Evaluating the Possibility to Perpetrate Tunneling Attacks Exploiting Short-Message-Service

Sara Narteni*, Ivan Vaccari, Maurizio Mongelli, Maurizio Aiello, and Enrico Cambiaso
Institute of Electronics, Information Engineering and Telecommunications,
National Research Council of Italy (CNR-IEIIT), Via De Marini 6, 16149, Genoa, Italy

Received: June 1, 2021; Accepted: July 26, 2021; Published: August 31, 2021

Abstract

In the cyber-security context, tunneling systems are exploited to bypass network restrictions to communicate outside of the targeted perimeter, without being detected. Such attacks represent a serious threat for the victim network, as they exploit legitimate protocols, encapsulating malicious payloads. In this paper, we design a tunneling architecture based on Short-Message-Service (SMS) and evaluate the possibility to adopt such communication medium for tunneling purposes. In order to evaluate the feasibility to set up an efficient SMS tunneling system, we perform some simulations, by varying both the payload size (from 10 Bytes to 1 MegaByte) and the SMS sending rate (up to 60 SMSs per minute). Results allow us to model the performance of a tunneling system, in terms of sending time. We derive indeed the underlying reference model through a mathematical analysis on the collected data. Results show that overall performance increases for an SMS sending rate greater or equal to 10 SMSs per minute, regardless of the message size.

Keywords: cyber-security, covert channel, data exfiltration, cyber-attacks, sms

1 Introduction

Mobile technologies are undergoing a huge expansion today and the related market is expected to constantly grow in the next years[1]. Mobile devices play an essential role in people's daily life, with many different mobile services available at low cost.

Many security issues emerged over the years, due to such technological advancements [23, 5]: as an example, a well-known phone company has recently declared that private and technical data have been stolen from users' SIM cards[2]. This is just an example of data exfiltration [33], which consists in stealing sensitive information through various forms of cyber-attacks. Considering data exfiltration attacks, covert channels establish secret communications in a hidden and malicious way [19].

In this context, Lampson [16] first introduced and defined covert channels in 1973, as communication channels *"not intended for information transfer at all"*. Later, Simmons made use of the well-known Prisoner Problem to describe the logic behind a covert channel [29]: prisoners Alice and Bob want to communicate to agree on an escape plan, but there's a warden, Wendy, who monitors all the messages. If Wendy discovers suspicious messages, she will place Alice and Bob into solitary confinement, so their escape plan will vanish. To bypass Wendy's control, Alice and Bob must exchange innocuous messages

*Corresponding author: Institute of Electronics, Information Engineering and Telecommunications, National Research Council of Italy (CNR-IEIIT), Via De Marini 6, 16149, Genoa, Italy, sara.narteni@ieiit.cnr.it, +39 010 6475 217

[1]More information is available at the following address: `https://www.grandviewresearch.com/industry-analysis/mobile-marketing-market`

[2]More information is available at the following address: `https://www.cybersecurity-help.cz/blog/1855.html`

(*overt channel*) containing hidden information (*covert channel*) [35]. Alice and Bob can be two nodes in a network that communicate exploiting different protocols.

There are many ways to classify covert channels: for example, they can be categorized according to the distinct Open System Interconnection (OSI) layers [9] or on the basis of the mechanism over which they're constructed. Secret data can be encapsulated into header fields of network and transport layers protocols (i.e. IP, TCP, ICMP), either by header bit modulation (e.g. unused header bits, TCP ISN field, checksum field, address fields, etc.), header bit crafting (e.g. header extension, padding, IP ID and Fragment Offset) or optional header extension [10].

An alternative technique, also adopted in this manuscript, consists in payload tunneling [36], that means embedding a forbidden protocol into the payload of an allowed one, such as IP over ICMP, SSH over HTTP, UDP/TCP over HTTP. Moreover, it is possible to employ headers of the application layer protocols, such as HTTP or DNS [36]. Covert channels described above are known as covert *storage* channels (CSC). In contrast, covert *timing* channels (CTC) are based on concealing information by modulating packet timing parameters [31]. Recent studies proposed covert channels in the field of mobile vocal calls belonging to CTC category, exploiting VoLTE (VoIP over LTE) Protocol [37] and VoIP [27]. In both works, covert data is hidden into silent periods of the voice signal.

Therefore, covert channels are a broad category of data exfiltration attacks, that may be perpetrated in many different ways: tunneling systems can be considered as a sub-category of covert channels, where the attacks involve the usage of network protocols payload. In this paper, we study the feasibility of a novel tunneling system based on CSC, in which application data are carried inside of Short-Message-Service (SMS) payloads. From the viewpoint of the proposed system, mobile devices become instruments used by the attacker. The current work represents an extension of [22], introducing performance optimization and discussing possible behaviors in function of the maximum number of SMS the network is able to send within a given amount of time.

Although covert channels affecting mobile networks can already be found in literature (see, e.g., [37, 28]), to the best of our knowledge, SMS messages are not exploited, while they can represent a new type of covert channel. Nevertheless, setting up a tunneling system based on SMSs is not trivial, due to the characteristics of the transmission medium (the SMSs) itself. A typical scenario where the proposed system could fit is represented by an insider threat [12], where the insider (i.e. a member of an organization who wants to secretly communicate outside) could make misuse of mobile devices to bypass his/her organization network by exploiting the proposed SMS tunneling scheme, thus exposing the organization to serious damages, such as the leak of sensitive information and data.

The remaining of the paper is structured as follows: Section 2 reports the related works on the topic, while Section 3 describes the concept idea of the proposed innovative covert channel. Instead, Section 4 reports our simulation-based testing and the subsequent mathematical analysis. Finally, Section 5 concludes the paper and reports further work on the topic.

2 Related Work

In the era of mobile technologies outbreak, users tend to make a massive use of their mobile devices, often without being aware of the risks they may encounter. This has paved the way to many works on cyber-security in the mobile field.

A lot of research has been done recently ([30], [21],[15], [26], [7]) to survey different vulnerabilities, threats and attacks affecting mobile devices and their Operating Systems. Some attacks address the victim device itself (such as viruses, spyware, Trojan, rootkit etc), while others eavesdrop the communication between different mobiles for information leakage (as for the case of Man-In-The-Middle attack) [30]. In the SMS context, some attacks exploit SMSs for data exfiltration, e.g. Smishing (phishing via

SMS) attack [21], that consists in stealing information by sending apparently innocuous SMSs to the users.

In another work ([34]), SMS interception and manipulation attacks are presented, where attackers disguise as SMSC and make use of fake MSC to identify the victim, with the aim of receiving, storing, modifying all its SMSs and then forward these messages to the recipient.

SMSs are now widely adopted for two-factors authentication, even if they're not so secure, as demonstrated in [13]. In this work, three methods to intercept SMSs are presented: GSM traffic capture; getting access to Signalling System No. 7 (SS7) architecture and exploiting its protocols (like CAMEL) flaws; SIM swapping, which consists in porting a user's SIM card by fooling mobile operators, so that the attacker receives all its data.

Regarding the Global System for Mobile Communication (GSM) network, a second generation standard for cellular networks that reached 90% of market share by 2017 [11], only a few works explored the possible exploitation of SMS protocols to achieve covert channels. The author in [25] developed a system to embed secret information inside SMS User Data Header redundant fields, in such a way that an independent warden (recalling Simmons's Prisoner Problem [29]) can mistake the SMS as normal.

A thesis work illustrated how to secretly exchange data through SMSs [14]: the aim of this study was to develop a system able of transmitting SMS messages from an Android device without the end user acknowledgement. This was achieved by using an Android application able to covertly transmit SMSs and a network node able to receive them; in particular, OpenBTS and SMSqueue softwares were used to establish a GSM network and manage SMS routing, acting like a SMSC.

Besides the SMS-based approaches mentioned so far, secret information can be hidden directly in text messages (or other media): this method is known as steganography. Different works have introduced different methodologies to perform steganography: as an example, secret data may be hidden in text manipulations (like abbreviations, blank spaces, synonyms and acronyms [18]) or into the widely used emoticons [24]. However, steganographic methods are a sub-field of covert channels, being based on modifications of the contents of the data being transmitted and, unlike tunneling systems, do not necessarily involve network protocols.

The idea of a tunneling system through SMSs was investigated in [8], where a so-called WebSIM was introduced, being a SIM card that acts as a web server for personal security. In this system, IP packets were embedded into SMSs in order to reach the WebSIM and provide connectivity. In contrast, in our system we use SMSs just as carriers for bringing data to a remote destination.

Another work [17] developed a query/retrieve system, called iTrust with SMS, which allowed users to search/get information cointained in the iTrust with HTTP network nodes[6] via SMS. The requests are made through text keywords and then this framework is able to convert them to GET HTTP requests. However, such a system has limited application because of the limitation in SMS size (140 bytes) and it cannot fit into a covert channel definition.

To the best of our knowledge, no study investigated the idea of modelling the SMS behavior for tunneling purposes (see Table 1 for a comparison with the reported literature), intended as a way to exchange forbidden data between two nodes via SMS messages.

3 The Proposed SMS tunneling system

A tunneling system is a well-established technique in network security, consisting in disguising data of a forbidden protocol into an allowed one: it is often used to bypass network limitations (i.e. network firewalls) and bring some private and sensitive data to an external network, without being recognized by the targeted network. A practical application can be represented by the insider threat scenario [12].

References	**Main topic considered**	**Strengths**	**Weaknesses**
[30],[21],[15], [26],[7]	Surveys on mobile security	Many cyber attacks against devices and networks are presented; some attacks about data exfiltration (MiTM) and based on SMSs (Smishing)	No mention on tunneling attacks through any protocol
[34]	SMS interception and manipulation	The presented attacks aim at information stealing through SMS architecture	The attacks differ from tunneling systems as they act at SMSC level
[13]	Two-factor authentication (2FA) via SMS	Data exfiltration based on SMS	The attack is perpetrated exploiting 2FA
[25]	Covert channel through SMS	Presents a way to encapsulate secret data into SMSs	The attack is based on SMS header protocols instead of payload
[14]	Secret transmission of SMSs	Presents a principle of hidden communications without user acknowledgement	The attack is not achieved through a tunneling architecture
[18], [24]	SMS-based steganography	Data hiding into SMS content	The attack is not based on network protocols, but exploits direct manipulations of SMS content
[8]	WebSIM	Embedding of IP packets into SMSs	The system aims at reaching the SIM card only, not remote destinations
[17], [6]	iTrustSMS	Usage of SMS to get information from web	It is not intended as a cyber-attack

Table 1: Strengths and weaknesses of existing literature compared to our proposed work.

Inspired by the fast development of mobile technologies and the sensitive cost reduction of SMSs, we studied if it is possible to implement a tunneling system built upon the SMS protocol.

Our SMS tunneling system can be described through the architecture scheme shown in Figure 1, where the red dashed arrow identifies the communication medium exploited by the tunnel (SMS, in our case). Our proposed system is made up of four software modules: socks, tunnel client, tunnel server,

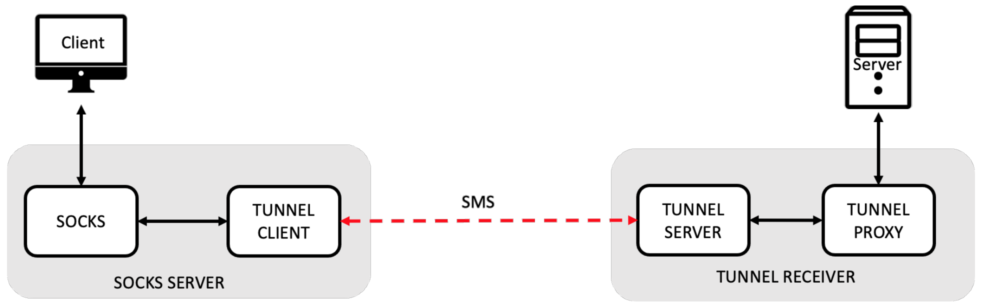


Figure 1: SMS Tunneling architecture

and tunnel proxy. In order to better describe the functionalities of our system, the first two blocks may be considered as a unique socks server and, similarly, the other two blocks can be referred to as a single tunnel receiver component. In our scenario, an attacker uses a client (e.g. a web browser) in communication with the socks server to establish a connection over the tunnel. We consider the socks server being located inside of a targeted private network and used by the malicious node to initiate the tunneling system. In contrast, the tunnel receiver component is thought as an external node, which is under the control of the attacker too. Therefore, by exploiting such a tunneling system, the client may be able to exchange messages with a remote server.

Suppose $C \to S$ and $S \to C$ being the request and response pathways respectively, involving both the tunnel client C and the tunnel server S modules. We now provide a more detailed description of the single modules:

- **Socks server**. In the $C \to S$ phase, it receives the messages from the client, splits their payload into portions of L characters and crafts an SMS string for each obtained fragment. Such SMS is then sent to the tunnel receiver. In $S \rightarrow C$, it listens for incoming connections from tunnel receiver and extracts the remote server payload part from each received SMS. Finally, such response reaches the client.

- **Tunnel receiver**. This component works in a symmetric manner with respect to the socks server. It overtly communicates with a remote server: while being in $C \rightarrow S$ phase, the tunnel receiver extracts the original request fragment from each SMS received by the socks server and forwards it to the remote server; in $S \rightarrow C$, it receives a server response, then it generates SMSs in the same way as socks server does for $C \rightarrow S$ and sends them to the socks server.

According to [22], the structure of each SMS, for both $C \rightarrow S$ and $S \rightarrow C$ phases, contains the following fields, separated, e.g., by a dash character: (i) the remote server IP address (`destination_ip` in the following), (ii) the remote server listening port number (`destination_port` in the following), (iii) the client port number, (iv) a message index (`index` in the following), and (v) the payload (portion) to send (`content` in the following). The first three fields function as a header used to match the connection between the client and the socks[3]. The tunnel proxy uses `destination_ip` and `destination_port` to establish the connection with the remote server. The `index` is adopted as a counter for the current fragment of the original message. Such information is useful to resemble the original message, once SMSs are received by the tunnel receiver.

The `content` field embeds the base64-encoded request/response fragment. Such encoding allows us to maintain the standard ASCII SMS characters format, thus avoiding extensions to the representation of a message, that would lead to a reduced amount of data carried out by a single SMS.

In order to handle the connection between the client and the socks server at the transport level, so that the system acknowledges when the data exchange between them must be interrupted, we designed and introduced a particular SMS, which we refer to as *zero packet*. Such SMS has the same header as application level SMSs (as described above), while its `index` is fixed to `0`. The `content` is set to either `1`, if the connection is open, or to `0`, when the connection is closed. Hence, when the communication between client and socks server ends, a closure zero packet with `content=0` is used to propagate the closure through the tunnel. Similarly, a zero packet with `content=1` may be sent to notify an established connection. In our case, we do not send such packet, as the connection with the external server is established at receiving of the first message, in order to avoid the expiration of TCP timeouts, waiting for data after a three-way-handshake is completed.

According to [22], the maximum size available for the `content` field is limited due to the presence of the other fields, that may reach a considerable length of up to 30 characters approximately. Moreover, we must take into account that the base 64 encoding we adopt for the `content` increases the original length too. In other words, each SMS in the communication will contain more than 30 extra characters besides the `content`. This may negatively impact on the performance of the tunnel, in terms of how much bandwidth it is possible to transfer over it. As a solution to such a limitation, in this work, we propose an optimization of the previous SMSs structure [22], in order to avoid the usage of the header inside each SMS message to be sent. Such improvement is based on the introduction of an `ID` that uniquely identifies the communication and replaces the long header. When a connection starts, the corresponding opening zero packet will be sent from the tunnel client to the tunnel server, that will respond back by sending another special zero packet with `index=2` and an `ID` used as `content`. Such `ID` is a number associated to the header (i.e. `destination_ip-destination_port-client_port`) that uniquely identifies it. Then, the tunnel client extracts the `ID` from the zero packet and crafts the other SMSs by inserting the

[3]Here, we assume a single client is present; in case of multiple clients, an additional field may be required, specifying the IP address of the involved client.

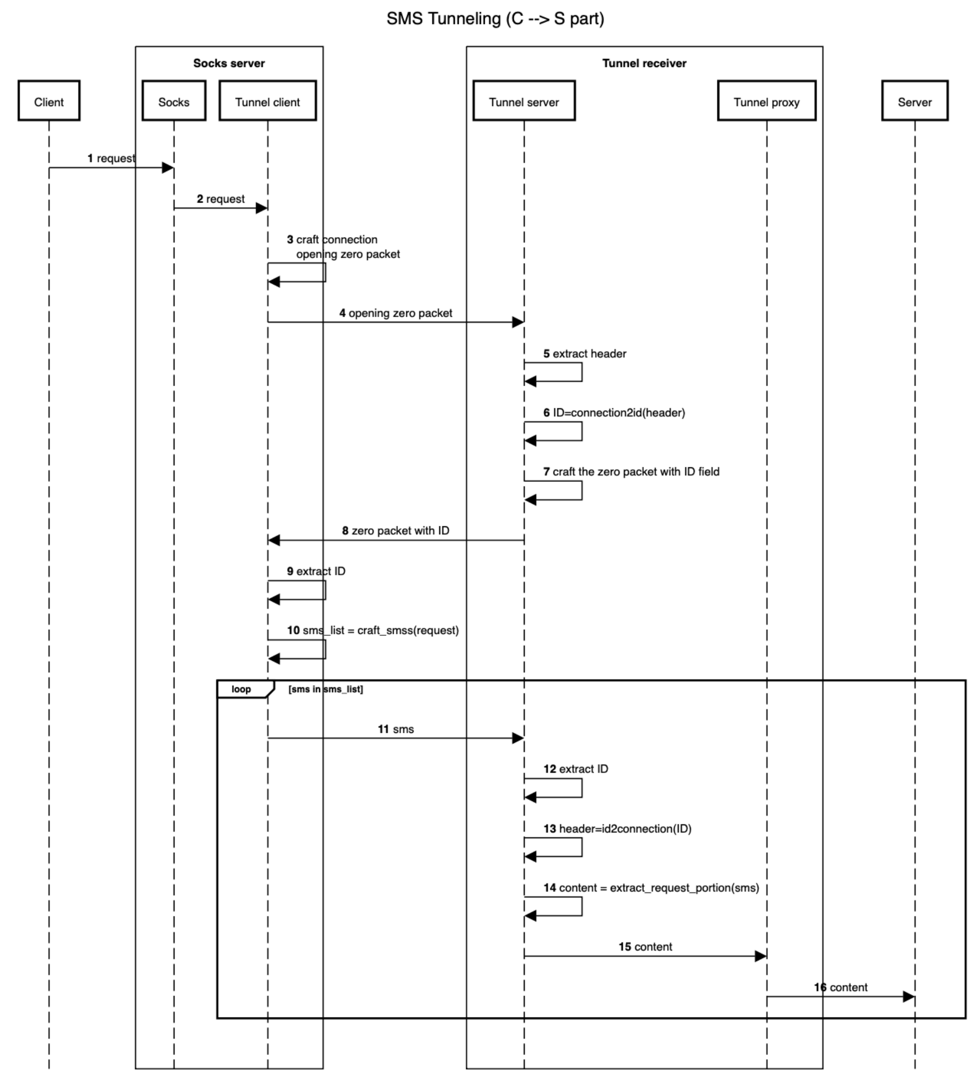


Figure 2: Sequence diagram of the request phase from Client to Server

`ID` in place of the header. Eventually, the optimized SMS structure during the actual communication is as follows: `ID-index-content`, with `index` greater than `0`. With such modifications, we expect to achieve a better performance, as the `content` can have a greater size, almost the entire length of a single SMS message.

For a deeper description of how our tunneling architecture works, we report the sequence diagrams related to $C \rightarrow S$ and $S \rightarrow C$ in Figures 2 and 3 respectively.

Focusing on the diagram in Figure 2, the following steps describe the communication occurring to send a message (a request) from Client to Server: (i) the *Client* indirectly sends a *request* message to the *Server*; (ii) the *Socks* forwards the *request* to the *Tunnel Client*; (iii) the *Tunnel Client* generates a *zero packet* for connection opening; (iv) the opening *zero packet* is sent to *Tunnel Server*; (v) the *Tunnel Server* extracts the header (`destination_ip-destination_port-client_port`); (vi) the *ID* is generated from the header; (vii) the SMS containing the *ID* is generated; (viii) the SMS containing the *ID* is sent back to *Tunnel Client*; (ix) the *ID* is extracted from the received SMS; (x) the SMSs with portions of *request* are created; (xi) the SMS is sent from *Tunnel Client* to *Tunnel Server*; (xii) the *ID* is extracted from the received SMS; (xiii) the *ID* is used to retrieve the header; (xiv) the received SMS is

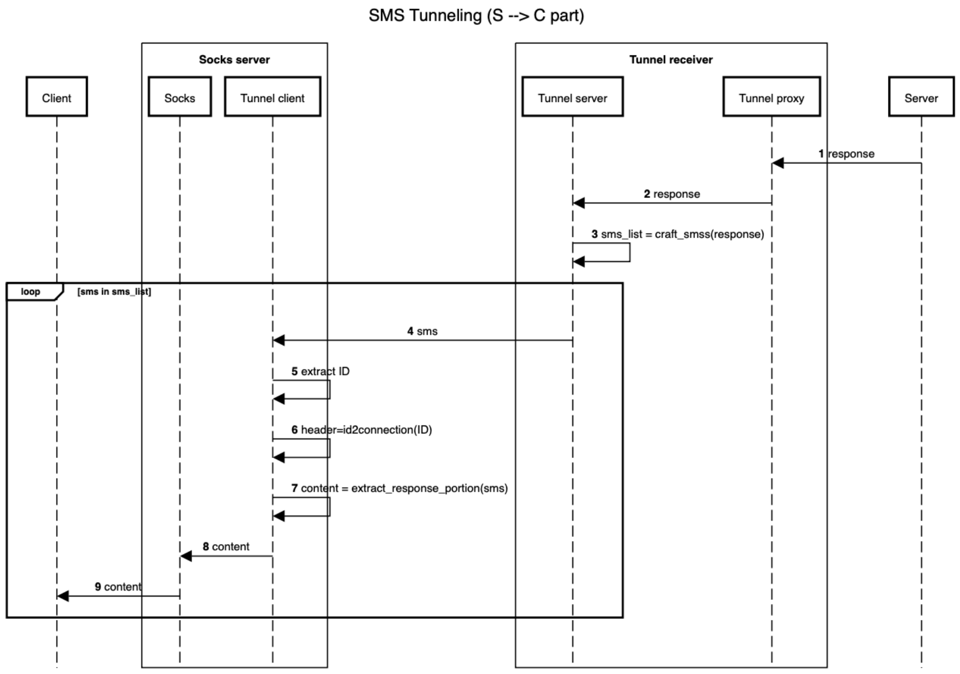


Figure 3: Sequence diagram of the response phase from Server to Client

parsed to get the `content` (portion of *request*); (xv) the `content` is sent to the *Tunnel Proxy*; (xvi) the *Tunnel Proxy* forwards the *request* portions to the *Server*. In the response phase, the communication is at some extent symmetric to the request phase: referring to Figure 3, it is as follows: (i) the *Server* indirectly sends a *response* message to the *Client*; (ii) the *Tunnel Proxy* forwards the *response* to the *Tunnel Server*; (iii) the SMSs with *response* portions are created, with the *ID* replacing the header; (iv) each SMS is sent from *Tunnel Server* to *Tunnel Client*; (v) the *ID* is extracted from each SMS; (vi) the header is generated from the *ID*; (vii) the `content` with payload is extracted from the received SMS; (viii) the payload is sent from *Tunnel Client* to *Socks*; (ix) the *Socks* forwards the *response* portions to the *Client*.

4 Tests and obtained results

For the evaluation of the proposed tunneling system, we decided to investigate its performance in two different directions. Since the correct functionality of SMS sending through our architecture is a fundamental requirement, in a previous work, we firstly conducted a preliminary activity aimed to identify the potential limits of SMSs, when used for tunneling purposes [22].In the present work, that represents an extension to the work presented in [22], we first executed accurate testing aimed at assessing the SMS behaviour, hence acquired benchmark performance data useful to analyze the performance of the proposed tunnel: we achieved it by simulating the communication via SMS using TCP sockets instead.

4.1 Tests on simulated SMSs

In order to exploit the versatility of the proposed tunneling architecture, able to encapsulate any kind of TCP-based application layer protocol, we designed a simulation system for SMS sending and receiving. In this case, TCP sockets are used in place of real SMSs.

In particular, according to the initial model proposed in [22], a message m of size $|m|$ bytes was split into smaller parts that did not exceed the maximum size of SMSs payload (160 characters) and simulated

SMS strings were crafted as explained in Section 3. The aim of our simulation was to study how the total amount of time, T_m, required for a complete communication (i.e. sending a request and receiving back a response) varied with the frequency of simulated SMSs, measured as the maximum number $N^{[60]} = N \cdot 60$ of messages exchanged between the tunnel client and tunnel server in one minute[4] (for simplicity, we will refer to it as N in the following).

Also, we considered messages of different sizes $|m|$ in order to understand what is their impact on such amount of time. In particular, we considered all the integer rates $N \in [2,60]$ and $|m| = \{10B, 100B, 1kB, 10kB, 100kB, 1MB\}$. In our scenario, the client sends the message m, which is an uppercase characters string of size $|m|$, to a remote server by using the tunneling architecture. The remote server then echoes the message back after converting it to lowercase. Hence, for our scenario, we always have that the length of the message encapsulated and sent during the $C \rightarrow S$ pathway is always equal to the length of the message/response encapsulated and sent during the $S \rightarrow C$ pathway. To compute the amount of time T_m, in seconds, we captured all the traffic involving the tunnel components from the beginning to the end of each process, until the connection between the client and the socks server closed, i.e. until the `FIN+ACK` packet was sent towards the socks server. For each process, involving a fixed N and a fixed $|m|$ in the considered values ranges, the capture of the network traffic generated by the client and the server resulted in P network packets exchanged, with $P > k_m$, for k_m defined in [22] as the number of SMSs needed to send message m. Let's suppose $T_i, i \in [0, P-1]$ being the time of the i-th captured packet. T_m is then computed as follows:

$$T_m = T_{P-1} - T_0 \tag{1}$$

The obtained T_m values at different fixed payload size $|m|$ are plotted in Figure 4, by varying $N \in [2,60]$.

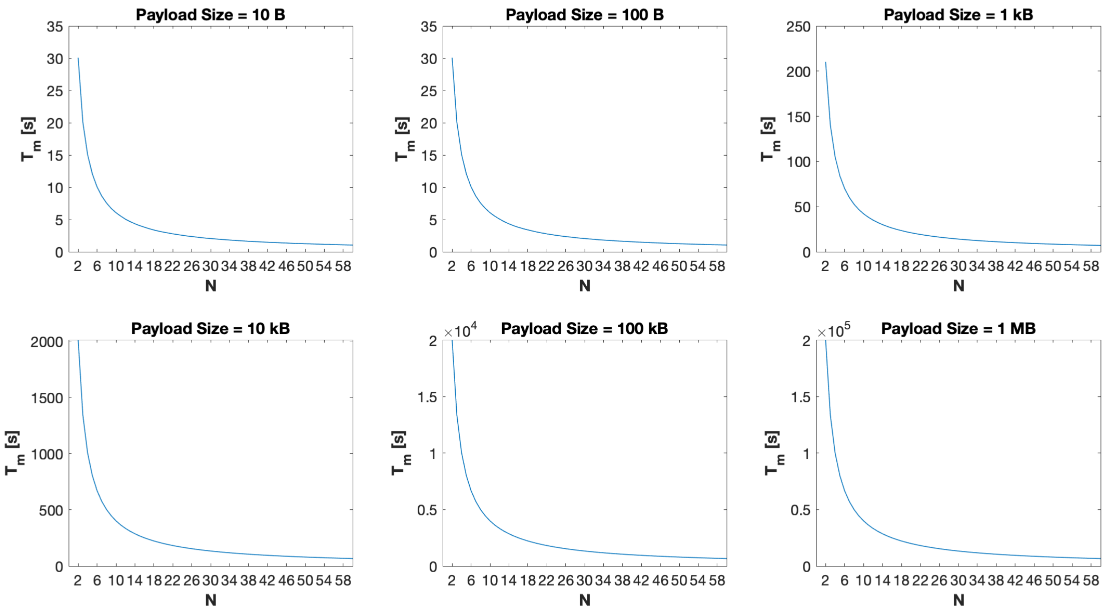


Figure 4: Plot of the obtained T_m from the traffic capture when different numbers of SMSs per minute were used, for different payload sizes $|m|$ of the message

[4]According to [22], N is equal to the number of SMSs sent in one second.

As expected, we can observe that, for every payload size, the amount of time T_m decreases with the increase of N. Based on such results, we introduce a mathematical model of this behaviour in the following sections.

4.1.1 Curve fitting on the obtained results

Starting from the obtained results of our simulation, we wanted to look for a function $t(n)$ able to approximate the time required for communication at varying of the simulated SMSs per minute n , defined for $n \in \mathbb{R}^+$, for a given payload $|m|$, based on the collected measurements of T_m in our tests.

When dealing with curve fitting tasks, the first step is the choice of a proper model to be fit to the measured data. In our case, the shape of the real curves (Figure 4) indicated a sharp decrease towards a horizontal asymptote for $T_m = 0$, so we chose an exponential model. After a fitting performance comparison between one-term and two-terms exponential models, we opted for the latter.

For a fixed payload size $|m|$, such model can be expressed by the following function:

$$t(n) = a \cdot e^{b \cdot n} + c \cdot e^{d \cdot n} \tag{2}$$

where a, b, c, d are parameters to be found in the fitting algorithm, which is based on trust-region approach and its variations [20, 3, 2]. The goodness-of-fit is evaluated by the coefficient of determination R^2 [4]: it is the ratio of the explained sum of squares (ESS) and the total sum of squares (TSS) and is widely used in this kind of evaluations. Its value varies in $[0,1]$, with $R^2 = 1$ being the ideal fitting performance.

In our case, the obtained fitting coefficients and the related R^2 for different payloads are reported in Table 2.

$\|m\|$	a	b	c	d	R^2
$10B$	54.7817	-0.4439	7.4387	-0.0397	0.9961
$100B$	54.7093	-0.4433	7.4361	-0.0396	0.9961
$1kB$	383.8990	-0.4459	52.3744	-0.0406	0.9962
$10kB$	3678.1241	-0.4465	502.0449	-0.0408	0.9962
$100kB$	36631.3932	-0.4468	5003.0483	-0.0408	0.9962
$1MB$	366230.9041	-0.4469	$50034,5807$	$-0,04093$	$0,9962$

Table 2: Results of fitting with two-terms exponential for different payload sizes

Such results indicate very good performances (being $R^2 > 0.99$ for every payload) and let us observe that coefficients b and d do not significantly change with the increased payload size, whereas a and c increase with it.

Such behaviour is in line with the curves obtained from real data (Figure 4), where the slope is similar regardless of the payload, but found values of T_m increase with the increase of the payload. This has suggested us to look for a more general function that expresses the time T_m not only based on the number of SMSs per minute, but also on the payload size. To achieve this, we have defined two other fitting functions for coefficients a and c at varying of payload size. Suppose s being the general payload size of a message m, we look for the following functions:

$$a(s) = p_1^a \cdot s + p_2^a \tag{3}$$

$$c(s) = p_1^c \cdot s + p_2^c \tag{4}$$

with p_i^x being the i-th coefficient of the linear fitting function, for the parameter x of the two-terms exponential, as defined in Equation 2.

In this case, both models are chosen as first degree polynomials as we observed a linear trend of a and c values, as obtained from Table 2, with respect to the payload size. The fitting algorithm in this case relies on QR factorization [1]: at the end, we got the coefficients and R^2, as shown in Table 3.

Function	p_1	p_2	R^2
$a(s)$	0.3662	22.9986	0.9999
$c(s)$	0.0500	2.7259	0.9999

Table 3: Results of the linear fitting conducted to relate coefficients a and c of Eq. 2 to the payload size s of the message conveyed through the tunnel

In light of the results of this further fitting procedure, we can rewrite $t(n)$ as a two-variable function, depending also on payload size s, as follows:

$$t(n,s) = (p_1^a \cdot s + p_2^a) \cdot e^{b \cdot n} + (p_1^c \cdot s + p_2^c) \cdot e^{d \cdot n} \tag{5}$$

In this expression, coefficients b and d are considered constant due to their minimal changes with payload size: we took the arithmetic mean of the values reported in Table 2, thus setting $b = -0.4456$ and $d = -0.0404$.

The function can be visualized as a 3D surface for a better understanding of the behaviour.

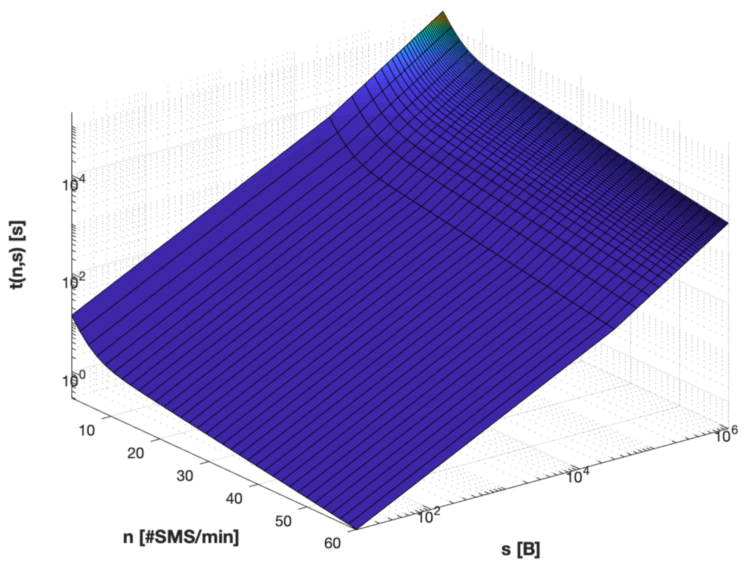


Figure 5: 3D surface plot of the function $t(n,s)$ at varying of number of SMSs per minute and payload size

The shape of the surface is consistent with what we expected from the tests results before fitting (Figure 4): with the increase of payload size, at low numbers of SMSs per minute (n), the required time for communications (expressed by $t(n,s)$) raises to higher values. However, when n raises up to 60, the required time sensitively lowers.

Moreover, the elapsed time T_m for a complete request/response path may be known in advance as it may be bounded to specific timeouts of the system, adopted, e.g., by components like a web browser, or even the kernel. As a consequence, the obtained function $t(n,s)$ can be inverted in order to get a function $s(t,n)$ to compute the maximum payload size which is possible to transfer over the tunnel at a given time $t = T_m$, in seconds, and by sending $n = N^{60}$ SMSs per minute.

4.1.2 Curve Fitting on Percentage Elapsed Time Variations

Besides the above considerations, we also studied the data collected from SMS simulations in order to find out a minimum threshold of SMS rate, $\tilde{N}$, that allows the message m to be sent in a proper time.

Looking at Figure 4, we noticed that the shape of the obtained curves is (almost) the same for all the payload sizes, while the scale on the elapsed time T_m, as expected, differs (higher T_m values for higher payload sizes). Hence, we decided to compute the percentage variations in elapsed time corresponding to each value of N and its previous, as follows:

$$\Delta T_m = \frac{T_{m_i} - T_{m_{i-1}}}{T_{m_i}} \forall i \in [3, 60] \tag{6}$$

where ΔT_m is expressed in percentage and T_{m_i} is the measured elapsed time for $N = i$.

As expected, results have shown the same variations for each payload size, thus reflecting the similar shape of the original curves. The trend of the obtained percentage variations is depicted in Figure 6.

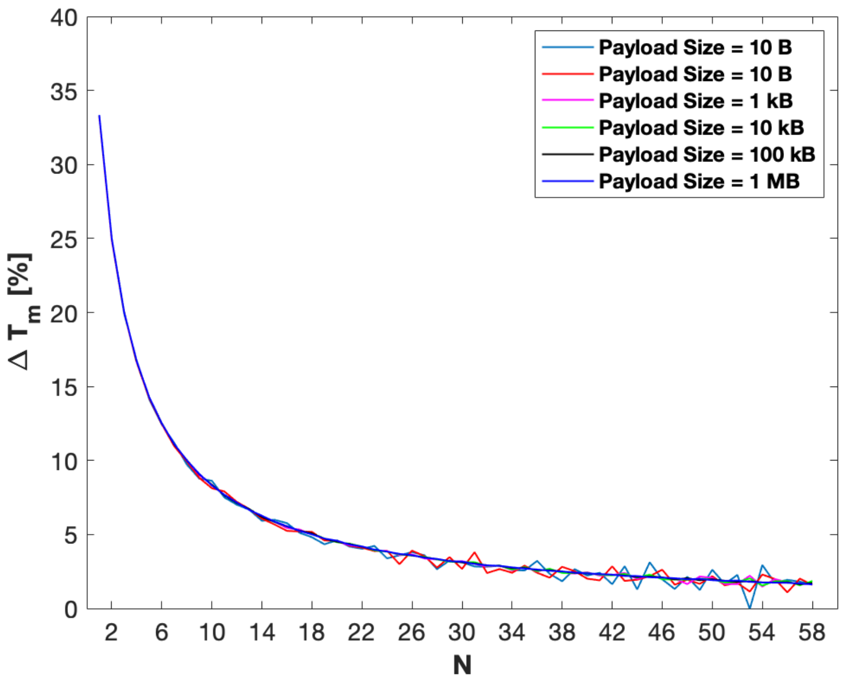


Figure 6: Percentage variations in elapsed time at varying of number of SMSs per minute, computed with the formula in Eq. 6

Observing the curves in Figure 6, it can be noted that, by increasing the payload size, the noise added to the curve decreases. This is due to the impact of the exploited protocol on the overall traffic, decreasing with the increase of the size of the encapsulated payload. Moreover, it is noticeable that the quantities on horizontal and vertical axis have now the same order of magnitude. Thus, we decided to define $\tilde{N}$ through the first derivative of a function that approximates the trend of ΔT_m. In fact, from a geometrical point of view, the first derivative of a function is equivalent to the angular coefficient of the tangent line in a given point of the function, which is also equivalent to the tangent of the angle α formed by such tangent line and the positive direction of the horizontal axis [32].

From the obtained curves, we decided to consider, as a proper threshold $\tilde{N}$, the value of N where the tangent line defines $\alpha = \frac{3}{4}\pi$. Hence, we have: $\tan\alpha = -1$.

To this purpose, the first step is to define a function $t_{\%}(n)$ that approximates each curve in Figure 6. Then, denoting $t'_{\%}(n)$ as its first derivative, the threshold value $n = \tilde{N}$ can be found by solving the following equation:

$$t'_{\%}(n) = -1 \tag{7}$$

In order to get the function $t_{\%}(n)$, we performed a fitting task. As the shape of the real curves (Figure 6) did not change with respect to the results in Figure 4, we adopted the same two-term exponential fitting model as in Equation 2, hence defined $t_{\%}(n)$ as follows:

$$t_{\%}(n) = a \cdot e^{b \cdot n} + c \cdot e^{d \cdot n} \tag{8}$$

In this case, the resulting coefficients and their related R^2 are reported in Table 4.

$\lvert m\rvert$	a	b	c	d	R^2
$10B$	62.5857	−0.3259	9.9578	−0.0346	0.9918
$100B$	62.7942	−0.3289	10.1224	−0.0353	0.9948
$1kB$	62.4797	−0.3261	10.0368	−0.0343	0.9971
$10kB$	62.4910	−0.3259	10.0225	−0.0342	0.9973
$100kB$	62.5787	−0.3265	10.0460	−0.0342	0.9975
$1MB$	62.6429	−0.3271	10.0777	−0.0343	0.9975

Table 4: Results of fitting on percentage variations with two-terms exponential for different payload sizes

By comparing the fitting coefficients in Table 2 and Table 4, it is possible to notice that in the latter all the coefficients have similar values with different payload sizes.

Now, we can consider the first derivative of $t_{\%}(n)$: given the expression in Equation 8, we can express $t'_{\%}(n)$ as:

$$t'_{\%}(n) = a \cdot b \cdot e^{b \cdot n} + c \cdot d \cdot e^{d \cdot n} \tag{9}$$

We then solved Equation 7 for each payload size (using the a, b, c, d values reported in Table 4) and obtained the solutions shown in Table 5.

$\lvert m\rvert$	$\tilde{N}$
$10B$	10.1062
$100B$	10.0807
$1kB$	10.0987
$10kB$	10.1003
$100kB$	10.0949
$1MB$	10.0898

Table 5: Values obtained by solving Eq. 7 for different payload sizes

Disregarding the decimal values from such results, we can finally conclude that if $N \geq \tilde{N} = 10$ SMSs per minute, our tunneling system will be effective regardless of the payload size.

For a visual understanding of the analysis carried on in this section of the paper, in Figure 7 we report a graphical explanation of our procedure of fitting on the percentage variations in elapsed time.

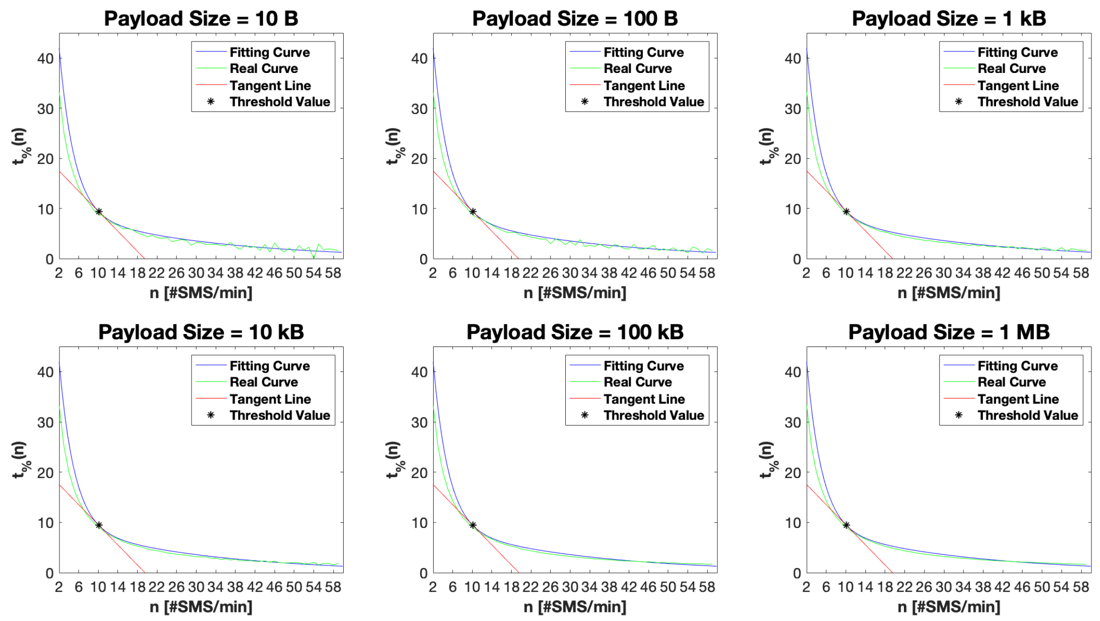


Figure 7: Visual summary of the obtained results: for each payload size, a fitting task was performed on real percentage variations in elapsed time (green), obtaining fitting curves (blue) with $R^2 > 0.99$ in every case. Then, the solution of Eq. 7 individuates the tangent line (red) to fitting curves in the point $n = \tilde{N}$, that is the sought threshold.

The resulting threshold value is then applicable also with real data, as reported in Figure 8, which is the same plot as Figure 4 with the obtained threshold point.

5 Conclusions and Future Works

In this paper, we have explored the opportunity of establishing a tunneling system encapsulating application layer payloads into SMS messages. In our previous study, we already assessed the feasibility of such kind of attack [22]. Based on it, in the present work, we refined and extended the structure of the SMSs to optimize the overall performance of the tunnel. Moreover, we designed and developed an experimental setting to derive a mathematical model for a simulated SMS-like communication that uses our tunneling architecture. These testing and analytic phases led us to individuate the minimum rate of 10 SMS messages per minute, that allows a potential malicious user to send a message through the tunnel in an adequate amount of time. At the same time, we have obtained general models that relate the timings with the size of the message conveyed through the tunnel.

Future studies in this direction will involve a further testing with real SMSs sent on real network scenarios, in order to validate the results of this work and identify the SMS sending rate on different real situations. Moreover, we plan to investigate more on the potentials and limitations of the SMS tunneling, also compared to specific communication media (e.g. emails sending, instant messaging, etc.).

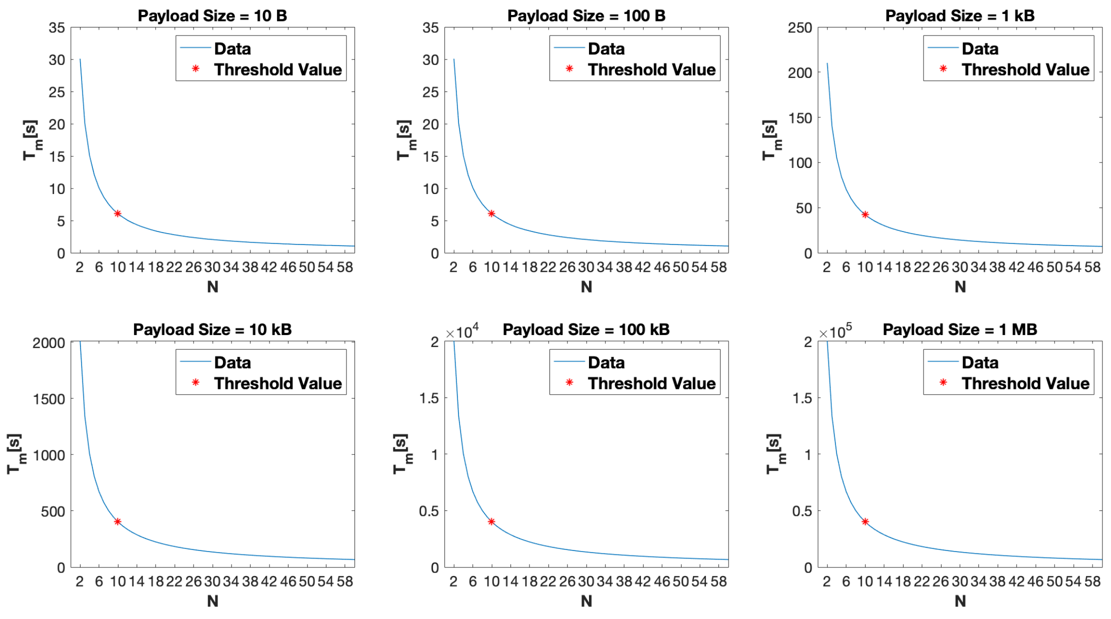


Figure 8: Plot of the obtained T_m from the traffic capture with highlighting of the threshold value for $\tilde{N} = 10$ SMSs per minute

References

[1] E. Anderson, Z. Bai, and J. Dongarra. Generalized qr factorization and its applications. *Linear Algebra and its Applications*, 162:243–271, February 1992.

[2] M. A. Branch, T. F. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

[3] R. H. Byrd, R. B. Schnabel, and G. A. Shultz. Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Mathematical programming*, 40(1):247–263, January 1988.

[4] A. C. Cameron and F. A. Windmeijer. An r-squared measure of goodness of fit for some common nonlinear regression models. *Journal of econometrics*, 77(2):329–342, April 1997.

[5] L. Caviglione, M. Gaggero, J.-F. Lalande, W. Mazurczyk, and M. Urbański. Seeing the unseen: Revealing mobile malware hidden communications via energy consumption and artificial intelligence. *IEEE Transactions on Information Forensics and Security*, 11(4):799–810, April 2015.

[6] Y.-T. Chuang, M. Isaí, L. Lombera, P. Moser, and Melliar-Smith. Trustworthy distributed search and retrieval over the internet. `https://itrust.ece.ucsb.edu/papers/ICOMP_FINAL.pdf` [Online; accessed on August 15, 2021], April 2012.

[7] S. Farhan, S. F. Zaidi, A. Munam, M. Shah, M. Kamran, Q. Javaid, and S. Zhang. A survey on security for smartphone device. *International Journal of Advanced Computer Science and Applications*, 7(4):206–219, April 2016.

[8] S. Guthery, J. Posegga, and M.-m. Deutsche. The websim - clever smartcards listen to port 80. `http://www.teco.edu/~posegga/papers/websim.pdf` [Online; accessed on August 15, 2021], January 2000.

[9] T. G. Handel and M. T. Sandford. Hiding data in the osi network model. In *Proc. of the 1st international workshop on Information Hiding (IH'96), Cambridge, UK*, volume 1174 of *Lecture Notes in Computer Science*, pages 23–38. Springer Berlin Heidelberg, May-June 1996.

[10] Y. Heda and R. Shah. Covert channel design and detection techniques : a survey. In *Proc. of the 3rd IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT'15), Bangalore, India*, pages 1–6. IEEE, July 2015.

[11] F. Hillebrand. The creation of standards for global mobile communication: Gsm and umts standardization from 1982 to 2000. *IEEE Wireless Communications*, 20(5):24–33, November 2013.

[12] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa. Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures. *ACM Computing Surveys*, 52(2):1–40, May 2019.

[13] R. P. Jover. Security analysis of sms as a second factor of authentication. *ACM Queue*, 18(4):37–60, August 2020.

[14] K. G. Kangas. Clandestine transmissions and operations of embedded software on cellular mobile devices. Master's thesis, Naval Postgraduate School, September 2011.

[15] M. La Polla, F. Martinelli, and D. Sgandurra. A survey on security for mobile devices. *IEEE Communications Surveys & Tutorials*, 15(1):446–471, March 2012.

[16] B. W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, October 1973.

[17] I. Lombera, L. Moser, P. Melliar-Smith, and Y.-T. Chuang. Mobile decentralized search and retrieval using sms and http. *Mobile Networks and Applications*, 18:22–41, February 2013.

[18] W. Mazurczyk and L. Caviglione. Steganography in modern smartphones and mitigation techniques. *IEEE Communications Surveys & Tutorials*, 17(1):334–357, August 2014.

[19] W. Mazurczyk, P. Szary, S. Wendzel, and L. Caviglione. Towards reversible storage network covert channels. In *Proc. of the 14th International Conference on Availability, Reliability and Security (ARES'19), Canterbury, UK*, pages 1–8. ACM, August 2019.

[20] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on scientific and statistical computing*, 4(3):553–572, 1983.

[21] S. Muthuswamy and P. Ganapathi. Mobile device security: A survey on mobile device threats, vulnerabilities and their defensive mechanism. *International Journal of Computer Applications*, 56(14):24–29, October 2012.

[22] S. Narteni, I. Vaccari, M. Mongelli, M. Aiello, and E. Cambiaso. On the feasibility of covert channels through short-message-service *(in print)*. In *Proc. of the Italian Conference on Cybersecurity (ITASEC'21), Online Conference*. Cybersecurity National Laboratory, April 2021.

[23] B. Narwal, A. K. Mohapatra, and K. A. Usmani. Towards a taxonomy of cyber threats against target applications. *Journal of Statistics and Management Systems*, 22(2):301–325, March 2019.

[24] S. Patiburn, V. Iranmanesh, and P. Teh. Text steganography using daily emotions monitoring. *International Journal of Education and Management Engineering*, 7(3):1–14, May 2017.

[25] M. Z. Rafique, K. Khan, K. Alghatbar, and M. Farooq. Embedding high capacity covert channels in short message service (sms). In *Proc. of the 8th FTRA International Conference on Secure and Trust Computing, Data Management, and Application (STA'11), Loutraki, Greece*, volume 186 of *Lecture Notes in Computer Science*, pages 1–10. Springer-Verlag, June 2011.

[26] B. Rashidi and C. Fung. A survey of android security threats and defenses. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 6:3–35, October 2015.

[27] S. Schmidt, W. Mazurczyk, R. Kulesza, J. Keller, and L. Caviglione. Exploiting ip telephony with silence suppression for hidden data transfers. *Computers and Security*, 79:17–32, November 2018.

[28] F. Shang, X. Li, D. Zhai, Y. Lu, D. Zhang, and Y. Qian. On the distributed jamming system of covert timing channels in 5g networks. In *Proc. of the 2nd IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA'20), Dalian, China*, pages 1107–1111. IEEE, June 2020.

[29] G. J. Simmons. *The Prisoners' Problem and the Subliminal Channel*, pages 51–67. Springer US, 1983.

[30] M. Taleby Ahvanooey, Q. Li, M. Rabbani, and A. Rajput. A survey on smartphones security: Software vulnerabilities, malware, and attacks. *International Journal of Advanced Computer Science and Applications*, 8(10):30–45, October 2017.

[31] Y.-a. Tan, X. Zhang, K. Sharif, C. Liang, Q. Zhang, and Y. Li. Covert timing channels for iot over mobile networks. *IEEE Wireless Communications*, 25(6):38–44, December 2018.

[32] S. P. Thompson and M. Gardner. *Geometrical Meaning of Differentiation*, chapter 10. Macmillan Education

UK, 1998.

[33] F. Ullah, M. Edwards, R. Ramdhany, R. Chitchyan, M. A. Babar, and A. Rashid. Data exfiltration: A review of external attack vectors and countermeasures. *Journal of Network and Computer Applications*, 101:18–54, January 2018.

[34] K. Ullah, I. Rashid, H. Afzal, M. M. W. Iqbal, Y. A. Bangash, and H. Abbas. Ss7 vulnerabilities—a survey and implementation of machine learning vs rule based filtering for detection of ss7 network attacks. *IEEE Communications Surveys Tutorials*, 22(2):1337–1371, February 2020.

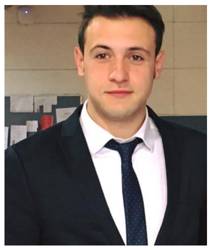
[35] S. Zander. Detecting covert channels in fps online games. In *Proc. of the 42nd IEEE International Conference on Local Computer Networks (LCN'17), Singapore, Singapore*, volume 1, pages 555–558, October 2017.

[36] S. Zander, G. Armitage, and P. Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys & Tutorials*, 9(3):44–57, September 2007.

[37] X. Zhang, Y.-A. Tan, C. Liang, Y. Li, and J. Li. A covert channel over volte via adjusting silence periods. *IEEE Access*, 6(1):9292–9302, February 2018.

Author Biography

Sara Narteni got her M.Sc. in Bioengineering at the University of Genoa on March 2020, with a thesis titled "Pleural line ultrasound videos analysis for computer aided diagnosis in acute pulmonary failure". She is currently a research fellow at the IEIIT institute of Consiglio Nazionale delle Ricerche. She works on data analytics and machine learning topics from different fields, such as industry, healthcare and automotive. Moreover, her research interests also concern computer security topics, including covert channels and Internet of Things.



Ivan Vaccari got his Ph.D. in computer science and a computer engineer degree with laude at the University of Genoa, respectively in 2021 and 2017. During his research activities, he worked in different European projects focused on security in healthcare data, IoT and financial infrastructures. He is currently a research fellow at IEIIT institute of Consiglio Nazionale delle Ricerche, working on IoT and network security focused on identification of vulnerabilities and developed of innovative cyber threats. Regarding detection and mitigation systems, he is working on machine learning and artificial intelligence approaches.

Maurizio Mongelli obtained his Ph.D. Degree in Electronics and Computer Engineering from the University of Genoa (UNIGE) in 2004. The doctorate was funded by Selex Communications S.p.A. (Selex). He worked for both Selex and the Italian Telecommunications Consortium (CNIT) from 2001 until 2010. During his doctorate and in the following years, he worked on the quality of service for military networks with Selex. He was the CNIT technical coordinator of a research project concerning satellite emulation systems, funded by the European Space Agency; and he spent three months working on the project at the German Aerospace Center in Munich. He is now a researcher at the Institute of Electronics, Computer and Telecommunication Engineering (IEIIT) of the National Research Council (CNR), where he deals with machine learning applied to bioinformatics and cyber-physical systems. He is co-author of over 100 international scientific papers and 2 patents.

Maurizio Aiello graduated in 1994, worked as a free-lance consultant both for universities and research center and for private industries. From August 2001, he is responsible of CNR network infrastructure. He is a Teacher at the University of Genoa and University College of Dublin; students Coordinator, fellowships and EU projects in the computer security field. His research activities are on network security and protocols.

Enrico Cambiaso got his Ph.D. degree in Computer Science at the University of Genoa, while working for Ansaldo STS and Selex ES, both companies are part of the Finmeccanica group. He has a strong background as computer scientist and he is currently employed at the IEIIT institute of Consiglio Nazionale delle Ricerche, as a technologist working on cyber-security topics and focusing on the design of last generation threats and related protection.