# On the feasibility of covert channels through Short-Message-Service

Sara Narteni, Ivan Vaccari, Maurizio Mongelli, Maurizio Aiello and Enrico Cambiaso

*Consiglio Nazionale delle Ricerche (CNR-IEIIT), Genoa, Italy*

**Abstract**

Short-Message-Service (SMS) is one of the most used ways to exchange text messages offered by mobile networks. In this paper, we explore the feasibility of establishing a tunneling system based on SMS. Our proposed scheme allows a client (e.g. a web browser) to connect to the Internet by encapsulating HTTP/HTTPS requests/responses into SMS messages. Along with the development of the tunneling architecture and the attack modelling, we conducted some preliminary tests to evaluate the feasibility of the proposed tunneling system. Obtained results indicate that our system is able to handle up to 1200 bytes of web request data when using equal mobile operators, whereas 1248 bytes for different mobile operators.

**Keywords**

covert channel, tunneling system, short-message-service, data exfiltration, cyber-security

## 1. Introduction

Covert channels represent a huge category of cyber-attacks consisting in communicating information and data in a hidden way, as suggested by the word *covert* itself. They represent one of the major threats affecting networks security, as they can lead to the exfiltration of sensitive data [1]. In this context, Lampson [2] introduced covert channels by 1973: according to his definition, they are communication channels *"not intended for information transfer at all"*. In order to understand how covert channels work, it's useful to remind Simmons's Prisoner Problem [3]: 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 be prevented. To bypass Wendy's control, Alice and Bob must exchange innocuous messages (*overt channel*) containing hidden information (*covert channel*). Alice and Bob can be two computers in a network that communicate exploiting different protocols [4].

Nowadays, there are many techniques to create a covert channel. Covert channels can be classified in different ways: for example, according to the distinct Open System Interconnection (OSI) layers [5] or on the basis of the mechanism over which they're constructed. Zander [6]

*ITASEC21: Italian Conference on Cybersecurity, April 7-9, 2021, Online*

EMAIL: sara.narteni@ieiit.cnr.it (S. Narteni); ivan.vaccari@ieiit.cnr.it (I. Vaccari); maurizio.mongelli@ieiit.cnr.it (M. Mongelli); maurizio.aiello@ieiit.cnr.it (M. Aiello); enrico.cambiaso@ieiit.cnr.it (E. Cambiaso)

ORCID: 0000-0002-0579-647X (S. Narteni); 0000-0001-7721-5737 (I. Vaccari); 0000-0001-6201-6225 (M. Mongelli); 0000-0001-5008-225X (M. Aiello); 0000-0002-6932-1975 (E. Cambiaso)

followed the last criterion to provide an overview of the existing covert channels. Protocols of the network and transport layers (IP, TCP, ICMP) present header fields that can be misused to convey secret data: this can be achieved 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 [7]. An alternative is payload tunneling [6], that is a covert channel where a forbidden protocol is encapsulated in 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 [6]. 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. Recent studies proposed covert channels in the field of mobile vocal calls belonging to CTC category, exploiting VoLTE (VoIP over LTE) Protocol [8] and VoIP [9]. Both works hide covert data into silent periods of the voice signal. Along with the discovery of new covert channels, researches on the development of new detection techniques have been carried out in recent times [10].

In recent years, we have assisted to a rapid growth of mobile technologies, so their security is now a very important issue to take into account. It is often compromised: as an example, a well-known phone company has recently declared that private and technical data have been stolen from users' SIM cards [1].

Covert channels affecting mobile networks worth to be investigated; in particular, text messages like SMSs can constitute a new opportunity to construct a covert channel, since they are now widely available at relatively low cost.

In this paper, we study the feasibility of a novel tunneling system based on CSC, in which application data are carried inside Short-Message-Service (SMS) payload. Since people bring their mobile phones everywhere today, as essential parts of their everyday activities, such devices may become instruments for an attacker too. In fact, thinking about the insider threat scenario [11], the insider (i.e. a member of an organization that wants to exfiltrate data to the outside) could make misuse of mobiles to bypass his/her organization network by exploiting the proposed SMS covert channel scheme, thus exposing the organization to serious damages.

The remaining of the paper is structured as follows: Section 2 introduces SMS and their functioning, while Section 3 describes the concept idea of the proposed innovative covert channel. Instead, Section 4 reports some preliminary tests on the feasibility to establish SMS-based tunnels. Section 5 reports the related works on the topic. Finally, Section 6 concludes the paper and reports further work on the topic.

## 2. Short-Message-Service

Short-Message-Service (SMS) is a service that allows the exchange of text messages between mobile devices. A first implementation of SMS was carried by Global System for Mobile Communications (GSM), the first standard developed for mobile telephony systems in the world. SMS was implemented by GSM according to the European Telecommunication Stardard Institute

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

(ETSI) technical specifications, then it has been carried out in the scope of 3G Partnership Project (3GPP) activities so that, by now, the service works in 3G and 4G networks too.

A single SMS comprises a header, with useful information, and a payload, which is the text content. The maximum size of a single SMS is 160 characters with 7-bit encoding.

The flow of an SMS through the GSM network involves the following steps [12]:

1. An SMS is sent from the Mobile Station (MS), that is the mobile device equipped with Subscriber Identity Module (SIM) card, to the Base Transceiver Station (BTS), which receives and digitalizes the radio signals from MS.

2. BTS forwards the SMS to the Mobile Switching Centre (MSC), which has the Equipment Identity Register (EIR) and Authentication Register (AUC) databases for equipment verification and user authentication.

3. The message is then transmitted from MSC to SMSC, where it is stored in a queue until it can't be delivered. SMSC interrogates its databases (Home and Visitor Location Register (HLR and VLR)) for the location information about the recipient's MS.

4. By finding the location of the target MS, the message is sent from SMSC to the respective MSC and from here to the final recipient.

There are two ways to submit a message on GSM network: Protocol Description Unit (PDU) mode, that is in bits, and text format, i.e. the normal text. PDU format is more structured: the message appears as a string of hexadecimal octets, grouped in different parts containing useful information regarding the message and also details about the user. In PDU mode, SMS packets are different depending on if the SMS is being sent or received: `SMS_SUBMIT` or `SMS_DELIVER` formats are used respectively (see Section 3.3 of [12] for further details).

### 2.1. AT Commands

AT (ATtention) commands are a set of short text string commands used to control GSM phones or modems. They can support many actions, including all the processes involved in sending/receiving of SMSs. To understand the AT commands usage, in Table 1 we report a list of the most common ones in the field of SMSs.

Parameters can be added to the AT commands after an = sign to enable functionalities (e.g. `AT+CGMF=1` for SMS text mode). There are lots of AT commands, but manufacturers usually do not implement all of them in the same way and with the same parameters. Hence, the use of AT commands cannot be generalized for all kind of devices.

## 3. The Proposed Covert Channel

In network security context, the aim of a tunneling system is to carry a given communication protocol into the payload of another one. Typically, such systems may be used to overcome network limitations (i.e. firewalls) and potentially to leak private and sensitive data. Such scenario can fit the insider threat problem [11].

| AT command | Description |
|---|---|
| AT | Returns OK when the communication between the device and the application has been verified |
| AT+CLIP | Refers to the GSM/UMTS supplementary service CLIP (Calling Line Identification Presentation) |
| AT+CRC | Controls whether or not the extended format of incoming call indication or GPRS network request for PDP context activation or notification for VBS/VGCS calls is used |
| AT+CMGF | Message format (0 for PDU mode, 1 for Text mode) |
| AT+CNMI | New message indications to Terminal Equipment |
| AT+CMGL | Lists all text messages that are stored in memory |
| AT+CMGD | Deletes messages from memory |
| AT+CMGW | Writes SMSs to message storage area of SIM card |
| AT+CMSS | Sends SMSs from message storage area |
| AT+CMGS | Sends SMSs directly to the recipient's phone number |

**Table 1**
List of common AT commands

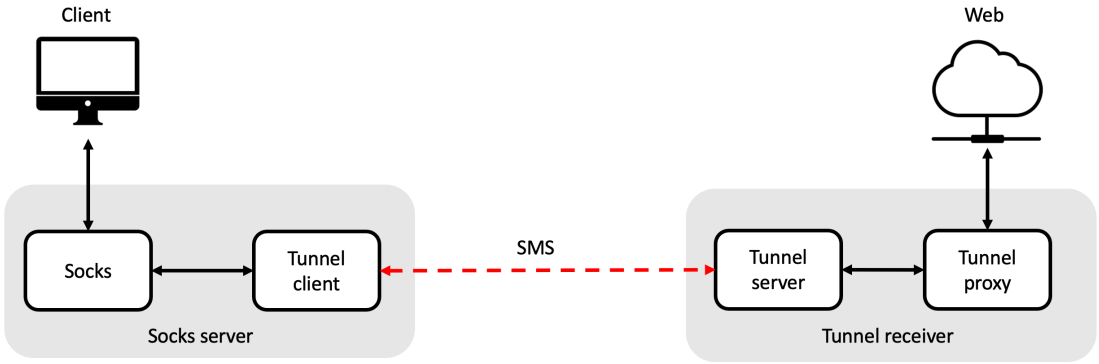


**Figure 1:** SMS Tunneling architecture

Inspired by the rapid development of mobile technologies and the drastic cost reduction of SMSs, we decided to investigate if it is possible to implement a tunneling system that exploits the SMS protocol.

Our SMS tunneling system can be described through the architecture shown in Figure 1.

The proposed system is composed of four software modules: socks, tunnel client, tunnel server, and tunnel proxy. In order to understand how the system works, the first two blocks may be ideally coupled to form a socks server and, similarly, the other two blocks are referred to as a unique tunnel receiver component. In our scenario, an attacker establishes a connection over the tunnel by communicating with the socks server through a client (e.g. a web browser). We assume that the socks server is located inside the 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 tunneling system, the client may be able to perform web browsing activities.

Suppose $C \rightarrow S$ and $S \rightarrow C$ being the request and response pathways respectively, involving tunnel client $C$ and tunnel server $S$ modules. We now provide a more detailed description of the single modules; in this context, our focus is on HTTP/HTTPS data, but the involved messages can actually be of any kind:

- **Socks server**. In the $C \rightarrow S$ pathway, it receives the HTTP/HTTPS raw requests from the client, splits their payload into $L$-characters-sized parts and builds an SMS string for each obtained portion. Such SMS is then sent to the tunnel receiver. In $S \rightarrow C$, it listens for incoming connections from tunnel receiver and reconstructs the web response from the received SMSs. Finally, such response reaches the client.

- **Tunnel receiver**. This component acts in a symmetric manner with respect to socks server. It overtly communicates with a web server: while being in $C \rightarrow S$ phase, the tunnel receiver reconstructs the original HTTP/HTTPS request from SMSs received by the socks server; in $S \rightarrow C$, it receives a web response, then it generates SMSs in the same way as socks server does for $C \rightarrow S$ and sends them to the socks server.

Each SMS, both in $C \rightarrow S$ and $S \rightarrow C$ phases, is made up of a string containing five fields, separated by a dash character (-), containing (i) the web server IP address (`destination_ip` in the following), (ii) the web 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. The first three fields represent a kind of header used to match the connection between the client and the socks[2]. In addition, the tunnel proxy connects to `destination_ip:destination_port` to get the information about the requested web page.

The `index` is calculated in order to count how many fragments of the original request/response are generated by splitting it into pieces of $L$ characters. This information is needed to control the correct order of SMSs after they travel along the tunnel.

The `content` field encapsulates the request/response fragment. Before being sent through the tunnel, it is encoded in base 64 in our design, in order to maintain the standard ASCII SMS characters and prevent possible SMS reading issues. Once the SMS reaches its recipient, this field will then be decoded from base 64 for further processings.

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 introduced a particular SMS, which we call *zero packet*. Such SMS has the same header as described above, while its `index` is fixed at `0`. The `content` is put to `1` if the connection is open, whereas it is `0` when closed. When the communication between client and socks server ends, the closure zero packet is used to close it. Similarly, a zero packet with `content=0` may be sent to establish connections, although this may lead a connection closure, if the message with `index=1` is not received before the expiration of the server-side timeout waiting for data after the three-way-handshake is completed.

[2]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.

In order to better define the proposed attack and evaluate if it is possible to exploit it for network communications, we will now model the attack behavior.

### 3.1. Attack modelling

In order to analyze if the proposed tunneling attack can be employed for a specific communication, it is required to analyze the time delays introduced in the communication by the tunnel architecture.

In particular, referring to the scheme depicted in Figure 1, let's suppose that a request message $m_r$ has to be sent from the client to the web server. Once $m_r$ is received and interpreted by the web server, the relative response message $m_s$ is generated and sent back from the web server to the client.

Suppose $|\cdot|$ being the length of $\cdot$, in bytes, and the addition symbol $+$ the concatenation of two strings, expressed in bytes. Let's also define $S_L$ as the maximum size (in bytes) of a single SMS message. Given a message $m$ to be sent through the tunnel by one endpoint to the other, we can compute the number of required SMSs as follows:

$$k_m = \left\lceil \frac{|m|}{S_L} \right\rceil \tag{1}$$

where, for simplicity, we assume that no overhead is introduced on each message (for instance, by headers, footers, or changes in data representation).

Referring to the $C \rightarrow S$ pathway, we have:

$$|m_r| > S_L \implies m_r = m_r^1 + \ldots + m_r^{k_{m_r}} \tag{2}$$

where $k_{m_r} > 1$ is the number of SMS messages required to send a message $m_r$, computed as in Equation 1. Instead, for $k_{m_r} = 1$, we have $|m_r| \leq S_L$.

The expression of $m_r$ in Equation 2 can then be simplified as:

$$m_r = \sum_{i=1}^{k_{m_r}} m_r^i \tag{3}$$

Similarly, for the $S \rightarrow C$ pathway, we define $m_s$ as follows:

$$m_s = \sum_{i=1}^{k_{m_s}} m_s^i \tag{4}$$

with $k_{m_s}$ being the number of SMSs needed to send $m_s$, defined as in Equation 1.

Now, in order to define if it is possible to exploit the proposed tunnel, it is required to analyze the network requirements of the specific channel which has to be established. Such channel depends both on the client and web server characteristics. For our scenario, such characteristics are browser and server timeouts.

In particular, considering a specific channel, let's suppose that, through the proposed tunnel, the number of sent SMSs per second is $N$.

| Notation | Description |
|---|---|
| $L$ | Maximum number of characters per SMS |
| $\lvert \cdot \rvert$ | Bytes length of $\cdot$ |
| $m_r$ | Request message |
| $m_s$ | Response message |
| $S_L$ | Bytes size of 1 $L$-characters SMS |
| $k_m$ | Number of SMSs needed to send message $m$ |
| $N$ | Number of sent SMSs in 1s |
| $T_C$ | Client timeout |
| $T_S$ | Server timeout |
| $T_m$ | Time required to send message $m$ |
| $T_\alpha$ | Time required for web server to process request message |

**Table 2**
Notation used in our attack modelling

Let's suppose $T_C$ be the timeout used by the client/browser and, similarly, $T_S$ be the server timeout. Let's also define $T_m$ the time required to send a message $m$ from one endpoint to the other one, through the proposed tunnel. Particularly, we define $T_m$ in function of $N$ , as follows:

$$T_m = \frac{k_m}{N} \tag{5}$$

In addition, for given $m_r$ and $m_s$, we define $T_\alpha$ the time required to the web server to process the $m_r$ request, in order to produce $m_s$.

Therefore, we expect a server side connection closure if the $T_S$ timeout, related to the $C \rightarrow S$ pathway, expires, according to the Equation 6.

$$T_S < T_{m_r} \tag{6}$$

Also, we expect a client side connection closure if the $T_C$ timeout, related to both the $C \rightarrow S$ and $S \rightarrow C$ pathways, expires, according to the Equation 7.

$$T_C < T_{m_r} + T_\alpha + T_{m_s} \tag{7}$$

Finally, we state that the proposed tunnel is effective if Equation 8 is satisfied.

$$T_{m_r} + T_\alpha + T_{m_s} \leq T_C \wedge T_{m_r} \leq T_S \tag{8}$$

For sake of clarity, in Table 2 we wrap up the notation adopted throughout our attack description.

## 4. Tests and obtained results

In order to perform an evaluation on the feasibility of the proposed attack, we conducted some preliminary tests based on AT commands (Section 2.1), aimed at studying the behavior of SMSs sent between the endpoints.

For the implementation of our system, we used two Raspberry Pis, one as the socks server and one as the tunnel receiver, each interfaced with a USB HSPA modem provided with a slot for a SIM card.

We performed a direct sending test, where SMSs were sent to the recipient without intermediate steps, using the AT+CMGS command (Table 1).

Let's suppose $m_i,\ i = [1, \ldots, N_s]$ as the sequence of $N_s$ different SMS messages that our system is able to continuously send in a fixed time window $T$ (expressed in seconds), being $N_s \leq k_m$ the number of messages which is possible to send in that time interval, with $k_m$ defined as in Equation 1. The goal of the preliminary tests we have accomplished is then to compute $N_s$ in two different test cases, involving (i) the same mobile operator (TIM Italian mobile carrier) for both sender and receiver, and (ii) a different mobile operator between sender (TIM Italian mobile carrier) and receiver (Iliad Italian mobile carrier). In both test cases, the content of each message $m_i$ was made up of a text string containing also a counter, which we use to evaluate the SMSs sending, also in terms of reception order or losses.

Such tests were conducted by fixing $T = 60$s as time window. Within such time window, we found out the following results:

$$N_s^1 = 15 \tag{9}$$

for the first scenario (same mobile operator), and

$$N_s^2 = 16 \tag{10}$$

for the second scenario (different mobile operators).

Referring to the attack model presented in Section 3.1, we exploited the obtained $N_s$ values to compute the total maximum size $|m|$ of HTTP/HTTPS requests/responses payload we can transfer over the tunnel, measured in bytes.

First of all, we computed the frequency of SMS sending, expressed by $N$ (i.e. how many SMSs are sent per second), as follows:

$$N = \frac{N_s}{T} \tag{11}$$

For our two scenarios we obtained $N^1 = 0.25$ SMS/s (same operator) and $N^2 = 0.26$ SMS/s (different operators).

Referring to Equation 8, we can assume, for simplicity, $T_\alpha = 0$ s , hence an extremely reduced server-side computation time. This may be reasonable, considered the greater impact of both $T_{m_r}$ and $T_{m_s}$ . We can also assume $T_C = 30$ s, corresponding to a client-side timeout (e.g. a browser timeout) of 30 seconds. Finally, we can assume $T_S = 300$ s, as it represents, for instance, the default Apache2 timeout [13]. Particularly, as the web server may be under the control of the attacker, the server timeout may also (mathematically) tend to infinite. Therefore, we can consider only the first/left part of Equation 8, where, for simplicity in notation, we define:

$$T_m = T_{m_r} + T_{m_s} \tag{12}$$

hence:

$$T_m \leq T_C \tag{13}$$

with $T_m$ defined in Equation 5.

At this point, according to Equation 5, by fixing $T_m = T_C = 30$ s (as we do not want to expire the client timeout), we get $k_m = T_m \cdot N$. The maximum payload length can be then computed as $|m| = T_m \cdot N \cdot S_L$ (Equation 1). By assuming, for simplicity, that the size of SMS messages to be sent through the tunnel is the highest supported by the SMS protocol, we fix $S_L = 160$ bytes/SMS (i.e., 160 characters with 7-bit encoding).

Therefore, in our two test cases we get:

$$|m^1| = T_m \cdot N^1 \cdot S_L = 1200 \text{ bytes} \tag{14}$$

for the first scenario (same mobile operator), and

$$|m^2| = T_m \cdot N^2 \cdot S_L = 1248 \text{ bytes} \tag{15}$$

for the second scenario (different mobile operators). All payloads exceeding such sizes will lead to a connection closure for the considered web scenario. Also, since the results show $|m^1| < |m^2|$, the exploitation of equal operators seems to slightly have a negative impact on the tunnel performance. Other scenarios (like chats or SMTP data exchange) will be investigated in the scope of further works on the topic.

Moreover, by monitoring the SMS reception, we noticed that the order of the received SMSs was not the same as the one in which the messages were sent: we expected such a behavior, in fact our tunneling system embeds a reordering mechanism for the messages received from a tunnel endpoint to another one.

## 5. Related Work

In the era of mobile technologies outbreak, people are prone to make a massive use of their mobile phones, they download lots of apps without being aware of the risks they may encounter. For this reason, security of mobile devices is always more under threat.

In recent years, a lot of work ([14], [15],[16], [17], [18]) has been done to collect different vulnerabilities, threats and attacks addressing mobile devices and their Operating Systems. Between them, there are attacks versus the victim device itself (such as viruses, spyware, Trojan, rootkit etc), while others steal information by eavesdropping the communication between different devices (as for the case of Man-In-The-Middle attack) [14]. As regards the SMS context, some attacks exploit SMSs for data leakage, e.g. Smishing (phishing via SMS) attack [15], that consists in stealing information by sending trustworthy SMSs to the users.

In another survey ([19]), SMS interception and manipulation attacks are presented, where attackers act as SMSC and make use of fake MSC in order to get IMSI and address of the victim, thus identifying it.

SMSs are now a common method for two-factors authentication, even if they're not so secure. In [20], 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.

Focusing on GSM network, a second generation standard for cellular networks that reached 90% of market share by 2017 [21], very few works explored the feasibility of creating covert

channels exploiting SMS protocols. The author in [22] demonstrated the possibility to embed secret information inside SMS User Data Header redundant fields, in such a way that an independent warden (recalling Simmons's Prisoner Problem [3]) can interpret the SMS as normal. Another work illustrated how to secretly convey data through SMS [23]: the aim of such study was to develop a system able of transmitting SMS messages from an Android device without the end user's knowledge. Such a system design includes 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. In addition to the SMS-based approaches mentioned so far, it's also possible to hide information directly in text messages or other media: this method is known as steganography. It can be achieved in many different ways: as an example, secret data may be hidden in text manipulations (like abbreviations, blank spaces, synonyms and acronyms [24]) or into the widely used emoticons [25]. However, steganographic methods are a sub-category of covert channels, being based on modifications of the contents of the data being transmitted. Moreover, all the SMS-based data exfiltration techniques mentioned so far are different from tunneling systems, since they do not involve network protocols.

The idea of an SMS tunneling was investigated in [26] too, where a so-called WebSIM was developed, i.e. a SIM card acting as a web server for personal security. In this context, 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 connection outside. Another work [27] developed a query/retrieve system, called iTrust with SMS, which allows users to search/get information cointained in iTrust with HTTP network nodes[28] via SMS. The requests are made through text keywords and then this framework is able to convert them to GET HTTP requests. 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 studies investigated the feasibility of a SMS tunneling, intended as a way to exchange forbidden information between two nodes via SMS.

## 6. Conclusions and Further Work

In this paper we have theorized an innovative tunneling attack based on the exchange of SMS messages. Such a system may represent a potentially dangerous threat, as the stealing of private and sensitive data could lead to serious problems in critical contexts like health, finance or industry. In addition, using a third-party network, potentially uncontrolled, may hinder detection and mitigation of the proposed threat. Despite many malicious tunneling applications already exist and are well-established, involving the most common network protocols (DNS, ICMP, SSH, etc), our proposed scheme allows to exploit a protocol in the field of mobile networks. In our concept, this feature represents a great advantage, in virtue of the huge development of mobile devices and the fact that the sending of SMSs has become cheaper in recent years.

Nevertheless, in this paper we have presented a preliminary investigation on the feasibility of the SMS tunneling. Due to the limited size of SMS messages, we expect performance limitations of the tunnel: however, the obtained preliminary results indicate a promising starting point for further investigations. Namely, future works will be addressed to a deeper testing of the

system, aimed to the individuation of the SMS tunneling capabilities and weaknesses.

## Acknowledge

This work was supported by the following research project: the Integrated Framework for Predictive and Collaborative Security of Financial Infrastructures (FINSEC) project, which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 786727.

## References

[1] W. Mazurczyk, P. Szary, S. Wendzel, L. Caviglione, Towards reversible storage network covert channels, in: Proceedings of the 14th International Conference on Availability, Reliability and Security, 2019, pp. 1–8.

[2] B. W. Lampson, A note on the confinement problem, Commun. ACM 16 (1973) 613–615.

[3] G. J. Simmons, The prisoners' problem and the subliminal channel, in: CRYPTO, 1983.

[4] S. Zander, Detecting covert channels in fps online games, in: 2017 IEEE 42nd Conference on Local Computer Networks (LCN), 2017, pp. 555–558.

[5] T. G. Handel, M. T. Sandford, Hiding data in the osi network model, in: Information Hiding, Springer Berlin Heidelberg, 1996, pp. 23–38.

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

[7] Y. Heda, R. Shah, Covert channel design and detection techniques : a survey, in: 2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), 2015, pp. 1–6.

[8] X. Zhang, Y. Tan, C. Liang, Y. Li, J. Li, A covert channel over volte via adjusting silence periods, IEEE Access 6 (2018) 9292–9302.

[9] S. Schmidt, W. Mazurczyk, R. Kulesza, J. Keller, L. Caviglione, Exploiting ip telephony with silence suppression for hidden data transfers, Computers and Security 79 (2018).

[10] L. Caviglione, Trends and challenges in network covert channels countermeasures, Applied Sciences 11 (2021). doi:10.3390/app11041641.

[11] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, M. Ochoa, Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures, ACM Comput. Surv. 52 (2019). URL: https://doi.org/10.1145/3303771. doi:10.1145/3303771.

[12] A. Chaudari, Security analysis of sms and related technologies, 30 Nov 2015.

[13] E. Cambiaso, G. Papaleo, G. Chiola, M. Aiello, Designing and modeling the slow next dos attack, in: Computational Intelligence in Security for Information Systems Conference, Springer, 2015, pp. 249–259.

[14] M. Taleby Ahvanooey, Q. Li, M. Rabbani, A. Rajput, A survey on smartphones security: Software vulnerabilities, malware, and attacks, International Journal of Advanced Computer Science and Applications 8 (2017) 30–. doi:10.14569/IJACSA.2017.081005.

[15] S. Muthuswamy, P. Ganapathi, Mobile device security: A survey on mobile device threats,

vulnerabilities and their defensive mechanism, International Journal of Computer Applications 56 (2012) 24–29. doi:10.5120/8960-3163.

[16] M. La Polla, F. Martinelli, D. Sgandurra, A survey on security for mobile devices, Communications Surveys & Tutorials, IEEE 15 (2013) 446–471. doi:10.1109/SURV.2012.013012.00028.

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

[18] S. Farhan, S. F. Zaidi, A. Munam, M. Shah, M. Kamran, Q. Javaid, S. Zhang, A survey on security for smartphone device, International Journal of Advanced Computer Science and Applications 7 (2016) 206–219. doi:10.14569/IJACSA.2016.070426.

[19] K. Ullah, I. Rashid, H. Afzal, M. M. W. Iqbal, Y. A. Bangash, 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 (2020) 1337–1371. doi:10.1109/COMST.2020.2971757.

[20] R. P. Jover, Security analysis of sms as a second factor of authentication, Commun. ACM 63 (2020) 46–52. URL: https://doi.org/10.1145/3424260. doi:10.1145/3424260.

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

[22] M. Z. Rafique, K. Khan, K. Alghatbar, M. Farooq, Embedding high capacity covert channels in short message service (sms), in: Communications in Computer and Information Science, volume 186, 2011, pp. 1–10.

[23] K. G. Kangas, Clandestine transmissions and operations of embedded software on cellular mobile devices, 2011.

[24] W. Mazurczyk, L. Caviglione, Steganography in modern smartphones and mitigation techniques, IEEE Communications Surveys & Tutorials 17 (2015) 334–357.

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

[26] S. Guthery, J. Posegga, M.-m. Deutsche, The websim - clever smartcards listen to port 80, 2000.

[27] I. Lombera, L. Moser, P. Melliar-Smith, Y.-T. Chuang, Mobile decentralized search and retrieval using sms and http, Mobile Networks and Applications 18 (2013). doi:10.1007/s11036-012-0412-0.

[28] Y.-T. Chuang, M. Isaí, L. Lombera, P. Moser, Melliar-Smith, Trustworthy distributed search and retrieval over the internet, 2012.