New Covert Channels in Internet of Things

Aleksandra Mileva, Aleksandar Velinov, Done Stojanov
Faculty of Computer Science
University "Goce Delčev"
Štip, Republic of Macedonia
email: {aleksandra.mileva, aleksandar.velinov, done.stojanov}@ugd.edu.mk

***Abstract* — Network steganography is a relatively new discipline which studies different steganographic techniques that utilize network protocols for data hiding. Internet of Things (IoT) is a concept which integrates billions of embedded devices that communicate to each other. To the best of our knowledge, there are not many attempts that utilize existing network steganographic techniques in protocols specifically created for IoT. Therefore, in this paper, we present several new covert channels that utilize the Constrained Application Protocol (CoAP), which is a specialized Web transfer protocol used for constrained devices and networks. This protocol can be used regardless of its transport carrier (Datagram Transport Layer Security - DTLS or clear UDP – User Datagram Protocol). The suggested covert channels are categorized according to the pattern-based classification, and, for each covert channel, the total number of hidden data bits transmitted per CoAP message or its Packet Raw Bit Rate (PRBP) is given.**

Keywords-CoAP; network steganography; covert channels; data hiding.

I. INTRODUCTION

Network covert channels are used to hide data in legitimate transmissions in communication networks by deploying different network protocols as carriers and concealing the presence of hidden data from network devices. Covert channels (first introduced by Lampson [8]) can be divided in two basic groups: storage and timing channels. Storage covert channels are channels where one process writes (directly or indirectly) to a shared resource, while another process reads from it. In the context of network steganography, storage covert channels hide data by storing them in the protocol header and/or in the Protocol Data Unit (PDU). On the other hand, timing channels hide data by deploying some form of timing of events, such as retransmitting the same PDU several times, or changing the packet order.

Network-based covert channels may have black hat or white hat applications. Black hat applications include coordination of distributed denial of service attacks, spreading of malware (for example, by hiding command and control traffic of botnets), industrial espionage, secret communication between terrorists and criminals, etc. On the other hand, white hat applications include covert military communication in hostile environments, prevention of detection of illicit information transferred by journalists or whistle-blowers, circumvention of the limitation in using Internet in some countries (e.g., Infranet [3]), providing Quality of Service - QoS for Voice over Internet Protocol - VoIP traffic [10], secure network management communication [5], watermarking of network flows (e.g., RAINBOW [6]), tracing encrypted attack traffic or tracking anonymous peer-to-peer VoIP calls [16][17], etc.

Nowadays, there are a plenty of choices in the landscape of network protocols for carriers. There are several surveys about different covert channels in many TCP/IP (Transmission Control Protocol/Internet Protocol) protocols [12][19]. To the best of our knowledge, there are only a few papers about network steganographic research addressing protocols specialized for constrained devices in the IoT (sensors, vehicles, home appliances, wearable devices, and so on) [2] [7]. The Constrained Application Protocol (CoAP) [15] is a specialized Web transfer application layer protocol which can be used with constrained nodes and constrained networks in the IoT. The nodes are constrained because they have 8-bit microcontrollers, for example, with limited random-access memory (RAM) and read-only memory (ROM). Constrained networks often have high packet error rates and small data rate (such as IPv6 over Low-Power Wireless Personal Area Networks - 6LoWPANs). CoAP is designed for machine-to-machine (M2M) applications and its last stable version was published in June 2014 in the RFC 7252 [15]. In fact, it is a Representational State Transfer - RESTful protocol with multicast and observe support. In this paper, we try to apply existing network steganographic techniques for creating covert channels in CoAP.

Wendzel et al. [18] presented a new pattern-based categorization of network covert channel techniques into 11 different patterns or classes. They represented the patterns in a hierarchical catalog using the pattern language Pattern Language Markup Language (PLML) v. 1.1 [4]. In our paper, we use their classification to characterize our covert channels.

Covert channels are analyzed through the total number of hidden data bits transmitted per second (Raw Bit Rate - RBR), or through the total number of hidden data bits transmitted per PDU (for example, Packet Raw Bit Rate-PRBR) [11]. For each new CoAP channel, its PRBR value is given, where PDU is a CoAP message.

The rest of this article is structured as follows. The related work is presented in Section 2. Details about the

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-661-3

CoAP header, messages, functionalities and concepts are presented in Section 3. The main Section 4 describes eight groups of new covert storage and timing channels in CoAP, that can be used regardless its transport carrier (DTLS or clear UDP). Some possible applications of these covert channels are also briefly suggested in this section. In Section 5 we present the performance evaluation. We conclude the paper in Section 6.

II. Related Work

The research on network steganography for IoT has seen an increased interest recently.

One example for this is the work of Islam et al. [7], which uses Internet Control Message Protocol (ICMP) covert channels for authenticating Internet packet routers as an intermediate step towards proximal geolocation of IoT devices. This is useful as a defense from the knowledgeable adversary that might attempt to evade or forge the geolocation. Hidden data are stored in the data field of the ICMP Echo Request and ICMP Echo Reply messages.

Some applications of steganography in IoT are not connected with the protocols themselves, but with the applications on top of these protocols. For example, Denney et al. [2] present a novel storage covert channel on wearable devices that sends data to other applications, or even to other nearby devices, through the use of notifications that are normally displayed on the status bar of an Android device. For that purpose, a notification listening service on the wearables needs to be implemented. Data are hidden in the notification ID numbers (32 bits), and their exchange is done by using two functions notify and cancel. If the notifying function is immediately followed by the canceling function, the notification is never displayed to the user although it can be seen in the log files, so the communication is hidden from the user who wears the device.

There are several papers that deploy steganography in the physical and medium access control (MAC) layers of the IEEE 802.15.4 standard [9][13].

III. How CoAP Works

Similar to HTTP, CoAP uses client/server model with request/response messages. It supports built-in discovery of services and resources, Uniform resource identifiers (URIs) and Internet media types. The CoAP sends a request message requesting an action (using a Method Code) to the resource (identified by a URI) hosted on a server. The server responds to this request by using the response message that contains the Response Code, and possibly some resource representation. CoAP defines four types of messages: Confirmable (CON), Non-Confirmable (NON), Acknowledgment (ACK) and Reset (RST). These types of messages use method and response codes to transmit requests or answers. The requests can be transmitted as Confirmable and Non-Confirmable types of messages, while the responses can be transmitted through these and via piggybacked and Acknowledgment types of messages.

CoAP uses clear UDP or DTLS on transport layer to exchange messages asynchronously between endpoints. As shown in Figure 1, each message contains a Message ID used for optimal reliability and to detect duplicates. A message that requires reliable transmission is marked as CON, and if does not, it is marked as NON. The CON message is retransmitted using a default timeout and binary exponential back-off algorithm for increasing the timeout between retransmissions, until the recipient sends an ACK message with the same Message ID. When the recipient is

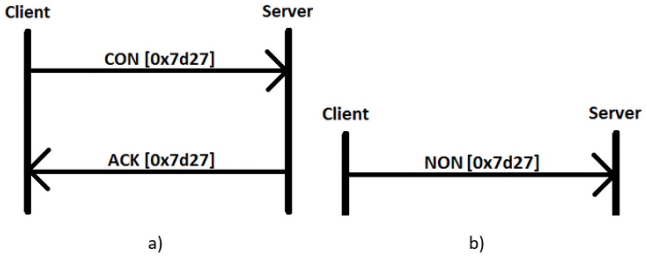

Figure 1. a) Reliable CoAP message transmission b) Unreliable CoAP message transmission.

not able at all to process CON or NON messages, it replies with a RST message.

CoAP messages are encoded into simple binary format (see Figure 2). Each message starts with a 4B fixed header, followed by a Token field, with size from 0 to 8B. Then comes the optional Options field and optional Payload field. If the Payload field is present it is preceded by one-byte Payload Marker (0xFF).

The fields that make up the message header are the following:

- Version (Ver) - 2-bit unsigned integer that identifies the CoAP version. Currently it must be set to 01.
- Type (T) – 2-bit unsigned integer that indicates the message type: Confirmable (0), Non-Confirmable(1), Acknowledgement (2), or Reset (3).
- Token Length (TKL) – 4-bit unsigned integer that stands for the length of the Token field (0-64 bits). Lengths 9-15 are reserved and must be processed as a message format error.
- Code – 8-bit unsigned integer. It is divided into two parts: 3-bit class (the most significant bits) and 5-bit details (the least significant bits). The format of the code is “c.dd”, where “c” is a digit from 0 to 7 and represents the class while “dd” are two digits from 00 to 31. According to the class we can determine the type of the message, such as: request (0), a successful response (2), a client error response (4), or a server error response (5). CoAP has a separate code registry that provides a description for all codes [1].
- Message ID - 16-bit unsigned integer that is used to detect duplicate messages and to connect Acknowledgment/Reset messages with Confirmable/ Non-Confirmable messages.

The message header is followed by the Token field with variable size from 0 to 64 bits. This field is used to link requests and responses.

The optional Options field defines one or more options. CoAP defines a single set of options that are used both for

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-661-3

requests and for responses. These are: Content-Format, Etag, Location-Path, Location-Query, Max-Age, Proxy-Uri, Proxy-Scheme, Uri-Host, Uri-Path, Uri-Port, Uri-Query, Accept, If-Match, If-None-Match, and Size1.

The payload of requests/responses that indicates success typically carries the resource representation or the result of the requested action.

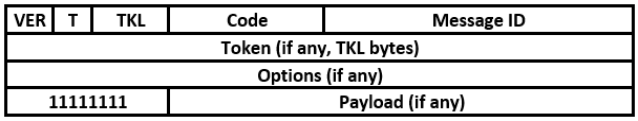

Figure 2. CoAP message format.

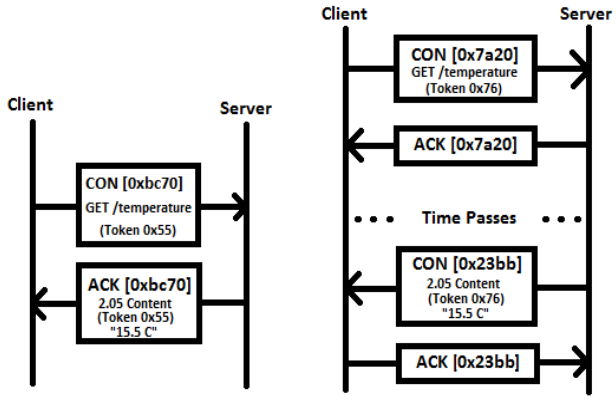

Figure 3. a) Piggybacked response b) Separate response.

There are two types of responses: piggybacked and separate (Figure 3). If the request is transmitted via CON or NON message, and if the response is available and transmitted via an ACK message, then it is piggybacked response. If the server is unable to respond immediately to the request, an Empty message (with code 0.00) is sent that tells the client to stop sending the request. If the server is able for later respond to the client, it sends a CON message that must then be confirmed by the client. This is called a separate response.

Similar to HTTP, CoAP uses GET (with code 0.01), POST (with code 0.02), PUT (with code 0.03), and DELETE (with code 0.04) methods.

IV. New Covert Channels in the CoAP

When someone creates a Covert Channel (CC) in network protocol, usually uses: a protocol feature that has a dual nature (i.e., the same feature can be obtained in more than one way), a feature that is not mandatory, a feature that can obtain random value, and so on. Therefore, if we use some of these features, we can create new covert channels in CoAP. From the beginning, CoAP offers some protection against network steganography. For example, by introducing a proper order in the appearance of different options in message, the steganographic techniques that deploy a different order of options can not be applied.

CoAP can be applied in different fields, such as: smart energy, smart grid, building control, intelligent lighting control, industrial control systems, asset tracking, environment monitoring, and so on. So, one useful scenario of application of the CoAP covert channels would be for support of the authentication of geolocation of IoT devices. Another possible scenario is clandestine communication between wearable devices in a hostile environment, for the needs of the soldiers, or, between nodes in a wireless sensor network.

As steganography offers security only through obscurity. A successful attack against any covert channel consists in detecting the existence of this communication. Next, the new CoAP covert channels are presented.

A. Covert Channel Using Token and/or Message ID Fields

The Message ID contains a random 16-bit value. In the case of piggybacked response for CON message, the Message ID should be the same as in the request, while in the case of separate response, the server generate different random Message ID (while the request Message ID is copied in the first sent Empty ACK message).

The same Message ID can not be reused (in the communication between same two endpoints) within the EXCHANGE_LIFETIME, which is around 247 seconds with the default transmission parameters.

The Token is another random generated field, with variable size up to 64 bits, used as a client-local identifier to make a difference between concurrent requests. If the request results in the response, the Token value should be echoed in that response. This also happens in the case when the server sends separate response. So, we can create an unidirectional or a bidirectional communication channel between two hosts, by sending 16 (from Message ID) plus/or 64 (from Token ID) bits per message (PRBR $\in$ {16, 64, 80}). According to the pattern-based classification [18], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Non-Payload
  --Structure Preserving
    --Modification of an Attribute
     --Random Value Pattern
```

B. Covert Channel Using Piggybacked and Separate Response

Since the server has a choice for sending piggybacked or separate response, one can create an one-bit per message unidirectional or a bidirectional covert channel (PRBR=1), such as:

- piggybacked response to be binary 1, and
- separate response to be binary 0.

At heavy load, the server may not be able to respond (sending binary 1), so this covert channel is limited to the times when the server has the choice. According to the

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-661-3

pattern-based classification [18], this channel belongs to the following class:

```
Network Covert Timing Channels
 --PDU Order Pattern
```

C. Covert Channel Using Payload of the Message

Both requests and responses may include a payload, depending of the Method or the Response Code, respectively. Its format is specified by the Internet media type and content coding providen by the Content-Format option. The payload of requests or of responses that indicates success is typically a representation of the resource or the result of the requested action.

If no Content-Format option is given, the payload of responses indicating client or server error is a Diagnostic Payload, with brief human-readable diagnostic message being encoded using UTF-8 (Unicode Transformation Format) in Net-Unicode form.

The CoAP specification provides only an upper bound to the message size - to fit within a single IP datagram (and into one UDP payload). The maximal size of the IPv4 datagram is 65,535B, but this can not be applied to constrained devices and networks. According to IPv4 specification in the RFC 791, all hosts have to be prepared to accept datagrams of up to 576B, while IPv6 requires the maximum transmission unit (MTU) to be at least 1280B. The absolute minimum value of the IP MTU for IPv4 is 68B, which would leave at most 35B for a CoAP payload (the smallest CoAP header size with Payload Marker before the payload is 5B, assuming 0B for Token and no options). On the other hand, constrained network presents another restriction. For example, the IEEE 802.15.4's standard packet size is 127B (with 25B of maximum frame overhead), which leaves (without any security features) 102B for upper layers. The sizes of the input/output buffers in the constrained devices are another restriction of the maximal payload. Thus, we can create a unidirectional or a bidirectional communication channel between two hosts, by sending a Diagnostic Payload with the smallest maximal size of 35B per message (PRBR=280). According to the pattern-based classification [18], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Payload Pattern
```

Another similar channel can be created by encoding the data in some specific Internet media format (for example, “application/xml” media type) and sending this format as payload of a message with appropriate Content-Format option (41 for “application/xml”).

D. Covert Channel Using Case-insensitive Parts of the URIs

CoAP uses “coap” and “coaps” URI (Uniform Resource Identifier) schemes for identification of CoAP resources and providing a means for locating the resource. The URI in the request are transported in several options: URI-host, URI-Path, URI-Port and URI-Query. They are used to specify the target resource of a request to CoAP origin server. The URI-host and the scheme are case insensitive, while all other components are case-sensitive. So, we can create a unidirectional covert channel between the client and the server using, for example:

- capital letter in the URI-host option to be binary 1, and
- lowercase letter in the URI-host option to be binary 0.

Taking into account that a valid Domain Name System (DNS) name has at most 255B, we can send at most 255B per message, or in other words, the PRBR of this channel is up to 255B. According to the pattern-based classification [18], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Non-Payload
  --Structure Preserving
   --Modification of an Attribute
    --Value Modulation
     --Case Pattern
```

CoAP supports proxying, where proxy is a CoAP endpoint that can be tasked by CoAP clients to perform requests on their behalf. Proxies can be explicitly selected by clients, using Proxi-URI option, and this role is “forward-proxy”. Proxies can also be inserted to stand in for origin servers, a role that is named as "reverse-proxy". So, we can create similar covert channel using schema and host part from the Proxi-URI option. A request containing the Proxy-URI Option must not include URI-host, URI-Path, URI-Port and URI-Query options.

E. Covert Channel Using PUT and DELETE Methods

The PUT method requires the resource identified by the URI in the request, to be updated or created with the enclosed representation. If the resource exists at the request URI, the enclosed representation should be considered as a modified version of that resource, and a 2.04 (Changed) Response Code should be returned. If no resource exists, then the server may create a new resource with the same URI that results in a 2.01 (Created) Response Code.

The DELETE method requires deletion of the resource, which is identified by the URI in the request. Regardless if the deletion is successful, or the resource did not exist before the request, a 2.02 (Deleted) Response Code should be send.

If somebody has a known representation of the existing resource R1 on the server and if he knows that specific resource R2 does not exist on the same server, a unidirectional covert channel to the server can be created, in this way:

- send request with PUT method to create the resource R1 with enclosed known representation as binary 1, and
- send request with DELETE method to delete non-existing resource R2 as binary 0.

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-661-3

In this way, one bit per message can be sent (PRBP=1). According to the pattern-based classification [18], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Non-Payload
  --Structure Preserving
   --Modification of an Attribute
    --Value Modulation Pattern
```

F. Covert Channel Using Accept Option

The Accept option can be used to indicate which Content-Format is acceptable to the client. If no Accept option is given, the client does not express a preference. If the preferred Content-Format if available, the server returns in that format, otherwise, a 4.06 "Not Acceptable" must be sent as a response, unless another error code takes precedence for this response. We can create a unidirectional one-bit per message covert channel (PRBP=1), in this way:

- sending a given message without Accept option to be binary 1, and
- sending a given message with Accept option to be binary 0.

According to the pattern-based classification [18], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Non-Payload
  --Structure Preserving
   --Modification of an Attribute
    --Value Modulation Pattern
```

G. Covert Channel Using Conditional Requests

Conditional request options If-Match and If-None-Match enable a client to ask the server to perform the request only if certain conditions specified by the option are fulfilled. In the case of multiple If-Match options the client can make a conditional request on the current existence or value of an ETag for one or more representations of the target resource. This is useful to update the request of the resource, as a means for protecting against accidental overwrites when multiple clients are acting in parallel on the same resource. The condition is not fulfilled if none of the options match. With If-None-Match option the client can make a conditional request on the current nonexistence of a given resource. If the target resource does exist, then the condition is not fulfilled.

If somebody knows for sure that given condition C1 is fulfilled (for example, the resource is created or deleted in previous message) and other C2 is not fulfilled, using either of If-Match and If-None-Match options, a unidirectional one-bit per message covert channel (PRBP=1) can be created in this way:

- sending a given message without fulfilled condition to be binary 1 (e.g., If-Match + C2), and
- sending a given message with fulfilled condition (e.g., If-Match + C1) to be binary 0.

According to the pattern-based classification [18], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Non-Payload
  --Structure Preserving
   --Modification of an Attribute
    --Value Modulation Pattern
```

H. Covert Channel Using Re-Transmissions

If we are using CoAP in channels with small error-rate (to cope with the unreliable nature of UDP), we can create a unidirectional or a bidirectional covert channel using retransmissions with PRBP=1, in this way:

- sending a given message only once to be binary 1, and
- sending a given message two or more times to be binary 0.

In this way, one bit per message can be sent. According to the pattern-based classification [18], this channel belongs to the following class:

```
Network Covert Timing Channels
 --Re-Transmission Pattern
```

V. PERFORMANCE EVALUATION

Suppose that two IoT devices communicate with CoAP every *t* seconds.

TABLE I. PERFORMANCE EVALUATION OF THE NEW COVERT CHANNELS FOR SENDING THE MESSAGE "HELLO, WORLD!"

No.	Type of CC	PRBR	Time (*s*)		
			t*=1*s	***t*=5*s***	***t*=10*s***
1	CC using token and/or message ID Fields	16	6	30	60
		64	2	10	20
		80	2	10	20
2	CC using piggybacked and separate response	1	91	455	910
3	CC using payload of the message	280	1	1	1
4	CC using case-insensitive parts of the URIs	≤2040	1	1	1
5	CC using PUT and DELETE Methods	1	91	455	910
6	CC using Accept option	1	91	455	910
7	CC using conditional requests	1	91	455	910
8	CC using re-transmissions	1	91	455	910

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-661-3

Any covert channel with a given PRBR will need at least

$$\text{ceil}(l / \text{PRBR}) \cdot t\,(s)$$

for sending a message with length l bits.

We can evaluate the minimum time for sending the message ”Hello, world!” using the newly suggested covert channels. The message has length of 13 7-bit ASCII characters or l=91 bits. Results are given in Table 1.

So, we can see that not all suggested covert channels in CoAP are able to send short messages in real time, especially the ones with PRBR=1. Still, the covert channels 3 and 4 can be used for sending a short message per one CoAP message, without raising any suspicions. If the time for sending the message is not so important, one can choose covert channels 1 or 2, without raising any suspicions.

Additionally, we can evaluate the minimum time for sending the 320x240 raw color image (with 24-bit pixels) using the newly suggested covert channels. The size of the image is 225KB or l=1843200 bits. Results are given in Table 2.

TABLE II. PERFORMANCE EVALUATION OF THE NEW COVERT CHANNELS WITH PRBR>1 FOR SENDING 320X240 RAW COLOR IMAGE (WITH 24-BIT PIXELS)

	Type of CC	**PRBR**	**Time(*s*)**	
			t*=1*s	***t*=5*s***
1	CC using token and/or message ID Fields	16	115200 (32h)	576000 (160h)
		64	28800 (8h)	144000 (40h)
		80	23040 (6,4h)	115200 (32h)
2	CC using payload of the message	280	6583 (>1,82h)	32915 (>9.1h)
3	CC using case-insensitive parts of the URIs	≤2040	904 (15 min)	4520 (76 min)

The results from Table 2 show that most of the new CoAP covert channels are not quite suitable for sending images, because of the large transmission time. The covert channel 3 is the most suitable for that purpose (it will send 225KB image in 15 minutes).

VI. CONCLUSION

New CoAP covert channels are suitable for sending short messages. CoAP is the first specialized IoT protocol for which network steganographic techniques are applied. Considering that IoT will consist of about 30 billion objects by 2020 [14], CoAP belongs to the group of most exploited protocols in the forthcoming years, and its traffic will not raise any suspicions. So, it is important to identify possible ways of hiding data in it and trying to mitigate them. This paper deals with the first part, leaving others to try to find a solution for mitigating presented covert channels. One solution is the deployment of active and passive wardens.

The next step is implementation and demonstration of some of these covert channels, to present their functionality and feasibility.

REFERENCES

[1] Constrained RESTful Environments (CoRE) Parameters, CoAP Codes [Online]. Available at: https://www.iana.org/assignments/core-parameters/core-parameters.xhtml [retrieved: July, 2018]

[2] K. Denney, A. S. Uluagac, K. Akkaya, and S. Bhansali, “A novel storage covert channel on wearable devices using status bar notifications,” Proc. 13th IEEE Annual Consumer Communications & Networking Conference, *CCNC 2016*, Las Vegas, NV, USA, 2016, pp. 845-848, doi: 10.1109/CCNC.2016.7444898.

[3] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger, “Infranet: Circumventing Web Censorship and Surveillance,” Proc. 11th USENIX Security Symposium, San Francisco, CA, 2002, pp. 247-262.

[4] S. Fincher et al., “Perspectives on HCI patterns: concepts and tools,” Proc. Extended Abstracts on Human Factors in Computing Systems (CHI EA ’03). ACM, New York, NY, USA, 2003, pp. 1044–1045, doi: 10.1145/765891.766140.

[5] D. V. Forte, “SecSyslog: An Approach to Secure Logging Based on Covert Channels,” Proc. First International Workshop of Systematic Approaches to Digital Forensic Engineering (SADFE 2005), Taipei, Taiwan, 2005, pp. 248-263, doi: 10.1109/SADFE.2005.21.

[6] A. Houmansadr, N. Kiyavash, and N. Borisov., “RAINBOW: A Robust And Invisible Non-Blind Watermark forNetwork Flows,” Proc. 16th Network and Distributed System Security Symposium (NDSS 2009), San Diego, USA, The Internet Society, 2009.

[7] M. N. Islam, V. C. Patil, and S. Kundu, “Determining proximal geolocation of IoT edge devices via covert channel,” Proc. 18th International Symposium on Quality Electronic Design, ISQED 2017, Santa Clara, CA, USA, 2017, pp. 196-202, doi: 10.1109/ISQED.2017.7918316.

[8] B. W. Lampson, “Note on the Confinement Problem,” Commun. ACM vol. 16, 10, Oct. 1973, pp. 613-615, doi: 10.1145/362375.362389.

[9] D. Martins and H. Guyennet, “Attacks with Steganography in PHY and MAC Layers of 802.15.4 Protocol,” Proc. Fifth International Conference on Systems and Networks Communications (ICSCN), Nice, France, 2010, pp. 31-36, doi: 10.1109/ICSNC.2010.11.

[10] W. Mazurczyk and Z. Kotulski, “New Security and Control Protocol for VoIP Based on Steganography and Digital Watermarking,” Annales UMCS Informatica AI 5, 2006, pp. 417-426, doi: 10.17951/ai.2006.5.1.417-426.

[11] W. Mazurczyk and K. Szczypiorski, “Steganography of VoIP Streams,” in On the Move to Meaningful Internet Systems (OTM 2008) Robert Meersman, Zahir Tari (Eds.). LNCS, vol. 5332, 2008, pp. 1001-1018, doi: 10.1007/978-3-540-88873-4_6.

[12] A. Mileva and B. Panajotov, “Covert channels in TCP/IP protocol stack - extended version-,“ Central European Journal

of Computer Science vol. 4, 2, 2014, pp. 45-66, doi: 10.2478/s13537-014-0205-6.

[13] A. K. Nain and P. Rajalakshmi, "A Reliable Covert Channel over IEEE 802.15.4 using Steganography," Proc. IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, USA, 2016, pp. 711-716, doi: 10.1109/WF-IoT.2016.7845486.

[14] A. Nordrum, "Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated," IEEE Spectrum. 18 August, 2016.

[15] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, 2014.

[16] X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter packet delays," Proc. 10th ACM Conference on Computer and Communications Security (CCS'03), 2003, pp. 20-29, doi: 10.1145/948109.948115.

[17] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer VoIP calls on the Internet," Proc. 12th ACM Conference on Computer and Communications Security (CCS'05), Alexandria, VA, USA, 2005, pp. 81-91, doi: 10.1145/1102120.1102133.

[18] S. Wendzel, S. Zander, B. Fechner, and C. Herdin, "Pattern-Based Survey and Categorization of Network Covert Channel Techniques," ACM Computing Surveys vol. 47, 3, Article 50, 2015, doi: 10.1145/2684195.

[19] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," IEEE Communications Surveys and Tutorials vol. 9, 3, 2007, pp. 44-57, 10.1109/COMST.2007.4317620.

Copyright (c) IARIA, 2018. ISBN: 978-1-61208-661-3