisoning attack can also be applied to transfer CI. DNS poisoning was described in several papers [26, 39, 52]. Even though not intended for CCs, cache poisoning may be applied for indirect CCs, cf. [2], and can also target client-side DNS caches [1]. We waive these potential channels as they would apply a highly similar methodology as DNS.1.

HTTP. The HTTP is a stateless network protocol that allows the transport of hypertext documents and is mostly used to deliver web-content from a web server to a client application (mostly web browsers). In [4], several HTTP-based indirect CCs have been described that allow the exchange of CI between two HTTP servers through a client system. For HTTP.1, a server forwards a client to transport CI via a redirect to another server. This has also been described in [11] with little variations. Cookies can be used by a CS to store CI on a client system for a later extraction by a CR (HTTP.2). For HTTP.3, referer headers are used by a CS to redirect CI via a HTTP-client to another system. Further it is described, how active content like ActiveX can be used to force the client to transfer CI to another server (HTTP.4). In [10], the authors describe how an indirect CC can be established via BitTorrent trackers. The CS encodes CI in a modified, so-called infohash that is stored at the centralized tracker. Read and write operations can be performed via HTTP-requests (HTTP.5). Moreover, a HTTP-based Cross Site Request Forgery (CSRF)-Request can be exploited to increase the CPU load of an intermediary node to transfer CI. The CC has been described in [34] and increases the CPU load of a web server by a CS, repeatedly loading its websites in iframes. The increased response time of the web server can be observed by the CR (HTTP.6).

ICMP. The ICMP protocol is utilized to transport information and error messages for the Internet Protocol. In [12, 50, 51], an indirect CC is presented that utilizes ICMP echo requests. The CS encodes the CI as payload and also modifies the source IP field with the IP address of the CR. The CS sends the request to an intermediary node that answers with an ICMP echo reply and redirects the payload to the CR (ICMP.1). Ovadia et al. [34] describe how repeated ICMP requests influence the response time for other requests due to elongated acknowledgement times (ICMP.2). This delayed acknowledgement can be detected by a CR, transferring 1 bit of CI via an intermediate node. Since CCs for ICMPv6 can be realized in a similar way as in case of ICMPv4, we do not discriminate between the two protocols in the remainder.

IGMP. The IGMP protocol is utilized to organize IP multicast groups. In [34], the authors describe how IGMP can be exploited to establish a CC by IGMP membership queries via multicast IPs. The CS achieves the CI transfer by joining and leaving an IGMP group, forcing the intermediary to query if there are still hosts in this group. The CI is transferred through the Group-IP field that is controlled by the CS (IGMP.1).

MQTT. The Message Queuing Telemetry Transport protocol is used for the communication between IoT devices. It specializes in conveying status information of so-called topics through an intermediary node, carrying so called properties. This system makes topic-related information available to other systems that can send related queries. Six indirect CCs based upon MQTT have been described in [42]. Since the implementation of the MQTT protocol was designed to store data at an intermediary, there are several options to implement a steganographic data storage. MQTT.1 exchanges information between a CS and CRs by agreeing on all sub-topics of a first level topic. The name of the sub-topics are carrying the CI, transmitting n Byte. MQTT.2 is based on topics that are arranged in a specific order. Each topic that is updated within a predefined time-frame by the CS represents a binary 1 and is observable by the CRs. For MQTT.3, the CS first needs to provide its unique identifier to the CR. Both, the CS and CR, are then able to create either a persistent session with the intermediary (representing a 1) or a non-persistent session, representing a 0. The previous session type can be detected by the response of the intermediary. MQTT.4 exploits retained messages stored at the intermediary. The CS can transmit CI to CRs by sending a topic update, and setting the RETAIN flag to 1, forcing the intermediary to send a retain message to CR subscribers. Retain messages are also exploited by MQTT.5 to transmit n bits. The CS and CRs must agree in advance on n ordered topics, where a present retain message represents a 1, while an absent retain message represents a 0, transferring n bits at once. MQTT.6 is realized by the CS manipulating the number of clients that are connected to the intermediary in a predefined time-slot. If more than the average number of clients are connected the CR, it represents a binary 1, otherwise a 0. Five additional indirect CCs have been described in [32]. MQTT.7 utilizes the order of n properties of a MQTT Packet and can transmit n bit of CI. MQTT.8 is created by either duplicating properties in one request or not (representing a boolean), sending n bit of CI, depending on the number of transmitted properties. For MQTT.9, two MQTT clients (CS and CR) agree on two topics, whereby each client offers one topic and subscribes to the other topic. CS and CR can establish a bi-directional indirect CI, encoding a binary 1 or 0, depending on in which combination the topics are updated. MQTT.10 can transport a message of n bit from a CS to CRs by agreeing on 2^n topics, representing a binary number. For MQTT.11, the CS (a client) forces the MQTT broker to reconnect third party clients to transmit CI to an observer (the CR). Each third party system represents a symbol of a predefined alphabet.

NTP. The NTP protocol provides the service of synchronizing the configured time between systems, which is achieved by so-called pool servers that can be accessed publicly. For NTP, two methods have been presented in [37] to transmit CI indirectly. NTP.1 describes that under certain circumstances, an intermediate node can be forced to store CI by establishing a broadcast-association. The extraction is accomplished by NTP command queries. The second indirect CC describes that CI can be stored in the Most Recently
used (MRU)-cache. The MRU-cache stores the last 120 communication partners with whom a time exchange took place and can be exploited to store CI. Since an NTP client rarely communicates with new NTP pool servers, these entries age slowly and remain in the MRU-cache with 4 bytes per entry (NTP.2). This procedure makes it also possible to store CI in NTP pool servers but the entries cannot be extracted within a few seconds since new systems keep making inquiries and thus rapid aging takes place (NTP.3).

**SSH.** The SSH protocol provides a protected shell service that allows to connect clients to remote servers. An indirect SSH CC is described in [34] and exploits the computational intensive ssh-key exchange between the CS and the intermediate node (SSH.1). This method can also be applied by the CR to extract information, by detecting increased reply runtimes due to increased CPU load.

**TCP.** The Transmission Control Protocol (TCP) defines how information are exchanged between two systems and is a reliable, ordered, and error-checked transport protocol. For TCP, an indirect CC was described in [35] (TCP.1). The CS encodes the CI into the ISN-Field and further modifies the source IP header field so that the IP of the CR is stored. The CS sends a SYN packet to the intermediate node, which replies with a SYN/ACK packet and redirects the CI to the CR that was designated in the source IP. Further, in [19, 33] an indirect CC for TCP based on packet rate and times tamp modulation has been described. The CS manipulates the CPU temperature and load of the intermediary by sending requests to the intermediate node in a predefined time window. The CR sends a TCP packet to the intermediate node and can detect delays in acknowledgments (TCP.2). However, there exists a third indirect CC TCP.3 that has been described in [12]. The CS influences the intermediaries IPID counter by choosing an adequate window size, forcing the intermediate node to send back more packets than necessary. The CR then can extract the CI represented by the counter ID by sending a TCP packet.

### 2.3 Related Work

CCs have been extensively investigated in their different variants over the last decades. An early overview of several network CCs was provided by Handley and Sandford [17] and a less comprehensive survey was conducted by Llamas et al. in [25]. The first major survey that also provided a terminological advancement as well as a comprehensive analysis of fundamentals and countermeasures was published by Zander et al. [50]. Zhiyong and Yong performed a categorization of covert channels on the basis of entropy (three classes of covert channels were defined for variety entropy, constant entropy, and fixed entropy) [53]. Mazurczyk reviewed covert channels that exploit VoIP communications in [28], where he also analyzed available countermeasures. In 2014, Mileva et al. surveyed state-of-the-art TCP/IP related covert channels [31]. In 2015, Wendzel et al. categorized all network CCs published between 1987-2013 into so-called hiding patterns [46], which were extended in 2016 by Mazurczyk et al. [29], adjusted and extended in the years 2016-2021, and finally revised in summer 2021 by Wendzel et al. [44].

Indirect CCs have been described in several publications and can for example be found in [50]. The principle of a steganographic storage CI within building automation was described in [45] – it is therefore a cyber-physical dead drop, instead of one exploiting network protocol caches or states. Side channels are CCs that leak (sensitive) information without a sending-intention — a survey can for instance be found in [23]. In this paper, we essentially focus on covert channels with a sending intention. In [5], Baumann et al. describe how malicious scripts can be stored in browser caches. The injection is performed through a TCP injection and a CC controls which scripts are executed by the browser of the victim. In [24], the authors describe how to remotely exploit the Rowhammer attack described in [48]. A robust cache CC for cloud computing based on CPU caches has been described in [27]. In [49], the authors introduced a taxonomy for covert channels that base on attacks on speculative executions of dynamically scheduled execution of instructions by processors.

Several publications also deal with countermeasures for (indirect) network CCs. Already in 1983, Simmons described how a so-called warden can detect a CC between Alice and Bob [41]. Therein three wardens are described, a passive warden that only observes traffic, an active warden that removes steganographic content, and a malicious warden that modifies the content of a covert message by intent to influence CS and CR. There exist various countermeasures against CCs. A survey on passive wardens that detect covert channels was performed by [16]. VoIP CC-specific countermeasures were surveyed in [28]. Examples as well as current challenges and trends for covert channel countermeasures were analyzed by Mazurczyk et al. [29] and Caviglione [8]. Caviglione’s results indicate that several methods are highly specialized, while there is a demand for more general approaches.

### 3 TAXONOMY AND NAMING CONVENTIONS

In this section, we first introduce our taxonomy for indirect CCs and define necessary terms. Afterwards, we identify crucial features for indirect CCs and introduce naming conventions to describe these channels precisely and enable their comparison.

#### 3.1 Taxonomy

All CCs described in Sect. 2.2 are indirect network-level CCs that necessarily need an intermediate node for their implementation. This intermediary is the crucial commonality and can be described by the following definition:

**Definition 1 (Intermediate Node).** An intermediate node for an indirect network covert channel is a crucial system that reflects, stores or represents CI. It is not necessarily aware of the fact that it is part of a covert channel.

Essentially, the intermediate node is a communication participant that is forced by the endpoints of a CC to take on an unforeseen role in the steganographic information exchange. However, the intermediate node may have been specifically and possibly unintentionally modified in advance so that it participates in the covert data transfer.

Nevertheless, there exist two variances of indirect CCs that allow a finer distinction. Some intermediary nodes are forced to unintentionally redirect CI to the CR, and some are forced to represent information so that the CR can later extract it. These dissimilarities allow the differentiation of indirect CC behavior into indirect CC...
patterns as presented in Fig. 2. The first indirect CC pattern introduced is the redirector pattern, in which an intermediate node is forced by a CS to unintentionally redirect CI to a CR. Further, a broker is forced by the CS to represent or store CI and has two variants: the proxy pattern, in which the CS manipulates the broker in such a way it represents the CI and the CR can extract the hidden information, and the dead drop pattern, in which the CS forces the intermediate node to store CI in a network protocol’s cache so that a CR can extract it. These three indirect CC patterns are described and defined in the following subsections.

3.1.1 Redirector Pattern. The redirector pattern is an indirect CC pattern that processes the steps as described in Fig. 3. In step a), the CS sends CI to a redirector. This intermediate node is forced to redirect the packet to a CR. Thus, a redirector is a specialized intermediate node that can be defined as follows:

**Definition 2 (Redirector).** A redirector for an indirect network covert channel is an intermediate node that is forced by a (set of) manipulated packets to unintentionally redirect CI to another host; however, the CI are neither intentionally stored nor manipulated by the redirector.

The redirector is an active part of the CC but does neither store CI nor manipulate packets. The communication is never initiated by the redirector but forced by the CS to unintentionally redirect CI to the CR. The redirector, for example, can redirect unicast acknowledgements to the CR (like in TCP) or reflect broad- and multicast acknowledgements (like in IGMP).

**Figure 3: Redirector Pattern**

The implementation of the redirector pattern has the advantages that CS and CR are separated by the intermediary. Though, in some cases there is the disadvantage of a protocol violation, and for some implementations, the CS has to address the CR in each flow. These disadvantages can negate the described advantage.

3.1.2 Broker. In contrast to the previous indirect CC pattern, brokers do not redirect packets but keep the state of a CI in a cache or are influenced in such way that the information can be extracted by the CR. Thus, a broker can be defined as follows:

**Definition 3 (Broker).** A broker for an indirect network covert channel always has a passive role in a CC and never redirects CI from the CS to the CR.

There are two types of brokers — the CI-representing proxy and the CI-storing dead drop.

**Proxy.** The proxy is a broker that is influenced by the CS but it never redirects messages directly. The principle is presented in Fig. 4 and can be split into two segments. In the first segment (step a), further on called manipulation), the CS manipulates the proxy, and receives acknowledgements if the utilized protocol requires this. The CS has to repeat the manipulation until the proxy answers a request of the CR in a specific way (step b), further on extraction).

**Figure 4: Proxy Pattern**

A proxy can be described by the following definition:

**Definition 4 (Proxy).** A proxy for an indirect network covert channel is a broker that is influenced by a CS is in such a way that it replies to the requests from a CR at will of the CS.

The proxy has several advantages:
1. CS and CR do not have to address each other to exchange messages;
2. the CI disappears after extraction and complicates forensics. However, the proxy pattern has disadvantages, too:
1. CS and CR have to be active immediately one after the other, which might be correlatable by an observer;
2. the CC is difficult to coordinate due to timing issues.

**Dead Drop.** The dead drop is manipulated by a CS to store CI in a cache, i.e. in contrast to the proxy pattern, the CI remain at the dead drop. The CR then requests the dead drop that responds with these stored CI.

The general procedure of this indirect CC pattern is presented in Fig. 5(a). In step a), the CS manipulates the broker, and receives an acknowledgment if required by the (manipulated) protocol. The dead drop is forced to store the CI in a network protocol’s cache (step b). Finally, the CR requests the extraction and receives the CI from the dead drop. Further on, step a) will be referred to as manipulation, step b) as storage and step c) as extraction.

The time sequence of activities is sketched in Fig. 5(b). The CS is only active at the beginning of the information flow and communicates with the dead drop. The dead drop stores the CI, which is subject to an aging process and disappears after a certain time. We refer to this disappearance process as fading. The CR becomes active before the CI faded and extracts the information.

Thus, a dead drop can be defined as follows:
Response

Covert Sender

Receiver

Sender

Covert

Covert

Dead

Amount of Stored CI

| |

Therefore, different scenarios were found for the surveyed CCs:

This is visualized in Fig. 6. The CS wants to store CI in a network protocol cache to enable the CI’s extraction by a CR.

As already mentioned, a dead drop’s CI fades after some time. There are two possibilities for fading to occur. Either the cache rejects entries after a certain time or the entries are deleted by an event (e.g., the cache is a ring buffer that is already full but has to store a new entry). If the fading depends on time, we will further call it timed because the timing of the CI-loss can be calculated. If the CI-loss depends on an event, it will be called logic. There is a possibility that the fading depends on both, timing and events. This will further be called hybrid fading. If the dead drop allows logic or hybrid fading, the CS and CRs can influence the time of the CI disappearance. Let C represent the covert information (CI). C consists of the substrings \(c_1, \ldots, c_n\), i.e. C = \(c_1 || c_2 || \ldots || c_n\), with || being the string concatenation. For both timed and logic fading, either individual substrings, or the combined substring can fade.

As already mentioned, timed fading can be exactly calculated if the storage time is known and depends on the injection speed. This is visualized in Fig. 6. The CS wants to store C on a dead drop and decides for an injection speed. \(c_1\) is injected at the time \(t_0\) and the injection of all succeeding substrings from \(c_1\) to \(c_n\) has to be completed under constraint of the maximum storage duration to reassemble the whole message at the CR, while the fading starts after \(\Delta_fading\) which is the point of \(c_1\) disappearing. \(\Delta_reset\) corresponds to the time that passes after \(\Delta_fading\) and until \(c_n\) has faded. Therefore, different scenarios were found for the surveyed CCs:

1. **Timed Fading of Individual Substrings**: If \(c_1\) was written at the time \(t_0\), it must be read by CR at the time \(t_0 + \Delta_fading\) while \(t_0\) must be read at \(t_0 + \Delta_fading\).
2. **Timed Fading of Combined Substrings**: In some cases, all information of a cache is lost simultaneously. In such a case, \(c_1, \ldots, c_n\) all fade at the time \(t_1 + \Delta_fading\) or after some time.

**Figure 5: Dead Drops and Fading**

**Definition 5 (Dead Drop).** A dead drop for an indirect network covert channel is a broker that is forced by the CS to steganographically store CI in a network protocol cache to enable the CI’s extraction by a CR.

**Fading of CIs in Dead Drops.** As mentioned, a dead drop’s CI fades after some time. There are two possibilities for fading to occur. Either the cache rejects entries after a certain time or the entries are deleted by an event (e.g., the cache is a ring buffer that is already full but has to store a new entry). If the fading depends on time, we will further call it timed because the timing of the CI-loss can be calculated. If the CI-loss depends on an event, it will be called logic. There is a possibility that the fading depends on both, timing and events. This will further be called hybrid fading. If the dead drop allows logic or hybrid fading, the CS and CRs can influence the time of the CI disappearance. Let C represent the covert information (CI). C consists of the substrings \(c_1, \ldots, c_n\), i.e. C = \(c_1 || c_2 || \ldots || c_n\), with || being the string concatenation. For both timed and logic fading, either individual substrings, or the combined substring can fade.

As already mentioned, timed fading can be exactly calculated if the storage time is known and depends on the injection speed. This is visualized in Fig. 6. The CS wants to store C on a dead drop and decides for an injection speed. \(c_1\) is injected at the time \(t_0\) and the injection of all succeeding substrings from \(c_1\) to \(c_n\) has to be completed under constraint of the maximum storage duration to reassemble the whole message at the CR, while the fading starts after \(\Delta_fading\) which is the point of \(c_1\) disappearing. \(\Delta_reset\) corresponds to the time that passes after \(\Delta_fading\) and until \(c_n\) has faded. Therefore, different scenarios were found for the surveyed CCs:

1. **Timed Fading of Individual Substrings**: If \(c_1\) was written at the time \(t_0\), it must be read by CR at the time \(t_0 + \Delta_fading\) while \(t_0\) must be read at \(t_0 + \Delta_fading\).
2. **Timed Fading of Combined Substrings**: In some cases, all information of a cache is lost simultaneously. In such a case, \(c_1, \ldots, c_n\) all fade at the time \(t_1 + \Delta_fading\) or after some time.

**Figure 6: Timed Fading of Individual Substrings**

logical condition was triggered (e.g., ARP cache limit was reached).

In general, if the CI are injected by even intervals, the increase is linear and also \(\Delta_fading\) will be linear. If the CS decides to inject CI exponentially with a decreasing interval, the fading will be increasing with time. If the CS injects CI logarithmically with a decreasing interval, the CI will fade with an increasing interval. For all cases, the fading is completed at \(\Delta_{reset}\) when all entries vanished. CS might apply classical real-time scheduling constraints to ensure that the secret message CI will be fully recoverable by the CR.

The dead drop pattern offers several advantages:

1. No direct communication between CS and CR is necessary after they agreed on transmission parameters;
2. a temporal distance between the activity of CS and CR is feasible;
3. several dead drops (for scaling and obfuscation) could be combined;
4. fading of the CIs complicates forensics and can be considered an advantage.

However, these advantages also come with the cost of a few disadvantages:

1. external influences can destroy CI;
2. there is a limited lifetime of the CI within the dead drop.

### 3.2 Features

Besides the indirect CC patterns, there are several features that affect the properties of indirect CCs. Many are relevant for all CCs but some only influence certain indirect CC patterns. These features are the fundamentals for a detailed description of indirect CCs and are incorporated into the naming conventions introduced in Sect. 3.3. A feature is represented by capitalized letters, while the characteristic of a feature is represented by a lower-case letter. We provide a brief overview of all features of each pattern in appx. B.

#### 3.2.1 Features for all indirect Covert Channel

The following features apply to all indirect CCs:
Indirect Covert Channel Pattern (ICCP). The indirect CC patterns introduced in Sect. 3 describe the fundamental difference of all indirect CCs. The indirect CC can either be a redirector (R), or a broker variant, e.g. a proxy (B.P.) or a dead drop (B.D.).

Protocol Violation (V). In some cases, CS and CR are forced to violate a protocol’s standard-conform behavior. This can be observed by wardens and has an influence on the detectability of the CC. The characteristic can be:

• conclusive (c);
• violated (v).

Authentication (A). Depending on the utilized protocol to submit CI, it may be necessary for the CS and CR to authenticate themselves to the intermediate node. Some protocols may require optional authentication. Thus, the communication between CS, CR and intermediate node can be:

• authenticated (a);
• unauthenticated (u);
• optional (o).

Remote Reachability (R). The utilized protocol can either be limited to local communication (for example within a collision or broadcast domain) or can allow remote communications across network boundaries (through a routable protocol). Thus, potential characteristics are:

• local (l);
• remote (r).

Bitrate (B). The bitrate describes the number of bits that can be transferred within one transmission attempt during the redirection, manipulation or extraction. The quantities are specified in n bits (n bit) or n bytes (n B).

Protocol (P). For indirect CCs, the chosen network protocol(s) provide fundamental conditions for most of the properties, especially authentication, reachability and the potential bitrate.

Information Hiding Pattern (H). The information hiding patterns describe how CI are embedded and represented using the particular network protocol. Further, they allow a quick overview of potential countermeasures and detectors as these can sometimes be applied to entire pattern groups. The naming convention of an information hiding pattern are described in [44] and consequently applied in this paper; too. We both refer to the embedding and the representation pattern for each CC. All hiding patterns mentioned in this paper are described in appx. A.

3.2.2 Proxy Features. For the proxy pattern, it is crucial to describe how the proxy is influenced by the CS. For this reason, the following feature is introduced:

Load Intensity (N). The CS has to influence the proxy, so it represents the CI. Therefore, a repetition of time-consuming actions (e.g., requests to the proxy) can be necessary so that the CR can recognize the CI. Obviously, such events ease detectability of the CC. The Load Intensity can thus be categorized as follows:

• high (h), i.e. many repetitions of actions have to be performed by CS (like required for influencing the CPU load in TCP.2);
• normal (n, i.e. no unusual repetition of actions by CS is required).

3.2.3 Dead Drop Features. As dead drops utilize caches to store CI, the storage itself and the behavior of the CI have to be described. The following features are additionally necessary for indirect dead drop CCs:

CI Capacity (C). The (potential) CI capacity of a proxy or dead drop corresponds to the amount of CI that can be represented or stored. The quantities should be specified in n bits (n bit) or n bytes (n B).

CI Lifetime (L). The lifetime of CI has an influence on the attractiveness of a dead drop for attackers. A short lifespan makes it necessary to precisely coordinate the manipulation and extraction. This may negate the advantage of the temporal distance of CS and CR and may lead to suspicious network activities. Further, if the lifetime is too long, it can lead to entries being discovered by an observer in the dead drop. We therefore suggest dividing the lifetime into three levels:

• volatile (v): < 1 min storage period;
• temporary (t): < 1 h storage period;
• persistent (p): ≥ 1 h storage period.

Volatile and persistent dead drops are suitable for the steganographic storage of information but are to be regarded as second choice for the reasons already mentioned. A temporary network protocol cache, however, offers a balanced lifetime until the CI fade. It is possible that the storage process allows all three states, e.g., if the CS can influence whether the CI fade or not. This is exemplified by the overflow of the brokers ARP-cache, as described in [38].

Fading (F). The fading is the second feature to complete the description for indirect dead drop CCs. This feature is important because it determines whether the CI’s fading can be controlled by the CS and whether CS and has influence on detection and throughput. Like explained in Sect. 3.1.2, fading can be

• timed (t);
• logic (l), or
• hybrid (h).

3.3 Naming Conventions

After we have described all the features, it is necessary to structure them to ensure clarity. Naming an indirect CC has always to begin with the implemented indirect CC pattern, followed by the optional storage features and the features of all utilized protocols. The naming construction is modular: if necessary, an additional second protocol, cache or the load intensity can be added.

Protocols can be described as P.\{V, A, R, C, H\}, a dead drop cache can be described as {F, L, C, H} and the proxy behavior as \{N, H\}. Thus, the long form naming conventions for all indirect CCs are:

ICCP\{F, L, C, H\} \{N, H\} \{P\_1, P\_2\}
\{F, L, C, H\}, \{N, H\} and P\_2 are optional and only added if the indirect CC implements these features. This facilitates the readability, as we reduce the number of descriptive features to a necessary minimum. Anyway, the naming conventions lead to long descriptions, especially for dead drop CCs. Hence, the naming convention for the redirector and proxy patterns are:
• Redirector: R. [P{V, A, R, B, H}]
• Proxy: B.P. [N, H], P1 {V, A, R, B, H}, P2 {V, A, R, B, H}

We propose to split for these CCs the description for each segment. The naming conventions allow to describe a dead drop CC in long terms as B.D. [F, L, C, H], P1 {V, A, R, B, H}, P2 {V, A, R, B, H} or split for each segment:

- Dead Drop: B.D [F, L, C, H]
- Manipulation: P{V, A, R, B, H}
- Extraction: P{V, A, R, B, H}

4 PATTERN-BASED CATEGORIZATION

For each indirect CC pattern, we first introduce a table that visualizes the features of each CC. According to these features, we will analyze the particular characteristics of each indirect CC pattern. Each table names the CC as designated in Sect. 2.2, its literature reference and the year of its publication. After these three columns, we first describe the features of the proxy and dead drop (if applicable). Afterwards, we describe the features of the applied protocols. Please note that we use ●, ○ and □ to ensure clarity. These symbols are applied to the Load Intensity (N) (● high, ○ normal), Fading (● timed, ○ logic, □ hybrid), the Lifetime variations (for each single characteristic v, t and p: ● possible, ○ not possible), Protocol Violation (● violation, ○ conclusive) Authentication (● authenticated, ○ unauthenticated, □ optional) and Remote accessibility (● remote, ○ local). However, some feature values have not been described or researched in the original sources. We decided to address these as unk. (unknown). If no protocol is specified or various protocols can be exploited, we will denote it as n.spec.

Second, we apply the naming conventions in a separate table.

4.1 Redirector Pattern

An overview of all redirector CCs is presented in Tab. 1 and the naming conventions are presented in Tab. 2. First, it seems like redirector CCs are mostly tailored for protocols without a (or mandatory) authentication. This may base on the fact that all implementations modify the response time, and this can be accomplished by numerous protocols. The bitrate is low and seems to be restricted to 1 bit. The hiding patterns used excessively for manipulation and extraction are the timing-based rate and throughput patterns ET2/RT.2n and ET1.1/RT1.1n. The CI is represented and embedded by the hiding pattern EN4.1/RN4.1n at the proxy. Indirect CCs implementing the proxy pattern had been known since 2003, but never had been further researched until 2019.

4.2 Proxy Pattern

The known proxy CCs and their features are presented in Tab. 3 and the naming conventions are applied in Tab. 4. Please note that for ARP.4, DHCP.2, HTTP.6, ICMP.2 and SSH.1, as described in [34], various protocols can be applied to extract information from the proxy. Such protocol switching was not described in [19, 33] but due to their similar implementation, we assume it will also be possible to extract CI with another protocol. This seems to be mutual for all known proxy CCs. However, we listed these CCs only once with the same protocol applied for manipulation and extraction. All proxy CCs resort to excessive use of network packets to influence the proxy’s acknowledgment behavior for other packets (which is recognizable by a CR). We assume it will also be possible to influence a proxy without numerous requests to represent CI, though it seems that until now it has never been implemented. This can for example be achieved by single requests, that block crucial resources of the proxy. For the existing CCs, the CI at the intermediary is always represented by a value like the CPU temperature or load. The utilized network protocols are not violated and for both, the manipulation and extraction, protocols that do not require authentication were preferred. Further, remote-access providing protocols are applied, though this may be because they are more widespread and common. These two observations can be explained by the fact that all implementations modify the response time, and this can be accomplished by numerous protocols. The bitrate is low and seems to be restricted to 1 bit. The hiding patterns used excessively for manipulation and extraction are the timing-based rate and throughput patterns ET2/RT.2n and ET1.1/RT1.1n. The CI is represented and embedded by the hiding pattern EN4.1/RN4.1n at the proxy. Indirect CCs implementing the proxy pattern had been known since 2003, but never had been further researched until 2019.

4.3 Dead Drop Pattern

Tab. 5 provides an overview of dead drop CCs and Tab. 6 applies the introduced naming conventions. We first analyze our observations for the intermediate system itself (i.e. CI lifetime and fading). Second, we analyze the features of the protocols that are used for the manipulation and extraction.

Dead Drop Features. The first dead drop CC had been described in 2003, nevertheless, a rapid growth of implementations over the last few years in various publications can be observed. Most dead drop CCs seem to exploit caches with logic fading. Until now, there are only few known timed fading caches (DNS.1, HTTP.2, HTTP.5 and NTP.1). The lifespan of the entries in the dead drop may either be exclusively volatile, temporary or persistent, or they can apply to all lifespan categories at the same time. Caches that have a single persistent and temporal CI lifespan seem to appear exclusively in combination with timed fading. It also can be observed that logic fading with a volatile lifespan occurs twice (NTP.3 and NET.1). Further, the CI capacity covers a large bandwidth from 1 bit to 800 bytes. The CI capacity depends on the nature of the dead drop and the number of maximum storable CIs. Furthermore, it is noticeable that almost all dead drop CCs use an information hiding embedding pattern that is based on a non-temporal modulation. In particular, the information hiding pattern EN4.1n/RN4.1n is most frequently used to embed CIs in dead drops. MQTT.11 has already been listed as a redirector, but can also be implemented as a dead drop if the CR has already subscribed to the topic at the time the CS sends the CI.
Table 1: Overview of Redirector Covert Channels

| Name       | Ref. | Description                                                                 |
|------------|------|-----------------------------------------------------------------------------|
| 802.11     | [54] | C.802.11f, u, l, log, n bit, ET1.1/RN1.1n                                   |
| ARP.3      | [34] | R.ARPv, u, r, 1024 B, EN4.2/RN4.1n                                         |
| CoAP.1     | [18] | R.CoAPf, u, r, log, n bit, ET1.1/RN1.1n                                    |
| DHCP.1     | [34] | R.DHCPv, u, r, 1024 B, EN4.2/RN4.1n                                       |
| HTTP.1     | [4]  | R.HTTPv, u, r, 1024 B, EN4.2/RN4.1n                                        |
| HTTP.3     | [4]  | R.HTTPv, u, r, 1024 B, EN4.2/RN4.1n                                        |
| HTTP.4     | [4]  | R.HTTPv, u, r, unk., RN1.1/RN1.1n                                         |
| ICMP.1     | [12, 51] | R.ICMPv, u, l, 4 B, EN4.2/RN4.1n                                           |
| ICMP.2     | [34] | R.IGMPv, o, r, 4 B, EN4.1/RN4.1n                                           |
| MQTT.1     | [42] | R.MQTTv, o, r, 4 B, EN4.1/RN4.1n                                           |
| MQTT.2     | [32] | R.MQTTv, o, r, n bit, EN2/RN2.n                                            |
| MQTT.3     | [32] | R.MQTTv, o, r, n bit, EN2/RN2.n                                            |
| MQTT.4     | [32] | R.MQTTv, o, r, 1 bit, EN1.1/RN1.1n                                        |
| MQTT.5     | [32] | R.MQTTv, o, r, 1 bit, EN1.1/RN1.1n                                        |
| MQTT.6     | [32] | R.MQTTv, o, r, 1 bit, EN1.1/RN1.1n                                        |
| TCP.1      | [35] | R.TCPv, u, r, 1024 B, EN2/RN2.n                                            |

Table 2: Naming Conventions Applied to Redirector Covert Channels

| Name       | Ref. | Transmission                                                                 |
|------------|------|-----------------------------------------------------------------------------|
| 802.11     | [54] | ○.802.11f, u, l, log, n bit, ET1.1/RN1.1n                                  |
| ARP.3      | [34] | ○.ARPv, u, r, 1024 B, EN4.2/RN4.1n                                        |
| CoAP.1     | [18] | ○.CoAPf, u, r, log, n bit, ET1.1/RN1.1n                                   |
| DHCP.1     | [34] | ○.DHCPv, u, r, 1024 B, EN4.2/RN4.1n                                       |
| HTTP.1     | [4]  | ○.HTTPv, u, r, 1024 B, EN4.2/RN4.1n                                        |
| HTTP.3     | [4]  | ○.HTTPv, u, r, 1024 B, EN4.2/RN4.1n                                        |
| HTTP.4     | [4]  | ○.HTTPv, u, r, 1024 B, EN4.2/RN4.1n                                        |
| ICMP.1     | [12, 51] | ○.ICMPv, u, l, 4 B, EN4.2/RN4.1n                                          |
| ICMP.2     | [34] | ○.IGMPv, o, r, 4 B, EN4.1/RN4.1n                                             |
| MQTT.1     | [42] | ○.MQTTv, o, r, 4 B, EN4.1/RN4.1n                                              |
| MQTT.2     | [32] | ○.MQTTv, o, r, n bit, EN2/RN2.n                                                  |
| MQTT.3     | [32] | ○.MQTTv, o, r, n bit, EN2/RN2.n                                                  |
| MQTT.4     | [32] | ○.MQTTv, o, r, 1 bit, EN1.1/RN1.1n                                               |
| MQTT.5     | [32] | ○.MQTTv, o, r, 1 bit, EN1.1/RN1.1n                                               |
| MQTT.6     | [32] | ○.MQTTv, o, r, 1 bit, EN1.1/RN1.1n                                               |
| TCP.1      | [35] | ○.TCPv, u, r, 1024 B, EN2/RN2.n                                               |

Manipulation & Extraction Features. The known dead drop CCs are not violating protocols and behave conclusively for both, their manipulation and extraction. Also, both remote and local communications can be exploited for the manipulation of dead drops. The extraction process of the CCs exploits only remote protocols to retrieve CI from the dead drop. Further, an authentication is usually not necessary to manipulate the dead drop. Only the MQTT CCs and NTP.1 can force the CS to use authentication. The only CC that necessarily requires an authentication for the manipulation is MQTT.6. Both, routable and local communications are used for storing CI at the dead drop, and the bitrate varies between 1 bit and 5 Byte. Like for the dead drop’s storage segment, a focus on non-temporal hiding patterns can be also observed in the manipulation and extraction segment. The information hiding representation pattern RN4.1n is used almost exclusively but the CCs MQTT.4, MQTT.5 and NET.1 also use timing-based information hiding patterns. During the extraction, the dead drop can force the CSs to authenticate themselves. Only with DNS.1, it is not possible to restrict the access. It can also be observed that the bitrate of the extraction is equivalent to the storage capacity. There seem to be no restrictions to limit a potential CI transfer, except for the amount of storable CI. The hiding pattern applied for the extraction also matches the hiding pattern that has been utilized for the manipulation. Though, it cannot be concluded that information hiding pattern-switching between manipulation and extraction is infeasible. Temporal information hiding patterns are used as well as non-temporal ones. Though, there seems to be a high focus on the information hiding pattern EN4.1/RN4.1n.

4.4 Discussion

After their first discovery in the late 1990s, indirect network CCs have been intensively analyzed by academic research. However, less than a third of all known CCs were described before 2019. It is remarkable that several indirect CC patterns have been rediscovered withing the last four years (maybe the earlier research was simply not widely recognized for over a decade). Our introduced taxonomy allows the systematic categorization of such indirect network CCs, even for future implementations. Especially the dead drop pattern emerged in various publications in academia for the last two years, probably because it is applicable in different scenarios. Though, we expect that direct CCs will continue to make up a large fraction of the CCs that can be observed in academia and potentially in real-world networks. However, we anticipate that indirect CCs might be used for more demanding scenarios than direct CCs — especially the one shown in Fig. 7, originally described in [34], seems to be interesting, as logically separated networks can be crossed, even if the CS and CR cannot communicate directly. Further, the timely distance between the CS and CR in case of dead drop CCs has a major advantage that is not well-understood by administrators and security experts. So far, there has been a focus on hiding patterns based on non-temporal behavior. In addition to the value modulation information hiding pattern, there are plenty of other information hiding pattern implementations conceivable, leading to the risk of undetected and unobstructed CI flows. An exception therefore are the proxy CCs, that, until now, rely on temporal behavior. Considered overall, it is difficult to make an exact classification of the best and most innovative covert channels, as...
these depend on various factors and each surveyed indirect CC contributes different ideas. If there is one to pick publication for these depend on various factors and each surveyed indirect CC pattern, in our opinion for the redirector pattern, [34] is impressive as the publication shows how redirectors can be implemented with various protocols, crossing logical separation. And to pick one publication for dead drops, we a large amount of CI utilizing a wide spread carrier protocol. For the redirector pattern, [34] is impressive as the publication shows how redirectors can be implemented with various protocols, crossing logical separation. And to pick one publication for dead drops, we

| Name   | Ref. | Year | Dead Drop Cl | Lifeline | Cap. | Hiding Patt. | Manipulation | Proxy | Extraction | Ref. | Description |
|--------|------|------|--------------|----------|------|--------------|--------------|-------|------------|------|-------------|
| ARP.1  | [38] | 2019 | 800 B        |          |      |              |              |       | 800 B      |      |             |
| ARP.2  | [38] | 2019 | 800 B        |          |      |              |              |       | 800 B      |      |             |
| DNS.1  | [2]  | 2005 |              |          |      |              |              |       |            |      |             |
| HTTP.2 | [2]  | 2005 |              |          |      |              |              |       |            |      |             |
| MQTT.1 | [42] | 2019 |              |          |      |              |              |       |            |      |             |
| MQTT.3 | [42] | 2019 |              |          |      |              |              |       |            |      |             |
| TCP.1  | [12] | 2008 |              |          |      |              |              |       |            |      |             |
| Table 5: Overview of Dead Drop Covert Channels |

| Name   | Ref. | Year | Default Drop Continuity (DCC) | Manipulation | Extraction | Ref. | Description |
|--------|------|------|-------------------------------|--------------|------------|------|-------------|
| ARP.1  | [38] | 2019 | ARP: u, l, 3 B, EN4.1/RN4.1n |              |            |      |             |
| ARP.2  | [38] | 2019 | ARP: u, l, 3 B, EN4.1/RN4.1n |              |            |      |             |
| DNS.1  | [2]  | 2005 | DNS: u, r, 1 bit, EN4.1/RN4.1n |              |            |      |             |
| HTTP.2 | [2]  | 2005 | HTTP: u, r, 4 kB, EN4.1/RN4.1n |              |            |      |             |
| MQTT.1 | [42] | 2019 | MQTT: u, r, 1 bit, EN4.1/RN4.1n |              |            |      |             |
| MQTT.3 | [42] | 2019 | MQTT: u, r, 1 bit, EN4.1/RN4.1n |              |            |      |             |
| TCP.1  | [12] | 2008 | TCP: u, r, 1 bit, EN4.1/RN4.1n |              |            |      |             |
| Table 6: Naming Conventions Applied to Dead Drop Covert Channels |
would like to emphasize the importance of [40] as it shows that the dead drop pattern can be utilized to access such a deeply rooted cache of a system, independent to a certain protocol.

5 APPLICATIONS AND COUNTERMEASURES

After we structured indirect CCs in patterns, we now can also discuss and compare application scenarios and countermeasures for each indirect CC pattern.

5.1 Application Scenarios

Although the implementations vary, the application scenarios for each CC of an indirect CC pattern are often similar. Though, not all application scenarios within the indirect CC pattern group are realizable with each indirect CC. In the following subsections, we will give an overview of the CCs of each indirect CC pattern.

5.1.1 Redirector Pattern. The basic application scenario for a redirector CC is visualized in Fig. 3. The first application scenario is that the CS forces the redirector to bounce an acknowledgement to the CR. The principle of this action is also exemplified in Fig. 10(a). Network subnets may be crossed, depending on the specification of the utilized protocol. Examples for this exploitation can be found in [12, 35, 50, 51] (ICMP.1 and TCP.1). Further, redirector CCs (ARP.3, DHCP.1 and TCP.1) have been applied to cross a logical network border [34]. The authors describe how a router logically separates two networks, the so-called node and guest network. Clients may be connected to one network but are not allowed to communicate to each other. The scenario is visualized in Fig. 7. The CS and CR are connected to a router, that separates the two networks of CS and CR. In the example of IGMP.1, the CS forces the router to send an IGMP membership-query via multicast to the CR, even if both, CS and CR, were logically separated. This procedure is also shown in Fig. 10(b). For ARP.1 and DHCP.1, a slightly varied method is applied. The source addresses are broadcast or multicast addresses, forcing the router to forward the message, even if the networks are separated. This can be achieved by a protocol violation as ARP- and DHCP-requests should never be sent from broadcast or multicast addresses. The method may also apply to other nodes than routers but has not been investigated yet. ICMP.1 and TCP.1 might also be available for this scenario as the source addresses are modified as described in Sect. 2.2 and can also contain broadcast or multicast addresses, which requires further investigation.

Figure 7: Logically Separated Covert Sender and Receiver

5.1.2 Proxy Pattern. The proxy CCs described in [34] have been applied in the same way as illustrated in Fig. 7 and already described in the previous section. However, the implementation does not rely on the redirection of packets. The channels ARP.4, DHCP.2, HTTP.6, ICMP.2 and SSH.1 influence a shared resource, the CPU of the router. The CS busies the proxy’s resources, and the resulting elongated packet runtimes can be recognized by the CR. This scenario can be applied to cross logical separation, due to the physical use of the same resource, and may also be applied to several other scenarios with physical shared resources between the CS and the CR. TCP.2 exploits this similarly, though the authors of [19, 33] have not described a logical separation. The intermediary is stressed by requests of the CS and delays its answer to the packets of CR. This may also be applied to the same scenario as in Fig. 7.

5.1.3 Dead Drop Pattern. The general scenario has already been visualized in Fig. 5(a). The CS stores CI at an intermediary system, where they remain until the CS extracts them. The application scenario of dead drops is in some cases limited to local communications. This applies to both manipulation (Fig. 8(a)) and extraction. The CS of ARP.1, ARP.2 and NTP.1, for example, can only apply this scenario as they are confined to their collision- and broadcast-domains. However, the extraction is implemented with a remotely usable protocol that may even pass firewalls and network borders due to seemingly legitimate utilization. SNMP has been used for both ARP channels, while NTP command-queries have been used for NTP.1. The other CCs are not limited by local communications and can also be exploited remotely, like presented in Fig. 8(b). Due to their seemingly legitimate protocol usage, they may pass firewalls and network borders both for the sending and receiving process. Further, especially DNS.1, NTP.3 and TCP.3 are available for long-distance communications as they can exploit public infrastructure servers (e.g., DNS and NTP) or servers in general (in case of TCP) as a dead drop. In some use cases, MQTT is also available for such a scenario. Though not yet described, also an application scenario with a logical separation of devices seems to be possible. Theoretically, the intermediate node with shared physical access to CS and CR as described in Fig. 7 can be exploited equivalent to the CCs already described in NTP.2, ARP.1, ARP.2 and TCP.3. NET.1 is further interesting because the CC is not bound to a scenario with a specific protocol.

Figure 8: Local and Remote Dead Drops

5.2 Potential and Evaluated Countermeasures

Indirect CCs always consist of a CS, an intermediate node, a CR and their according connections. At each of these elements, a warden
can be placed like visualized in Fig. 9 [43]. The wardens will be referred to as \( W_1 \) to \( W_5 \), corresponding to their location. All wardens may apply already existing countermeasures against direct CCs with existing steganalysis techniques. Eventually, the modification and influencing of temporal and non-temporal behavior is the basis to CI exchange between a CS and a CR. Existing countermeasures can be found, for example, in [3, 8, 13, 29, 50]. There are groups of countermeasures that can specially be applied for each indirect CC pattern. These will be described in the following subsections.

### 5.2.1 Redirector Pattern

The redirector often applies a redirection of acknowledgments without any information stored at the intermediate node. The normal behavior of such a protocol is shown in Fig. 10(b) for IGMP.1, while the violation is presented in Fig. 10(a) for TCP.1. The application of IGMP.1 is conclusive because it mimics normal behavior of the protocol, while TCP.1 sends an ACK to a host that never sent an associated SYN. Each single warden is in a position to detect this violation on its own. \( W_1 \) and \( W_2 \) will for example only observe the SYN packet but never the ACK. \( W_3 \) can detect the protocol-violating behavior of an acknowledgment sent to another host. \( W_4 \) and \( W_5 \) may detect an ACK in absence of a SYN, which is never supposed to happen (except for the so-called cold start problem [21]). Such a behavior is also observable with ARP.3, where the protocol is violated by broadcast source IPs that are not used for ARP-requests and can be observed by all wardens. However, a redirector CC does not always depend on protocol violations. DHCP.1 and IGMP.1 for example utilize broadcast and multicast messages combined with a legitimate acknowledgment. This is visualized in Fig. 10(b). However, the wardens may be able to detect this behavior anyway. \( W_1 \) and \( W_2 \) can detect an increased volume of packets from a single host, which may lead to detection. \( W_4 \) and \( W_5 \) may also recognize frequent broadcast and multicast messages but cannot determine the source and maybe have to assume network problems or more than one system causing these events. If a deviation from the normal amount of occurring network packets from a specific host is detected, the wardens \( W_1 \), \( W_2 \) and \( W_3 \) could also limit a potential covert channel. A targeted limitation typically requires the determination of normal network behavior and the application of detection heuristics, such as those focusing on inter-packet gaps (e.g., [7], [14]). Detected covert channels can then be slowed down or eliminated. Further, if a protocol is violated, the wardens \( W_4 \), \( W_5 \) and \( W_5 \) might block the suspicious traffic, for instance if only acknowledgments are detected. The wardens \( W_1 \) and \( W_2 \) are not able to block the traffic in advance, because the violation is not observable by them at this point.

### 5.2.2 Proxy Pattern

The proxy pattern influences an intermediate node in such a way that it represents the CI. This is often achieved by influencing the temperature or load of a CPU. Examples therefore are ARP.4, DHCP.2, HTTP.6, ICMP.2, SSH.1 and TCP.3. The wardens \( W_1 \), \( W_2 \) and \( W_5 \) can detect requests, which are repeated many times within a short timespan, especially if they are sent from a single node. Besides this, \( W_3 \) can also observe that the CPU temperature or load increases in intervals, encoding CI. The wardens \( W_4 \) and \( W_5 \) can detect delayed packets and the CI encoded within them. For this scenario, methods for detecting regularities in inter arrival times had been described in [7]. As all these CCs rely on delayed response times, the CI exchange can be limited and eliminated by response-time normalization performed by \( W_3 \) and \( W_4 \). Examples for normalization were originally described in [36] for timing side channels and can be achieved by randomized response delays or fixed response times.

### 5.2.3 Dead Drop Pattern

Dead drop CCs store CI at the intermediate node. Therefore, \( W_5 \) may detect them by an adequate technique. In particular, countermeasures against non-temporal behavior and value modulation like described for instance in [13] are good choices, as this is currently the dominating hiding pattern. However, also temporal hiding methods are conceivable, though not yet described for dead drops. Traditional approaches, such as the SRM (see below), might be applicable to eliminate the dead drops on \( W_5 \). Also, an abnormal number of entries in network protocol caches can be observed by \( W_3 \). It will be hard to detect manipulations as the frequency of packets can be varied by the CS because CI can remain for a longer timespan at the intermediary. The wardens \( W_1 \) and \( W_2 \) could apply countermeasure techniques that detect the hiding patterns RN4.1n and RT2.n. It is likely that the cache will be flushed after an extraction. \( W_5 \) may be able to detect this on its own as well as \( W_1 \) and \( W_2 \), if they are somehow connected to \( W_4 \) and \( W_5 \) or vice versa, or if any of these wardens are connected to \( W_5 \). The wardens \( W_5 \), \( W_4 \) and \( W_5 \) can detect a singular event that transfers CI to the CR as the number of extracted CI seems not to be controllable and correlates with the maximum of stored CI in the cache. TCP.3 influences the IPID counter of the dead drop; the CS and CR packets have to be well coordinated to transfer the CI, as the IPID counter would otherwise increase and destroy the CI. \( W_5 \) can detect the timely occurrence of these coordinated requests. This behavior of CS and CR can only be detected by \( W_1 \) and \( W_2 \) if they are connected to \( W_4 \) and \( W_5 \) (and vice versa), or if any of these wardens can exchange information with \( W_3 \). Further, \( W_4 \) and \( W_5 \) might apply a slightly modified detector for LSB steganography, as the least significant bit is the modified value. The limitation to a necessary number of entries can limit dead drop covert channels.

---

Figure 9: Warden Locations for Indirect Covert Channels

Figure 10: Protocol Violating and Conclusive Channel
We expect to observe the different indirect CC pattern-related ideas in modern IT environments, it is not necessary to allow that many entries, as the collision domains are most likely restricted to fewer hosts, or even only two communication partners. Further, Tab. 5 visualizes that protocols used for extraction often implement the ARP-caches, for example, can store up to 128/256 entries, depending on the operating system, and are exploited by ARP 1 and ARP 2. In modern IT environments, it is not necessary to allow the dead drop to hereby be secured at least against extraction by unauthorized CRs. Limitation and elimination can also be performed by the warden W3, as dead drop CCs base on CRs that are stored in network protocol caches of intermediate nodes. This can be achieved by regularly clearing caches, which leads to the destruction of CRs. Regarding the detection, one can assume increased detectability when a longer lifespan of CI in the cache is given.

SRM-based Protocol Cache Analysis. One envisaged approach to analyze communication protocols regarding their potential for cache exploitation and dead drops could be based on the classical Shared Resource Matrix (SRM) methodology, which was originally developed by Kemmerer [6, 20] and later enhanced by McHugh [30]. In the SRM, a software system is analyzed in a way that for each attribute a check is performed whether operations (performed by exclusive entities – CS and CR) can modify/read the attribute. If such a modify-read cycle is feasible, a potential covert channel is discovered. This concept could be adapted for communication protocol caches in a way that the protocol states and cache entries are considered as attributes and the sending and receiving actions are considered as operations.

6 THE ROAD AHEAD

We expect to observe the different indirect CC pattern-related ideas to be applied to additional network protocols and network environments in the future. Especially IPv6 and IoT networks will be an interesting research field for indirect CCs due to their increasing deployment. Further protocol types, such as VoIP-related ones or low-level protocols like Bluetooth, have not yet been investigated but can be considered to be exploitable, too. In [34], the authors mentioned an idea for the proxy pattern by passing physical barriers. This opens several options for proxies and might also be applied to indirect dead drop CCs by exploiting shared network resources like WiFi hotspots comparable to [54] or even specialized network-connected and shared devices like printers; several resources that have to be accessible by clients from various networks can act as intermediate nodes for indirect CCs. Further, a maintainable development has been described in [40]. The authors exploited a cache that is not considered to be a network-protocol cache and was never meant to be accessible by a remote device. Another crucial development can be observed in [9]. Therein, the authors describe how to exploit the cloud service Dropbox to exchange information between a CS and a CR by modifying files and their meta-data. The article has not been covered in our survey as it mixes network- and application-level CCs. However, the article is important for two reasons. First, it exploits a cloud service, which has not yet been well-investigated for indirect CCs. Second, it mixes concepts as it represents a novel hybrid class of CCs. Finally, future research work might show whether proxies without high load intensity (N) and dead drops with hybrid fading (F) can be realized. We assume that NET I can be applied as a hybrid fading dead drop, however this has not been analyzed in [40].

7 CONCLUSION

We described the state and commonalities of the known indirect network-level covert channels (CCs). Similarities of these CCs were transferred into a taxonomy, which consists of three essential indirect CC patterns: the redirector, the proxy, and the dead drop. We provided definitions for the introduced terms, identified crucial features of indirect CCs and introduced naming conventions. This enabled us to systematically present and compare indirect CCs. Additionally, we described application scenarios, identified possible locations of wardens as well as potential and already evaluated countermeasures against each indirect CC pattern and potential future developments.

Our results have shown that the majority of indirect CCs appeared within the last three years, which can be considered a hint for a growing trend. Further, we determined that the known indirect CC patterns applied only a subset of the known information hiding patterns, i.e. several of the known steganography hiding techniques are currently not exploited for indirect CCs. Additionally, some of the information hiding patterns appeared to only be used for specific indirect CC patterns. The application of countermeasures against the most recent indirect CCs has not yet been investigated in full detail. We hope that our systematization of indirect CCs aids the development of precise countermeasures against current and upcoming indirect CCs.

REFERENCES

[1] F. M. Alharbi, Y. Zhou, F. Qian, et al. 2022. DNS Poisoning of Operating System Caches: Attacks and Mitigations. IEEE Trans. Dependable and Secure Computing (2022). https://doi.org/10.1109/TDSC.2022.3142331
[2] Anonymous Authors. 2005. DNS Covert Channels and Bouncing Techniques. https://seclists.org/fullDisclosure/2005Jul/alt-452/p63_dns_worm_covert_channel.txt
[3] A. Ayub, S. Smith, and A. Siraj. 2019. A Protocol Independent Approach in Network Covert Channel Detection. In 2019 IEEE Int. Conf. Computational Science and Engineering, CSE 2019, and IEEE Int. Conf. Embedded and Ubiquitous Computing, EUC 2019, IEEE, 165–170. https://doi.org/10.1109/CSE/EUC.2019.00040
[4] M. Bauer. 2003. New covert channels in HTTP: adding unwitting Web browsers to anonymity sets. In Proc. of the 2003 ACM Workshop on Privacy in the Electronic Society. WPES 2003. ACM, 72–78. https://doi.org/10.1145/1005140.1005152
[5] L. Baumann, E. Heftrigs, H. Shulman, and M. Waidner. 2021. The Master and Parasite Attack. In 2021 51st Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN). 141–148. https://doi.org/10.1109/DSS4987.2021.00029
[6] M. Bishop. 2018. Computer Security Art and Science (2 ed.). Addison-Wesley Prof.
[7] Serdar Cabuk, Carla E. Brodley, and Clay Shields. 2009. IP Covert Channel Detection. ACM Trans. Inf. Syst. Secur. 12, 4 (2009), 22:1–22:29. https://doi.org/10.1145/1513601.1513604
[8] L. Caviglione. 2021. Trends and Challenges in Network Covert Channels Countermeasures. Applied Sciences 11 (02 2021). https://doi.org/10.3390/app11021641
[9] L. Caviglione, M. Podolski, W. Mazurczyk, and M. Ianiro. 2017. Covert Channels in Personal Cloud Storage Services. The Case of Dropbox. IEEE Trans. Ind. Informatics 13, 4 (2017), 1921–1931. https://doi.org/10.1109/TII.2016.2627503
[10] M. Cunche, M. A. Kääflar, and R. Boreli. 2014. Asynchronous Covert Communication Using BitTorrent Trackers. In IEEE Int. Conf. on High Performance Computing and Communications, 6th IEEE Int. Symp. on Cyberspace Safety and Security, 11th IEEE Int. Conf. on Embedded Software and Systems, HPCC/CSS/ICESS 2014. IEEE, 827–830. https://doi.org/10.1109/HPCCC.2014.138
[11] G. Daneault and D. Johnson. 2016. Client-initiated HTTP covert channels using relays. In 2016 4th International Symposium on Digital Forensic and Security (ISDFS). 32–37. https://doi.org/10.1109/ISDFS.2016.7473513
[12] G. Danezis. 2008. Covert Communications Despite Traffic Data Retention. In Security Protocols XV – 16th International Workshop, Cambridge, UK, April 16-18, 2008. Revised Selected Papers (LNCS, Vol. 6615). Springer, 198–214. https://
T. Schmidbauer and S. Wendzel. 2020. Covert storage caches using the NTP.

S. Schinzel. 2012.

C. H. Rowland. 1997. Covert Channels in the TCP/IP Protocol Suite.

A. Ovadia, R. Ogen, Y. Mallah, N. Gilboa, and Y. Oren. 2019. Cross-Router Covert Channels. In Proc. ECNS (2005).

C. Gray Girling. 1987. Covert Channels in LANs. IEEE Transactions on software engineering 13, 2 (1987), 292.

S. Z. Goher, B. Javed, and N. A. Saqib. 2012. Covert channel detection: A survey based analysis. In High Capacity Optical Networks and Emerging/Enabling Technologies. pp. 657–695. https://doi.org/10.1109/HONET.2012.6424345

T. G. Handel and M. T. Sandford II. 1996. Hiding Data in the OSI Network Model. In Information Hiding, First International Workshop, May 30 – June 1, 1996, Proc. (LNCS, Vol. 1174). Springer, 23–38. https://doi.org/10.1007/3-540-61996-8_29

L. Hartmann, S. Zillien, and S. Wendzel. 2011. Reset- and Reconnection-based Covert Channels in CoAP. In EICC 2021: European Interdisciplinary Cybersecurity Conference. ACM. https://doi.org/10.1145/3474053.3476600

A. Hints. 2003. Covert Channels in TCP and IP Headers. https://www.defcorn.org/images/defcorn-10-dc-10-presentations/de10-hints-coverpt.ppt.

R. A. Kemmerer. 1983. Shared resource matrix methodology: An approach to identifying storage and timing channels. ACM Transactions on Computer Systems (TOCS) 1, 3 (1983), 256–277.

Christian Kreieh, Mark Handley, and V. Paxson. 2001. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In Proc. USENIX Security Symposium, Vol. 2001.

W. L. Lampson. 1993. A Note on the Connection Problem. Commun. ACM 16, 10 (1973), 613–615. https://doi.org/10.1145/383725.362839

C. Lavaud, R. Gerzague, M. Gastier, et al. 2021. Whispering Devices: A Survey on How Side-channels Lead to Compromised Information. J. Hardw. Syst. Secur. 5, 3 (2021), 143–168. https://doi.org/10.1109/JSWS.2021.00124-6

M. Lipp, M. Schwarz, L. Raab, L. Lamster, M. T. Aga, C. Maurice, and D. Gruss. 2020. Netherhammer: Inducing Rowhammer Faults through Network Requests. In 2020 IEEE Eu. Symp. on Sec. and Priv. Workshops (EuroSPW) PW, 710–719. https://doi.org/10.1109/EuroSPW51379.2020.00102

D. Llanas, C. Allisson, and A. Miller. 2005. Covert channels in internet protocols: A survey. In Proc. 6th Annual Postgraduate Symposium about the Convergence of Telecommunications, Networking and Broadcasting, PENET.

K. Man, Z. Qian, Z. Wang, X. Zheng, Y. Huang, and H. Duan. 2020. DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels. In CCS ’20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9–13, 2020. ACM, 1337–1350. https://doi.org/10.1145/3372297.3417280

C. Maurice, M. Weber, M. Schwarz, L. Giner, D. Gruss, C. A. Boano, S. Mangard, and K. Römer. 2017. Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud. In 24th Ann. Net. and Dist. Sys. Sec. Symposium, NDSS 2017, February 26 - March 1, 2017. The Internet Society.

W. Mazurczyk. 2013. VoIP steganography and its detection—a survey. ACM Computing Surveys (CSUR) 46, 2 (2013), 1–21.

W. Mazurczyk, S. Wendzel, S. Zillien, A. Houmansadr, and K. Szczypiorksi. 2016. Information Hiding in Communication Networks: Fundamental, Mechanisms, Applications, and Countermeasures (1st ed.). Wiley-IEEE Press.

J. McHugh. 2001. An information flow tool for gypsy-an extended abstract. In Proc. 17th Annual Comp. Sec. Applications Conf. (ET1.1).

I. Merabti, S. Zander, B. Fechner, and C. Herdin. 2015. Pattern-Based Survey and Categorization of Network Covert Channel Techniques. ACM Comput. Surv. 47, 3 (2015), 1–50. https://doi.org/10.1145/2759201.2759212

Manfred Wolf. 1989. Covert channels in LAN protocols. In Local Area Network Security Workshop. Springer, 89–101.

K. S. Yim. 2016. The Rowhammer Attack Injection Methodology. In 2016 IEEE 38th Symp. on Rel. Dist. Systems (SRDS) 1–10. https://doi.org/10.1109/SRDS.2016.012

J. Yu, M. Yan, A. Khyzha, et al. 2021. Speculative taint tracking (STT): a comprehensive protection for speculatively accessed data. Commun. ACM 64, 12 (2021), 105–112. https://doi.org/10.1145/3491201

S. Wendzel W. Mazurczyk and J. Keller M. Chourib. 2019. Countering adaptive network covert communication with dynamic wardens. In ARES 2021: The 16th ARES Conference. ACM. https://doi.org/10.1145/3465841.3470069

S. Wendzel, W. Mazurczyk, and G. Haas. 2017. Steganography for Cyber-PhysicalSystems. J. Cyber Sec. Mobil. 6, 2 (2017), 105–126. https://doi.org/10.1007/jcss/2017-119.621

S. Wendzel, S. Zillien, B. Fechner, and C. Herdin. 2015. Pattern-Based Survey and Categorization of Network Covert Channel Techniques. ACM Comput. Surv. 47, 3 (2015), 1–50. https://doi.org/10.1145/2759201.2759212

I. Zelenchuk. 2004. Skeevy – ICMP Bounce Tunnel. https://www.gray-world.net/poc_skeevy.shtml.

X. Zheng, C. Lu, J. Peng, et al. 2020. Poison Over Troubled Forwarders: A Cache Poisoning Attack Targeting DNS Forwarding Devices. In 29th USENIX Sec. Symp. USENIX Assoc., 577–593.

K. Z. Yang and Z. Yong. 2009. Entropy based taxonomy of network covert channels. In Proc. Int. Conference on Power Electronics and Intelligent Transportation System (PEITS) 451–455. https://doi.org/10.1109/PEITS.2009.5406973

S. Zillien and S. Wendzel. 2021. Reconnection-Based Covert Channels in Wireless Networks. In Proc. 36th IFIP Int. Conf. on ICT Systems Security and Privacy Protection (SEC 2021). Springer, 118–133. https://doi.org/10.1007/978-3-030-78120-9_0

INFORMATION HIDING PATTERNS

We introduced indirect CC patterns while we also employed another form of patterns — namely information hiding patterns. While several of these information hiding patterns are defined in the original work [44], we only cover those information hiding patterns that we actually refer to in this paper. Moreover, the patterns taxonomy currently faces a revision which will result in improved pattern descriptions and will be available here: https://patterns.ztt.hs-worms.de.

Embedding patterns. All embedding patterns start with the upper-case letter E, followed by a letter that indicates the type (temporal (T) or non-temporal (N)) and an identification number. Embedding patterns describe how information is introduced into a steganographic carrier. We referred to the following embedding patterns:

ET1.1 Rate/Throughput Modulation: This pattern embeds a secret message by modulating the rate of events/elements. In the network CC context, events are, e.g., the appearance of a flow or a disconnect, while elements are network packets.

ET2 Event Occurrence: The secret message is encoded through the temporal location of events; for instance the time at

559
which a disconnect is triggered or at which a connection is established [44].

EN2 Elements/Features Positioning: The secret data is encoded through the position of elements/features in a sequence of the same (features are, e.g., the packet size or a header field). [44] provides the example of placing an IPv4 option header at a specific location of the option’s list.

EN3 Elements/Features Enumeration. The overall number of appearances of elements or features is altered to encode a secret message.

EN4.1 Reserved/Unused (State)/Value Modulation: This pattern embeds the secret message by modulating reserved or unused states/values. In the network context this is, e.g., a value of a packet header field. A classical example is to store a secret bit in the unused IPv4 reserved bit.

EN4.2 Random State/Value Modulation; in comparison to EN4.1 the modified packet header value is not reserved or unused but a randomized field, such as the TCP Initial Sequence Number (ISN), which predominantly allows the embedding of secret messages that appear pseudo-random (e.g., encrypted data).

Representation patterns. It is possible for a CC to correspond to a different embedding and representation pattern at the same time [44]. Representation patterns start with an R and describe how a previously embedded steganographic information is represented so that it can be recognized by a CR. The trailing letter (n) of each named pattern indicates its context, which is networking (in contrast to, e.g., f for filesystem steganography). We referred to the following representation patterns:

RT1.1 Rate/Throughput Modulation: This pattern is the counterpart to the embedding pattern ET1.1. Here, the rate or throughput of a network connection is monitored by the CR to extract the secret information.

RT2.n Event Occurrence: counterpart to ET2 (the secret information is represented by the occurrence of events visible for CR).

RN2.n Elements/Features Positioning: counterpart to EN2.

RN3.n Elements/Features Enumeration: counterpart to EN3.

RN4.1.n Reserved/Unused State/Value Modulation: counterpart to EN4.1.

RN4.2.n Random State/Value Modulation: counterpart to EN4.2.

B PATTERN-SPECIFIC FEATURE OVERVIEW

We defined patterns for indirect network-level CCs and introduced crucial features for each indirect CC pattern. To ensure clarity and facilitate the description of new CCs, the following tables provide an overview of each pattern’s features.

The redirector pattern can be found in Tab. 7, while features of the proxy pattern are shown in Tab. 8, followed by the those of the dead drop pattern in Tab. 9.