Journal of Computer Virology and Hacking Techniques (2024) 20:27–39
https://doi.org/10.1007/s11416-023-00484-z

ORIGINAL PAPER

Persistent MobileApp-in-the-Middle (MAitM) attack

Christian Catalano[1] · Franco Tommasi[1]

Received: 12 October 2022 / Accepted: 9 May 2023 / Published online: 30 June 2023

Abstract
The recent publication of the "Browser in the Middle" attack has demonstrated an effective way to compromise a good number of variants of Multifactor Authentication and to control the information flow between the victim an the accessed service. That attack was mainly aimed at the victim use of a desktop browser to access a service. The present paper shows how that attack may be extended to involve the mobile environment and how, thanks to that enhancement, the attack may also gain the persistence attribute. The new attack is named MobileApp-in-the-Middle (MAitM). Again, as in BitM, no installation of malware on the victim's platform is needed with MAitM.

Keywords Browser in the middle · Man in the Middle (MitM) · Web-based Trojan malware · Mobile malware · Phishing · Cyber attack · Advanced persistent threat · Multi-factor authentication

1 Introduction

The publication of the "Browser in the Middle" (BitM) attack [1] has aroused some interest in cybersecurity circles and even among the generic public [2–6]. The present research work aims at proving that the technique used for that attack can be refined so as to be also utilized in the mobile environment, where it can also be enhanced with the persistence attribute.

It is worth noting that very recently (January 2023) the BitM attack technique has also been accepted and acknowledged by MITRE on CAPEC list (CAPEC ID-701) as an attack pattern which is reported to stand as an high typical severity with medium likelyhood of success. The present research work aims at proving that the technique used for that attack can be refined so as to be also utilized in the mobile environment, where it can also be enhanced with the persistence attribute.

We maintain that, when properly combined with other current technologies, such technique can represent a serious threat in a few of the current technological contexts in which we live and operate. At the base of the attack is a complete reversal of the classic attack paradigm. In the classic case the attacker aims at compromising the security of a target system. Within the new paradigm, the victim is lured into visiting the attacker's system and led to use it in the belief it is his/her system. By the definition of an APT (Advanced Persistent Threat) [7–9] we also believe the attack technique here introduced can be a component of a set of tools and techniques used to engineer an APT.

In the following a novel attack scenario and a threat model will be analyzed. A PoC for the Android platform and a description of how an attacker could violate in a persistent and unnoticed way the confidentiality, integrity and availability of a mobile user's data will be given. The threat model is a variation of the BitM attack technique which will be called "MobileApp-in-the-Middle" (MAitM). Any mobile application in the market is a potential target for such a technique.

In order to accomplish the PoC (Proof of Concept), the BitM platform has been extended by the introduction of a malicious Progressive Web Application (PWA) [10] designed on purpose and of a reverse proxy named ngrok [11] which is able to expose on the Internet the web server included in the attacker platform. To improve the victim's navigation experience, a malicious mobile device, which will be unwittingly accessed by the victim, had also to be added to the platform.

The attack could be launched through one of the many phishing techniques described in literature [12, 13], then made a persistent one through the PWA technology [10].

✉ Christian Catalano
christian.catalano@unisalento.it

Franco Tommasi
francesco.tommasi@unisalento.it

1 Dipartimento di Ingegneria dell'Innovazione, University of Salento, Via per Monteroni, 73100 Lecce, Italy

1.1 Progressive web application

In these days a web browser is much more than a tool to view web pages. It has turned into an environment which is able to run other applications. Along the years, browsers' vendors tried to standardize the technology by which the browsers give access to their applications, in order to make the access more and more easy and secure and to integrate cloud services and physical devices. With such perspective in mind, in 2015 Google introduced a new browser-centric technology, the Progressive Web Applications. Progressive Web Applications (PWA) [14, 15] are a new type of Web applications aiming at providing native app-like browsing experiences even when a browser is offline. A progressive web application (PWA) is a piece of software delivered through the web, built using common web technologies including HTML5, CSS and JavaScript together with push notifications [16], caches [17], and service workers [18, 19]. It is intended to work on any platform that uses a standards-compliant browser, including desktop and mobile devices. Since PWAs are ultimately a type of webpage or website, they do not require separate bundling or distribution. Developers can just publish the web application online, ensure that it meets the minimum "installability requirements" [20] intended to guarantee an adequate level of security, and users will be able to add the application to their home screen. As a matter of fact they do not require the access to the user's platform official online app store (i.e. Apple App Store, Google Play, Microsoft Store or Samsung Galaxy Store) and are in fact a way to get around the mobile devices vendors verifications. PWAs are a rather recent trend and researchers are now busy studying their impact on security and privacy [21, 22]. The present research work will demonstrate how the PWA technology on the Android OS, combined with the BitM attack technique, may be exploited in a dishonest way by an attacker, even when the PWA security requirements are met. The described threat model will exploit the PWA technology to make the attack persistent and unnoticed.

2 MAitM attack scenario

An overview of the MobileApp-in-the-Middle (MAitM) threat model will be provided in the present section. The basic idea is to lure the victim by a phishing technique into installing a malicious PWA into his/her device. The malicious PWA will allow the victim to unwittingly navigate the mobile application installed as usual, from the OS official store, but actually resident on the BitM attack platform (now renamed MAitM), which in turn will be connected to the services offered by the originally intended target.

The code needed to allow the victim the installation of the malicious PWA is hosted on the MAitM attacker platform. It must respect the security requirements imposed by Google otherwise it will not run on the victim platform. Such code will redirect the victim in a totally transparent way to the attacker platform mobile application. The purpose of the interposition is to intercept, to record and to manipulate any data exchange between the victim and the service provider in a persistent way through the official service application.

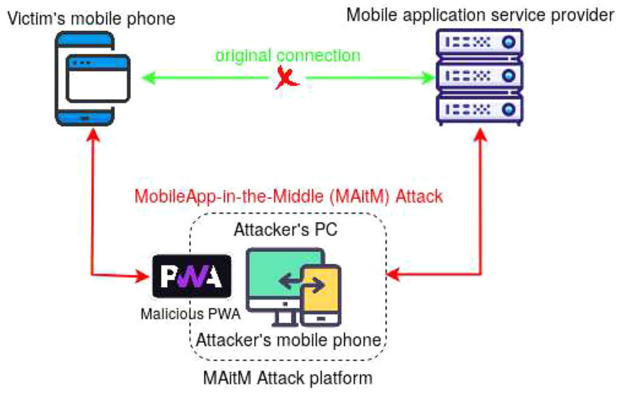


Fig. 1 MobileApp-in-the-Middle (MAitM)

Figure 1 illustrates the basic concept.

The involved parties are:

- *The attacker*: a computer criminal aiming at violating the victim's privacy and security together with confidentiality, integrity and availability of the data exchanged by two end-points. The attacker sets up and controls a malicious MAitM attack platform which allows the victim to access the target, that is a transparent mobile application located in the same attack platform.
- *The victim*: an user lured by phishing techniques into accessing the malicious server (hosting a mobile application and the installation code for the malicious PWA) setup by the attacker. The victim will be induced to install the malicious PWA on his/her own device.
- *The target*: a mobile application able to provide sensitive and/or valuable services, both accessed through an authentication mechanism (e.g. a mobile banking app, mobile Microsoft Office 365, etc.) or not (e.g. Google mobile browser). Exactly as it was with BitM, the present technique allows bypassing many of the multifactor authentication mechanisms implemented by vendors. In order to be able to operate freely and to avoid potential legal problems with more sensitive services (e.g. Internet banking), during the entire experimentation conducted for the present research work, the mobile Microsoft Office 365 app was chosen.

The following steps summarize the MAitM attack technique 2:

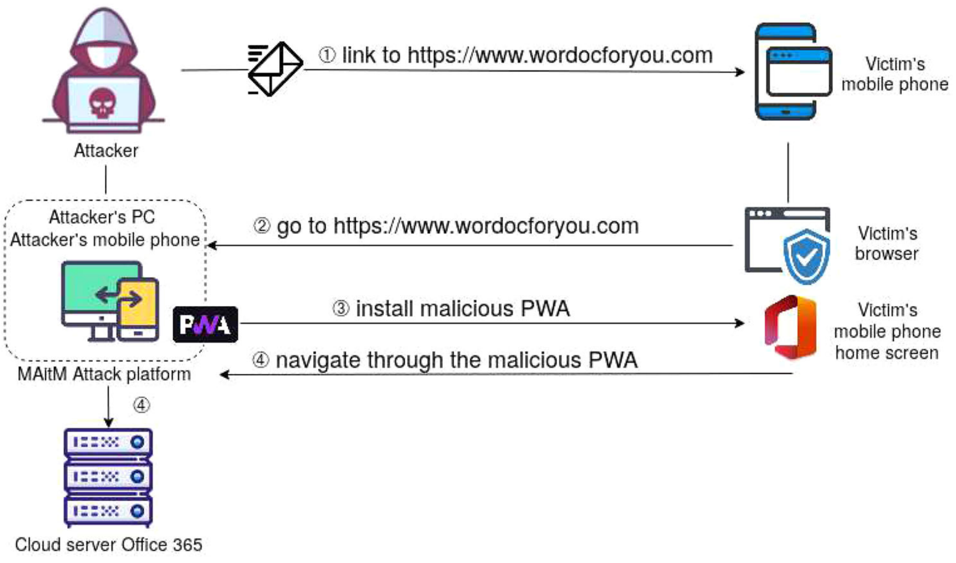


Fig. 2 Persistent MobileApp-in-the-Middle (MitM) attack scenario

① The attacker sets up a web site containing a link pointing to a Microsoft Word document. In order to be allowed to read it, the victim is offered the installation of the malicious PWA, actually hosted on the MAitM attack platform.

② After completing the malicious PWA installation, the victim will be unwittingly connected to the malicious server where the real mobile application (the mobile Microsoft Office 365 app in our experimentation) is transparently residing, together with a number of programs (e.g.web proxy, sniffer, keylogger, etc.) the attacker will use to intercept, record and manipulate the data exchanged by the victim and the target mobile application server.

③ The malicious PWA installation causes a fake Microsoft Office 365 app icon to appear on the victim mobile device screen so that, in the future, no phishing attempt is needed. The icon will be tapped whenever the victim wishes to access the desired service.

④ Through the malicious PWA, the victim is redirected to the target mobile application in a totally transparent way. At this point the victim will be able to use all the services he/she could access as if the target mobile application were really installed on his/her mobile device (plus the fraudulent ones the rogue server is now able to offer in addition). All the victim's operations may now be intercepted and therefore recorded and manipulated by the attacker in a persistent and hidden way, no matter which security protocols the mobile application has set up.

3 Malicious PWA and MAitM platform architecture and implementation

The following section will provide a detailed description of the techniques used. The MAitM platform architecture is based on a desktop computer and a mobile phone, like the one we adopted in the past for another research work, the Mobile Session Fixation Attack in Micropayment Systems [23]

The basic idea is to serve the victim a PWA which will transparently allow the victim to execute a mobile application located on the attacker phone, with the belief the application is actually installed on his/her phone (while it is not) as it looks and behaves exactly as the desired mobile application. It may be useful to rephrase the concept: the purpose of the attack platform is to lure the victim into navigating by his/her phone a mobile application installed on another, malicious, mobile phone in a fully transparent way, so that the victim security and privacy are totally compromised (Fig. 2).

The starting point to mount the attack was a modification of the 4th section of [1], titled "BitM platform architecture and implementation". Figure 3 draws on Fig. 5 in that paper to describe the MAitM architecture, by labelling with "new" the functional blocks which have been added to the original BitM design.

In the following a description of the blocks will be given:

- *PWA code*: it is the code needed to implement the Malicious Progressive Web Application. To install a PWA and to allow the access to a web site through it, certain requirements imposed by Google must be met. Therefore the malicious PWA code hosted on the MAitM attack platform includes the following files:
  - *manifest.json*: a web manifest, with the correct fields filled in [24]
  - *serviceWorker.js*: a javascript file, registered to allow the app to work offline (this is required only by Chrome for Android currently) [25]

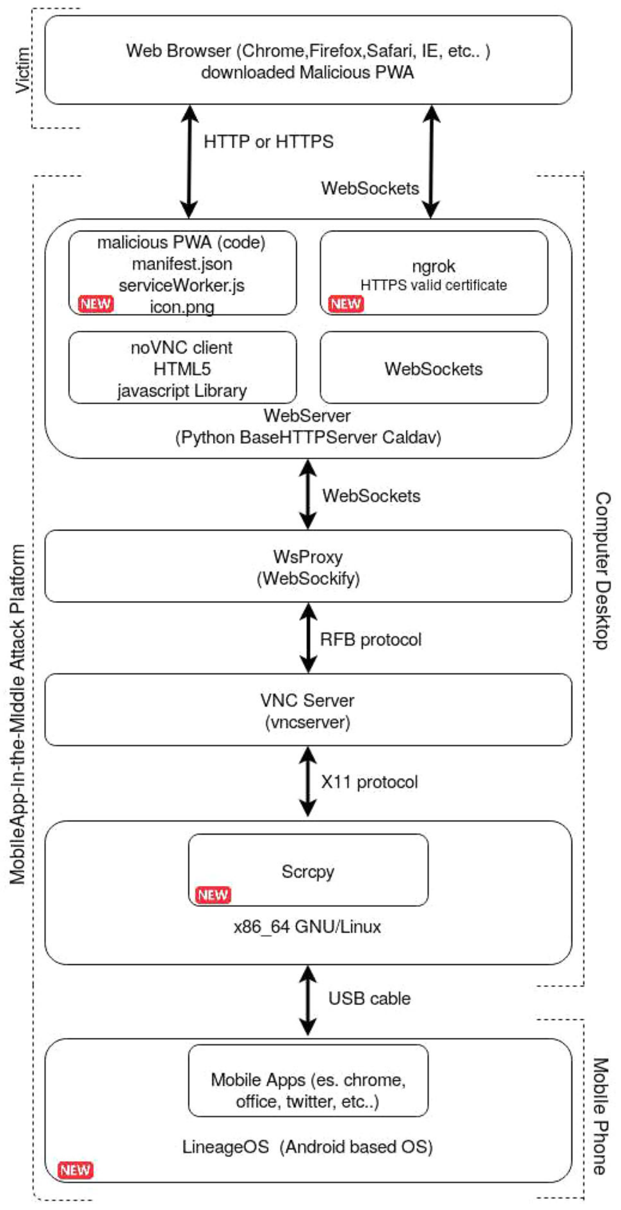


Fig. 3 Persistent MobileApp-in-the-Middle (MitM) attack scenario

  – *mobile app icon*: an icon to represent the app on the device (the mobile Microsoft Office 365 app, in our case)

- *ngrok* [11]: a further security requirement imposed by Google to allow the installation of the PWA on the victim's device is the use of the HTTPS protocol to access the web site by the PWA. To this purpose the piece of software called *ngrok* was used. *ngrok* is a cross-platform application that, with minimal effort, enables developers to expose a local development server to the Internet with a valid SSL certificate.
- *Screen Copy (Scrcpy)* [26]: This application provides display and control of Android-based devices connected via USB or over TCP/IP. This program is used to export the physical device (a mobile phone in our case) on the MAitM attack platform through an USB connection. By its services, the victim can be directed to unwittingly operate with any mobile application installed on the mobile phone included in the attacker platform.
- *LineageOS* [27]: an Android-based operating system installed on the USB-connected malicious mobile phone, the fundamental element of the attack platform. LineageOS was chosen for its being fully customizable and because it allows root access. On the malicious mobile phone running it, the mobile application to be exposed to the victim could be installed (through Google Play Store). Being the OS administrators also allows the installation of a number of malicious programs (e.g. LokiBoard-Android-Keylogger [28]), the removal of the OS status bar and the full control of the mobile device.

4 Experimental results

In this section the results of an experiment executed by the mobile Microsoft Office 365 app [29] will be reported. The same experiment may be tried with any mobile application installable on Android. The scenario is the one described in the "MAitM attack scenario" section.

The testbed was set up as follows:

- *Victim's mobile phone*: a phone with the following specifications OnePlus 7, Android Oxygen OS 11, Display 6.41" 2340x1080.
- *Attacker's platform*: the attack platform is set up and equipped with the software described in Sect. 3. The attack platform hardware is the following: Computer laptop ThinkPad with processor Intel Core i7 7th Gen, 16 Gb RAM, GNU_Linux Distro Ubuntu 20.04, connected through an USB cable to the following mobile phone: Mobile Phone OnePlus 7 Pro, LineageOS 17.1-20210207-NIGHTLY-guacamole, Display 6.67" 3120x1440.
- *Mobile application target*: mobile Microsoft Office 365 application. It is assumed the victim owns an active account allowing the full use of the selected mobile application. Although Microsoft Office 365 was selected on account of being a very popular application and of allowing fast and easy tests, it must be stressed the method here introduced applies exactly in the same way to any cloud web application based on a two-factor authentication procedure (Internet banking apps, Telegram, Gmail, Facebook, Twitter, etc.). The victim is redirected through social engineering techniques (for example through phishing techniques) to a website (es. https://wastasy.eu.ngrok.io) containing a link to a word

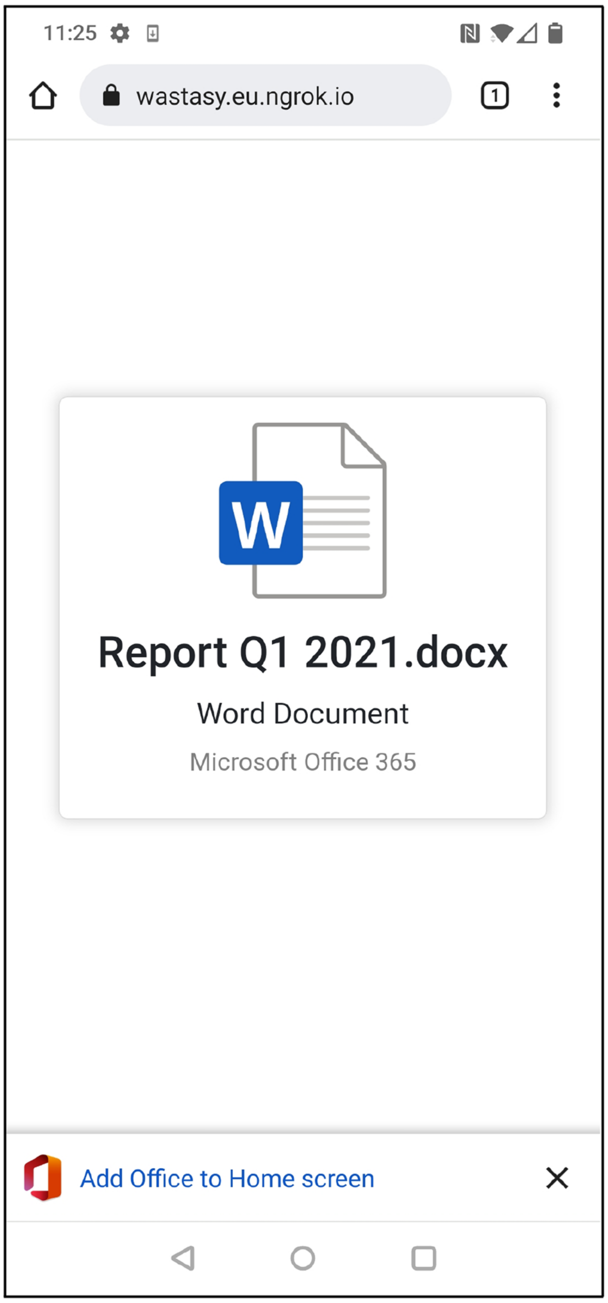


Fig. 4 Victim's mobile phone view: malicious PWA ready to be installed

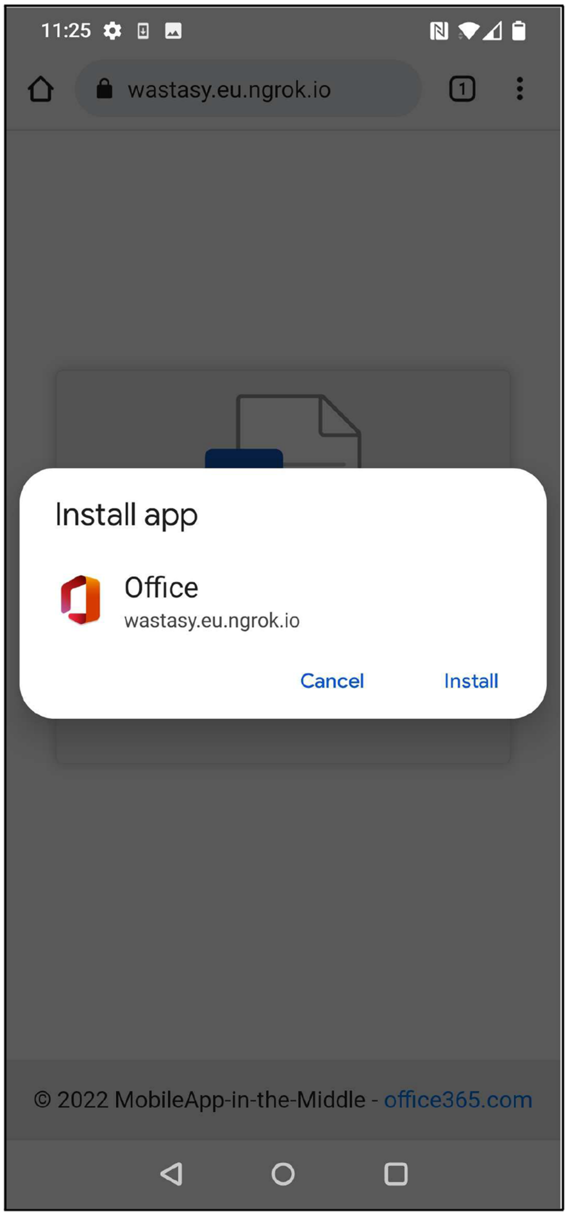


Fig. 5 Victim's mobile phone view: download and execution of the malicious PWA

document and a malicious PWA ready to be installed Fig. 4.

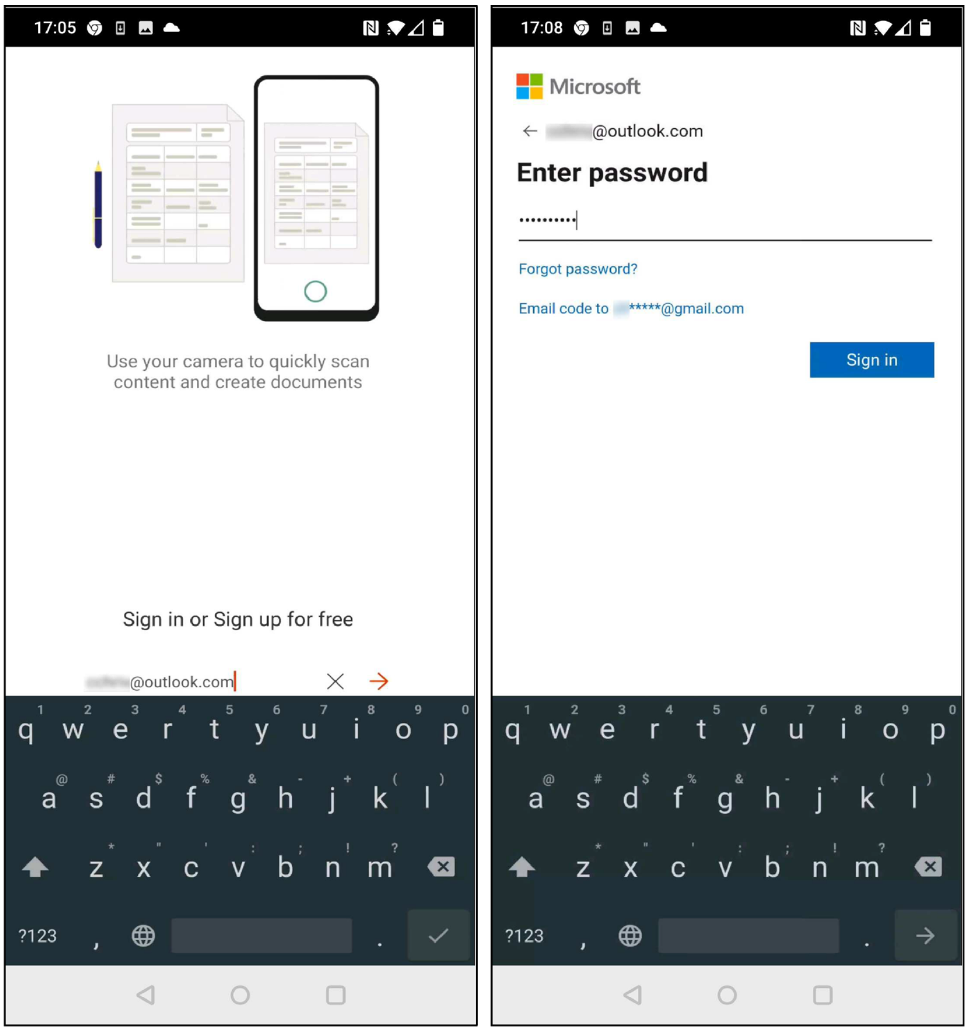
Any click on this screen will trigger the following proposal to install the Office application Fig. 5, which is silently assumed to be needed to read the document. A click on "Install" will eventually cause both the download and execution of the PWA on the victim's phone and the addition of the "Office" icon on the victim's Home screen.

Once the installation of the PWA is completed, the victim will be able to proceed entering the credentials for the use of the Microsoft Office 365 mobile app (see Figs. 6a, b).

Once again it is important to emphasize that, while the device in the figures is the victim's, the shown keyboard is actually on the MAitM attack platform and the authentication control is managed by the attacker remote mobile app and not by the browser.

The following reports what happens at the same time on the MAitM attack platform Fig. 7.

Fig. 6 Victim's mobile phone view


17:05
Use your camera to quickly scan content and create documents
Sign in or Sign up for free
@outlook.com
17:08
Microsoft
@outlook.com
Enter password
Forgot password?
Email code to *****@gmail.com
Sign in

(a) Victim's e-mail **(b)** Victim's password

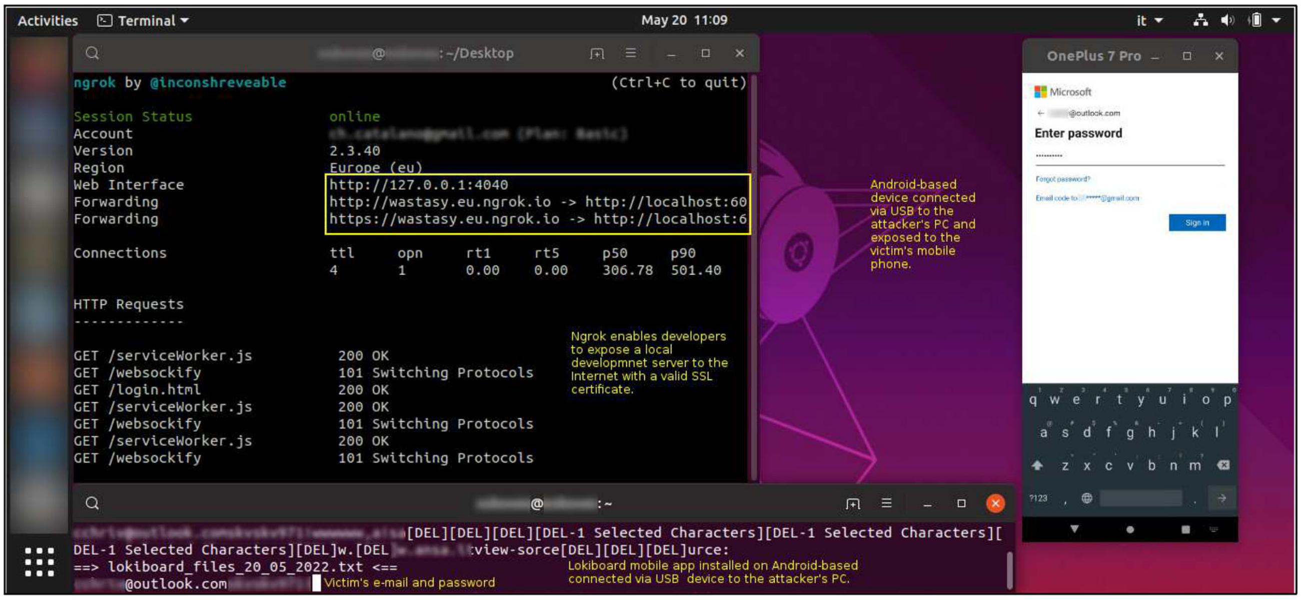
The malicious PWA is permanently installed on the victim's mobile phone (see the "Office" icon) and will always point to the MAitM platform Fig. 8.

At this point, the attacker will not only have gained the victim's credentials (through a mobile application, i.e. LokiBoard-Android-Keylogger, installed on the malicious mobile phone), but he/she will also be able to control the victim in real-time whenever the victim accesses the remote mobile app through the installed icon (that is, intercept, record and manipulate the data exchange between the victim and the remote accessed service). As it was with the *Browser in the Middle attack* [1], in the experiment here described, a *Passive* and *Server-Side* MAitM attack was carried on by installing a keylogger mobile application (LokiBoard-Android-Keylogger) on the attacker mobile phone. As the simple example here documented demonstrates, it is possible to carry on a successful attack without exploiting a zero-day or any other known vulnerability at the two end-points (the victim mobile phone and the mobile Microsoft Office 365) or in the communication channel. Besides, the attack was entirely conducted by a remote location, simply by an improper use of known technologies.

5 Mobile browser in the mobile browser attack (MBitMB)

Because of its general use to access many different services, a particular mobile application, the mobile browser Google Chrome, is particularly interesting to be "exported" in the same fashion seen for the mobile Microsoft Office app, from the attacker platform to the victim's phone, so that, whenever the victim launches his/her own browser, he/she is in fact using the remote attacker browser app.

In this section, a demonstration of how such result can be obtained will be given. To provide a better representation of



```
[redacted]@[redacted]:~/Desktop

ngrok by @inconshreveable                                    (Ctrl+C to quit)

Session Status                online
Account                       [redacted]
Version                       2.3.40
Region                        Europe (eu)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://wastasy.eu.ngrok.io -> http://localhost:60
Forwarding                    https://wastasy.eu.ngrok.io -> http://localhost:6

Connections                   ttl     opn     rt1     rt5     p50     p90
                              4       1       0.00    0.00    306.78  501.40

HTTP Requests
-------------

GET /serviceWorker.js          200 OK
GET /websockify                101 Switching Protocols
GET /login.html                200 OK
GET /serviceWorker.js          200 OK
GET /websockify                101 Switching Protocols
GET /serviceWorker.js          200 OK
GET /websockify                101 Switching Protocols
```

```
[redacted]@[redacted]:~

[redacted][DEL][DEL][DEL][DEL-1 Selected Characters][DEL-1 Selected Characters][
DEL-1 Selected Characters][DEL]w.[DEL[redacted]:view-sorce[DEL][DEL][DEL]urce:
==> lokiboard_files_20_05_2022.txt <==
[redacted]@outlook.com[redacted]
```

Activities
Terminal
May 20 11:09
it
Ngrok enables developers to expose a local developmnet server to the Internet with a valid SSL certificate.
Android-based device connected via USB to the attacker's PC and exposed to the victim's mobile phone.
Victim's e-mail and password
Lokiboard mobile app installed on Android-based connected via USB device to the attacker's PC.
OnePlus 7 Pro
Microsoft
[redacted]@outlook.com
Enter password
Forgot password?
Email code to [redacted]*****@gmail.com
Sign in

Fig. 7 MobileApp in the Middle (MAitM) attack platform

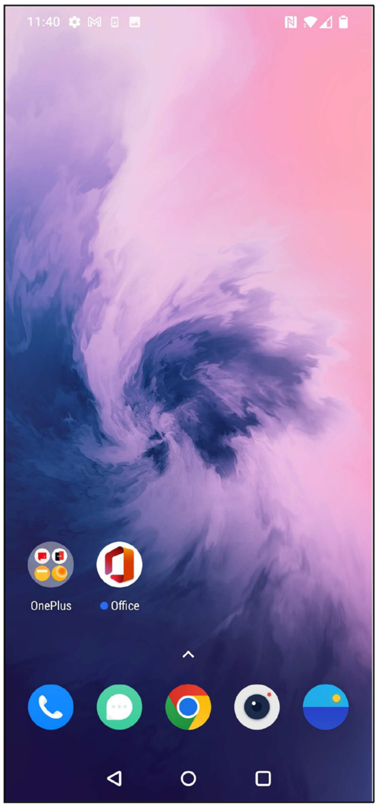


Fig. 8 Malicious PWA is permanently installed on the victim's mobile phone

the differences between the use of the victim native browser and the use of the one hosted on the attacker remote platform, the victim device has been connected through an USB cable to a PC and the web sites remote debug procedure for Android devices has been run [30].

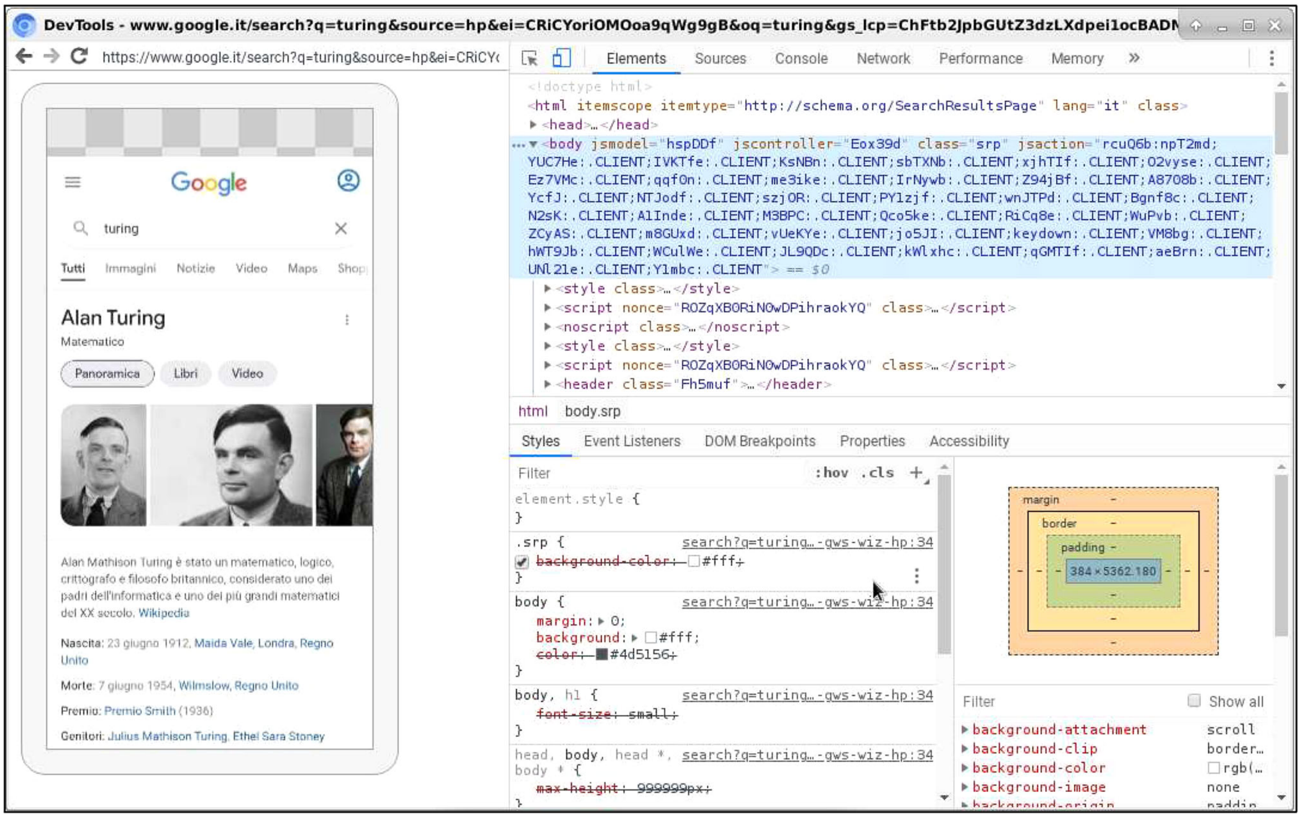
Following are the results of the performed experimentations. Figure 9 reports the results of the debug procedure for a normal use of the victim's native browser.

In the following Fig. 10 the results of the same debug procedure are shown when the victim is using the local mobile browser to actually display what is in the attacker remote mobile browser (without realizing it).

The debugger allows to verify the true URL displayed by the victim. In the first case it will be the address returned by an authentic Google search. In the second case, the address is that of the MAitM attack platform, as exposed on the Internet by the *ngrok* program.

Also to be noted is the page source code displayed by the debugger. In the first case it is the expected HTML code for the results of a Google search. In the second case it consists of the simple iframe tag that has been used to incorporate the screen of the mobile phone connected to the attacker platform.

The last particular worth noting is the different appearence of the keyboard. In the first case (Fig. 11), the keyboard is not shown as the debugger shows the web page and the keyboard is not part of it (being displayed by the operating system). In the second case (Fig. 12) the keyboard is shown because the entire view of the screen on the mobile phone connected to the attacker platform is exported through VNC (thus including the keyboard seen on that screen).



Alan Mathison Turing è stato un matematico, logico, crittografo e filosofo britannico, considerato uno dei padri dell'informatica e uno dei più grandi matematici del XX secolo. Wikipedia

Nascita: 23 giugno 1912, Maida Vale, Londra, Regno Unito

Morte: 7 giugno 1954, Wilmslow, Regno Unito

Premio: Premio Smith (1936)

Genitori: Julius Mathison Turing, Ethel Sara Stoney

```
<!doctype html>
<html itemscope itemtype="http://schema.org/SearchResultsPage" lang="it" class>
 ▶<head>…</head>
…▼<body jsmodel="hspDDf" jscontroller="Eox39d" class="srp" jsaction="rcuQ6b:npT2md;
 YUC7He:.CLIENT;IVKTfe:.CLIENT;KsNBn:.CLIENT;sbTXNb:.CLIENT;xjhTIf:.CLIENT;O2vyse:.CLIENT;
 Ez7VMc:.CLIENT;qqf0n:.CLIENT;me3ike:.CLIENT;IrNywb:.CLIENT;Z94jBf:.CLIENT;A8708b:.CLIENT;
 YcfJ:.CLIENT;NTJodf:.CLIENT;szjOR:.CLIENT;PYlzjf:.CLIENT;wnJTPd:.CLIENT;Bgnf8c:.CLIENT;
 N2sK:.CLIENT;AlInde:.CLIENT;M3BPC:.CLIENT;Qco5ke:.CLIENT;RiCq8e:.CLIENT;WuPvb:.CLIENT;
 ZCyAS:.CLIENT;m8GUxd:.CLIENT;vUeKYe:.CLIENT;jo5JI:.CLIENT;keydown:.CLIENT;VM8bg:.CLIENT;
 hWT9Jb:.CLIENT;WCulWe:.CLIENT;JL9QDc:.CLIENT;kWlxhc:.CLIENT;qGMTIf:.CLIENT;aeBrn:.CLIENT;
 UNl21e:.CLIENT;Y1mbc:.CLIENT"> == $0
   ▶<style class>…</style>
   ▶<script nonce="ROZqXBORiNOwDPihraokYQ" class>…</script>
   ▶<noscript class>…</noscript>
   ▶<style class>…</style>
   ▶<script nonce="ROZqXBORiNOwDPihraokYQ" class>…</script>
   ▶<header class="Fh5muf">…</header>
```

```
element.style {
}
.srp {                        search?q=turing…-gws-wiz-hp:34
✓ ~~background-color: □#fff;~~
}
body {                        search?q=turing…-gws-wiz-hp:34
   margin: ▶ 0;
   background: ▶ □#fff;
   ~~color: ■#4d5156;~~
}
body, h1 {                    search?q=turing…-gws-wiz-hp:34
   ~~font-size: small;~~
}
head, body, head *,           search?q=turing…-gws-wiz-hp:34
body * {
   ~~max-height: 999999px;~~
}
```

```
▶ background-attachment     scroll
▶ background-clip           border…
▶ background-color          □rgb(…
▶ background-image          none
▶ background-origin         paddin
```

Fig. 9 Remote debugging of the web site: https://www.google.it

6 Physical attacks

An important implication of the threat so far described is worth mentioning. While some type of phishing technique was needed to lure the victim into installing the PWA pointing to the attacker platform, whoever has a physical access to a device, even for a short time, could be able to directly install the same PWA on it, thus again pointing to the attacker platform where every malicious action above described can be performed. No test was performed on this potentiality within the present experimentation but it is quite easy to imagine it can represent a very serious problem in automotive or industrial environments.

7 About the MAitM attack and multi factor authentication countermeasures

Because of its invisibility and persistence, the attack so far described lends itself rather easily to the design of an APT. As of this writing, it can be claimed that most of the apps available for Android devices may be manipulated in order to engineer a MAitM attack. While exploring MAitM countermeasures, the most obvious recommendation for the generic user is the urge to pay the utmost attention to the sites he visits and to the proposed installations of applications.

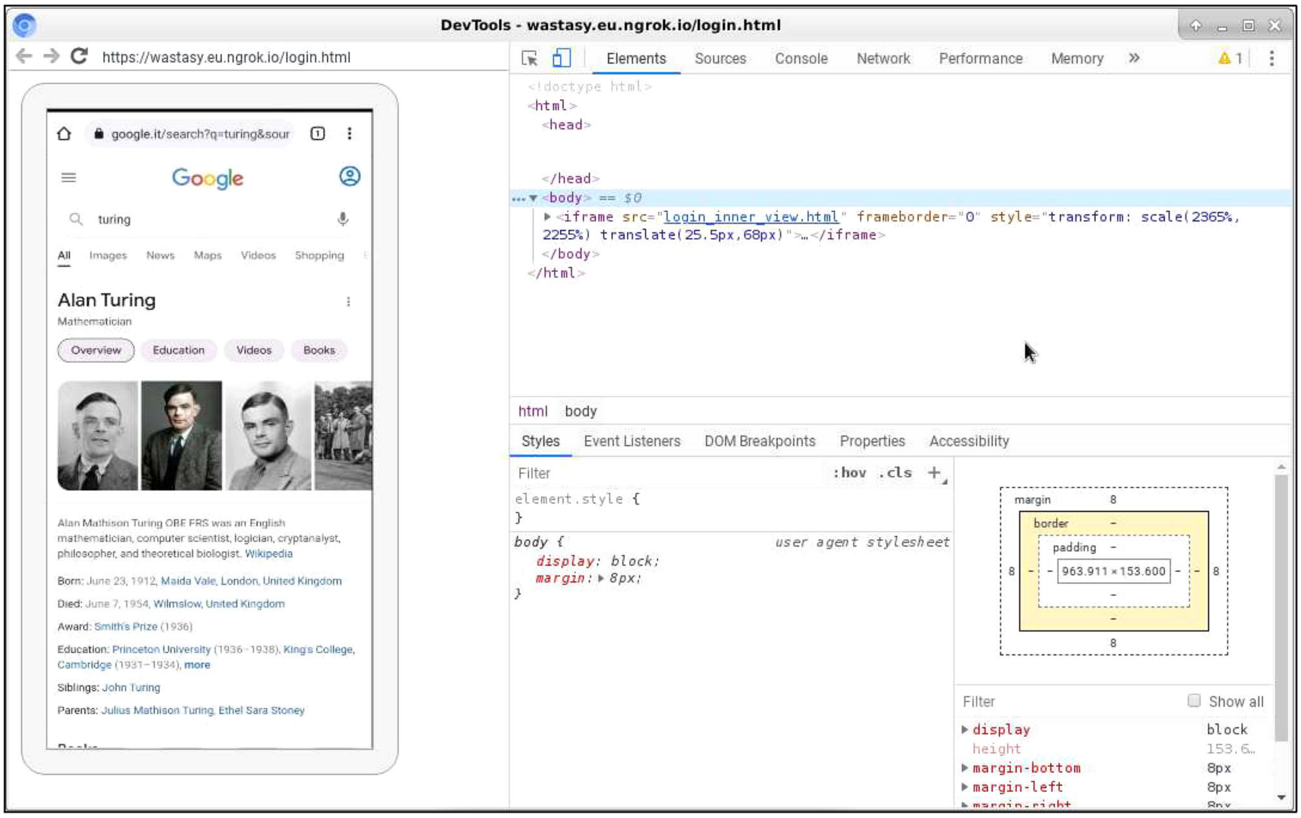
It should also be observed that some mobile applications are more resilient than others for this type of attacks.

This may be true for applications relying on some type of multifactor authentication.

Nowadays user accounts that need identity verification can choose among a great number of Multi-Factor Authentication (MFA/2FA) methods [31–33]. Multi-factor Authentication is a method of account access control which a user can pass successfully accomplishing various authentication stages. Instead of being asked for just a single piece of information, like a password, users are required to provide additional information which makes it more difficult for an intruder to fake the identity of the actual user. Such additional information (i.e. factor) can entail the use of secondary devices, fingerprints, biometric authentication, security tokens etc.

Authentication factors can be classified in this three categories [34–36]:

1. *Knowledge factor*: something that the user knows, e.g., a username and a password;
2. *Possession factor*: something the user has, e.g., a hardware token (as a security token);



```
<!doctype html>
<html>
  <head>

  </head>
... ▾ <body> == $0
    ▸ <iframe src="login_inner_view.html" frameborder="0" style="transform: scale(2365%,
    2255%) translate(25.5px,68px)">…</iframe>
    </body>
  </html>
```

```
element.style {
}
body {                              user agent stylesheet
  display: block;
  margin: ▸ 8px;
}
```

```
▸ display          block
  height           153.6…
▸ margin-bottom    8px
▸ margin-left      8px
```

Alan Turing
Mathematician

Alan Mathison Turing OBE FRS was an English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist. Wikipedia

Born: June 23, 1912, Maida Vale, London, United Kingdom
Died: June 7, 1954, Wilmslow, United Kingdom
Award: Smith's Prize (1936)
Education: Princeton University (1936–1938), King's College, Cambridge (1931–1934), more
Siblings: John Turing
Parents: Julius Mathison Turing, Ethel Sara Stoney

Fig. 10 Remote debugging of the web site: https://wastasy.eu.ngrok.io

3. *Inherence factor*: something verifying who the user is, e.g., fingerprints

Multi-factor authentication can be performed in various ways - the most common of them being the use of login credential with some additional information. Another different trend in authentication methods includes the analysis of usage patterns of input data to determine the authenticity of the user identity, like, for example, the time taken by user to input his details, or the pressure performed by the user's finger.

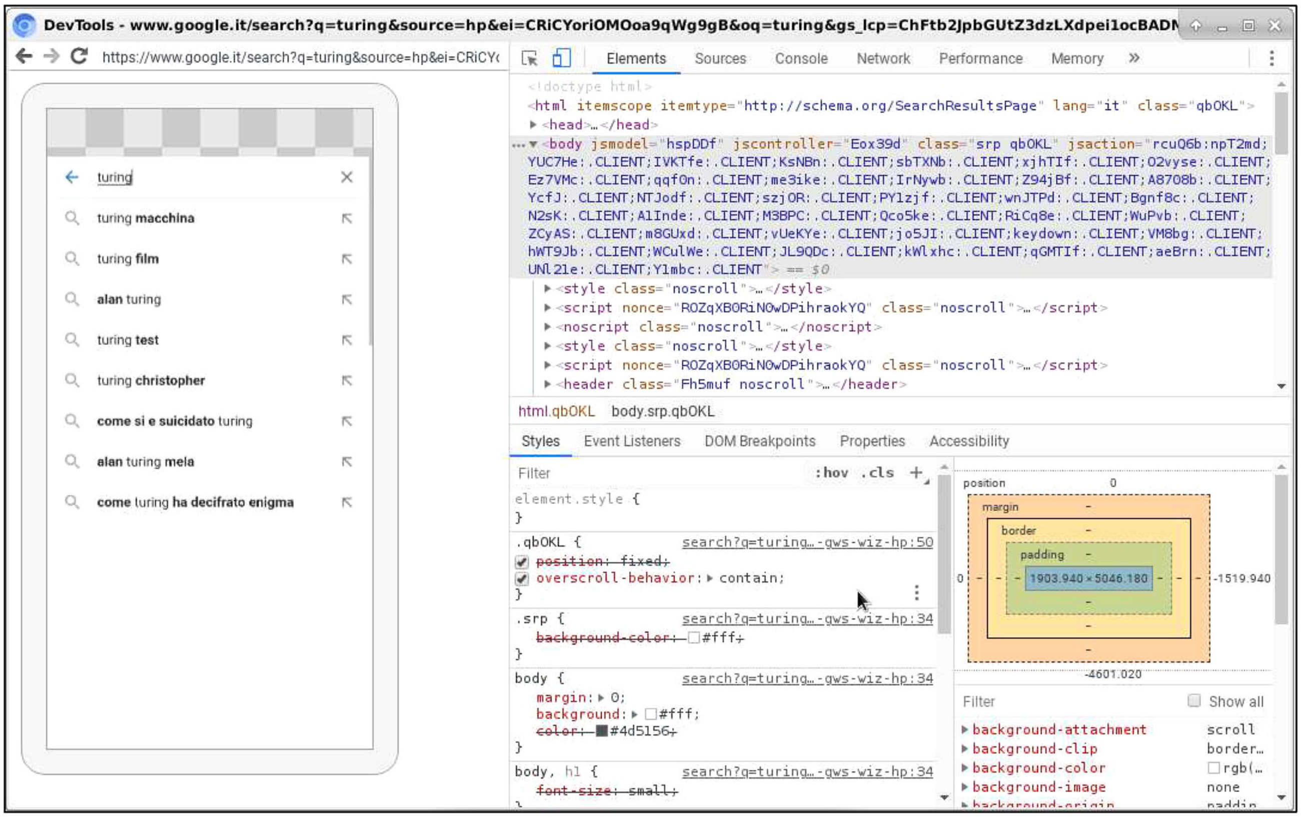
Here is a list of the most common options available today:

- SMS OTP passcode;
- Email Link;
- Push Notification;
- Hardware or Software OTP Token;
- QR Code;
- U2F/WebAuthn Security Key;
- Biometrics (Fingerprinting, Face Recognition, and more).

After a short description of each of the above MFA methods, the next section will make clear in a synopsis (Table 1) which of these methods is bypassable by the MAitM technique.

SMS OTP passcode

One-Time Password sent to the legitimate user's mobile device via an SMS message is one of the oldest forms of 2FA [33, 37–40]. In this scheme, the onetime code is generated on the server side whenever a user tries to login into his account with username and password and is sent via SMS over the cellular network to the user's registered mobile phone number. Authentication occurs when the server recognizes that the user enters in the correct code for login. The user can receive the OTP either as a text message or via an automated call using text-to speech conversion. Additionally, OTP validity is also restricted to a very short period of time and will expire automatically. There is no additional software or hardware requirement in authentication systems that use SMS based OTPs to authenticate or authorize a valid user. As a matter of fact, this is the simplest way of delivering OTPs as SMS enabled devices are available to anyone using the Internet these days. This simple and widely used mechanism for providing Multistep authentication scheme in fact very popular in banking and credit-debit card transactions.



```
<!doctype html>
<html itemscope itemtype="http://schema.org/SearchResultsPage" lang="it" class="qbOKL">
 ▶<head>…</head>
…▼<body jsmodel="hspDDf" jscontroller="Eox39d" class="srp qbOKL" jsaction="rcuQ6b:npT2md;
 YUC7He:.CLIENT;IVKTfe:.CLIENT;KsNBn:.CLIENT;sbTXNb:.CLIENT;xjhTIf:.CLIENT;O2vyse:.CLIENT;
 Ez7VMc:.CLIENT;qqf0n:.CLIENT;me3ike:.CLIENT;IrNywb:.CLIENT;Z94jBf:.CLIENT;A8708b:.CLIENT;
 YcfJ:.CLIENT;NTJodf:.CLIENT;szjOR:.CLIENT;PYlzjf:.CLIENT;wnJTPd:.CLIENT;Bgnf8c:.CLIENT;
 N2sK:.CLIENT;AlInde:.CLIENT;M3BPC:.CLIENT;Qco5ke:.CLIENT;RiCq8e:.CLIENT;WuPvb:.CLIENT;
 ZCyAS:.CLIENT;m8GUxd:.CLIENT;vUeKYe:.CLIENT;jo5JI:.CLIENT;keydown:.CLIENT;VM8bg:.CLIENT;
 hWT9Jb:.CLIENT;WCulWe:.CLIENT;JL9QDc:.CLIENT;kWlxhc:.CLIENT;qGMTIf:.CLIENT;aeBrn:.CLIENT;
 UNl21e:.CLIENT;Y1mbc:.CLIENT"> == $0
   ▶<style class="noscroll">…</style>
   ▶<script nonce="ROZqXBORiNOwDPihraokYQ" class="noscroll">…</script>
   ▶<noscript class="noscroll">…</noscript>
   ▶<style class="noscroll">…</style>
   ▶<script nonce="ROZqXBORiNOwDPihraokYQ" class="noscroll">…</script>
   ▶<header class="Fh5muf noscroll">…</header>
```

```
element.style {
}
.qbOKL {                         search?q=turing…-gws-wiz-hp:50
  ✓ ~~position: fixed;~~
  ✓ overscroll-behavior: ▶ contain;
}
.srp {                           search?q=turing…-gws-wiz-hp:34
    ~~background-color: □#fff;~~
}
body {                           search?q=turing…-gws-wiz-hp:34
    margin: ▶ 0;
    background: ▶ □#fff;
    ~~color: ■#4d5156;~~
}
body, h1 {                       search?q=turing…-gws-wiz-hp:34
    ~~font-size: small;~~
```

```
▶ background-attachment    scroll
▶ background-clip          border…
▶ background-color         □rgb(…
▶ background-image         none
```

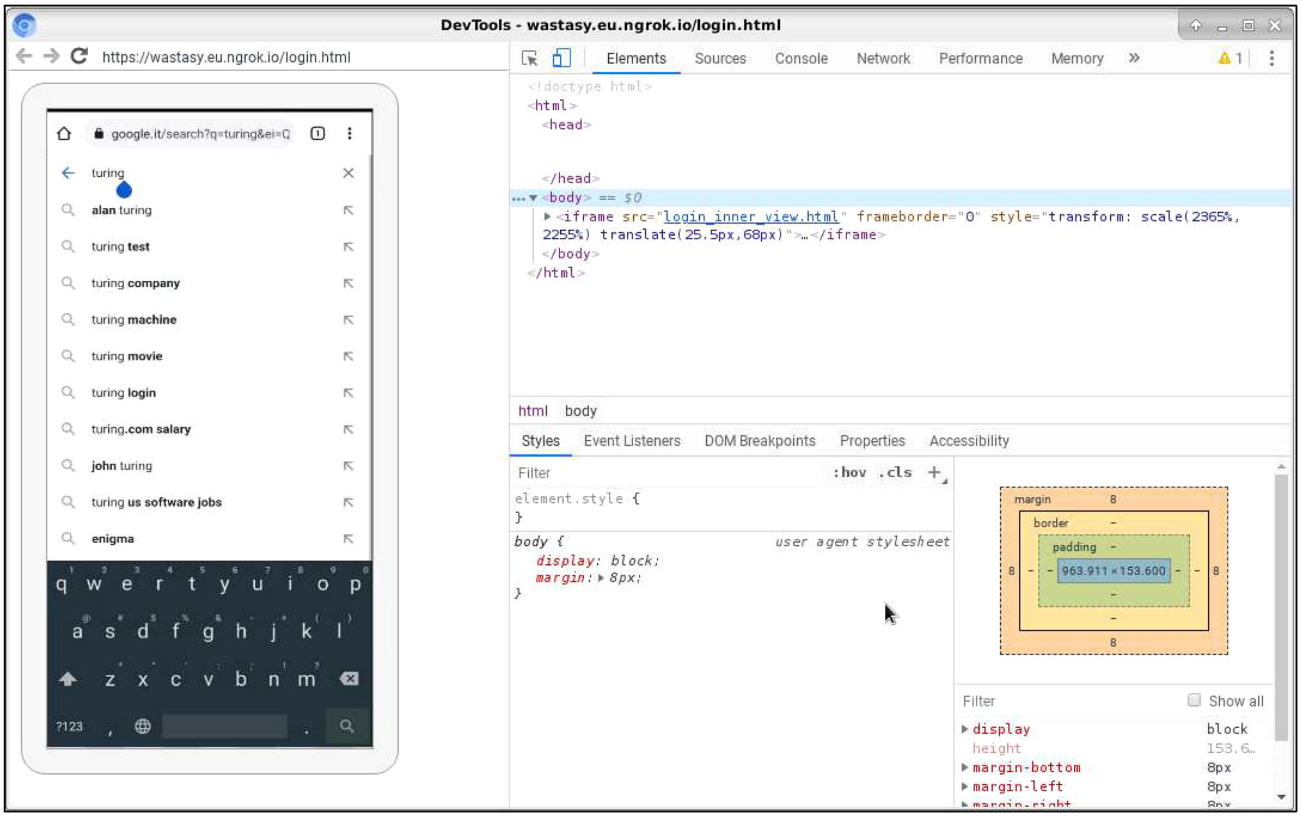
Fig. 11 Remote debugging of the web site: https://www.google.it without keyboard

Email link
Email link is an email message that contains a link user must click to authenticate into the application [33, 41]. Email link is a convenient way to verify identity and, again, it does not require any additional hardware or software. However, confirmation links sent by email cannot be considered a strong MFA method. An email link is just another instance of the Knowledge Factor. If a malicious actor knows the password to your email account, the authentication is compromised. And, as a matter of fact, many users choose the same password for different services. Many are not even aware credentials for a service are a different thing than credentials for another service (it is very common, for example, seeing users registering for a cloud service using their email address and then choosing the same email password), which makes compromising their email accounts much easier.

Push notification
MFA via push notifications utilizes smartphone notifications to assert authentication [33, 42, 43]. This puts this kind of MFA in the category of "something you have," as the users will need to have their smartphone with them to utilize push notifications as a second factor. After inputting their username and password, end users simply need to unlock their phone and then press a button to either approve or deny the access request by a simple interaction.

Hardware or software OTP token
Hardware OTP Token [33] is a physical token which is given to the computer user. It generates and displays One Time unique Passwords (OTPs) that last for very limited time span. While this authentication mechanism provides an extra layer of security, hardware tokens can be physically stolen from the legitimate owners. Also, the user may not immediately report the theft of the security token. This will give the intruder a span of time to breach the protected system. However this could only occur if the unique username and password of account is known to the thief. Ever since mobile apps generating OTPs have emerged, Hardware OTP Tokens have become less and less popular in favor of software tokens that perform the same task.

QR code
The QR Code authentication method is a type of OTP in which the one-time password is a QR code that the user needs to scan using his/her phone [44]. Notable examples of services that use this kind of MFA are WhatsApp, Telegram and Google Authenticator. It resembles the same mechanism of a software token where an application generates an OTP, but in this case the OTP is encoded in a QR code and is auto-



```
<!doctype html>
<html>
  <head>


  </head>
... ▼ <body> == $0
    ▶ <iframe src="login_inner_view.html" frameborder="0" style="transform: scale(2365%,
    2255%) translate(25.5px,68px)">…</iframe>
    </body>
  </html>
```

```
element.style {
}
body {                          user agent stylesheet
   display: block;
   margin: ▶ 8px;
}
```

```
▶ display          block
  height           153.6…
▶ margin-bottom    8px
▶ margin-left      8px
```

Fig. 12 Remote debugging of the web site: https://wastasy.eu.ngrok.io with keyboard

Table 1 MAitM attack feasibility

MFA method	MAitM attack feasibility
SMS OTP passcode	Yes, always
Email Link	Yes, always
Push notification	Yes, always
Hardware OTP Token	Yes, always
Software OTP Token	Yes, always
QR Code	Yes, depending on MFA implementation
U2F/WebAuthn	No
Biometrics	Yes, depending on MFA implementation

matically acquired by the user's mobile device camera. In order to successfully authenticate, the device that frames the QR code with its camera must be a device that was already registered and marked as owned by the legitimate user. It is therefore a type of "possession factor".

U2F/WebAuthn security key
U2F/WebAuthn Security Keys [33, 45] are hardware tokens you plug into the computer to confirm your identity and hence it is again a kind of "possession factor". It usually consists in plugging a physical key dongle into a USB port of the computer used to perform login. After that, the end user has to either tap or touch such key to authenticate into the application.

Biometrics (fingerprinting, face recognition, and more)
Biometric authentication is a method for verifying the users' identity asking them to prove who they are. Such proof must be provided through so called "inherence factors", such as their fingerprint, facial features, iris structure, voice, or typing behavior [33]. These factors contain a large number of unique data points that are really hard to replicate. Because of this, many organizations regard biometric authentication as one of the strongest, if not the strongest, method for verifying users' identities. Biometrics is also very convenient as you always have your identifying tracts with you. No password needs to be remembered and no personal device needs to be carried around in order to authenticate. Many services and platforms offer biometrics on mobile devices, for example fingerprinting or facial recognition are supported on both Android and iOS devices. U2F/WebAuthn Security Keys can use biometric authentication too.

7.1 MAitM effectiveness assessment

All of the above listed methods are more secure than passwords, but some are more secure than others. Each authentication method comes with a unique set of advantages and drawbacks.

More generally, it is possible to establish a criterion to determine whether a MFA method adopted by the target service is vulnerable to a BitM/MAitM attack. The criterion may be stated as follows:

- LET S be a set of N devices (desktop, notebook, smartphone, tablet) in the hands of a legitimate owner;
- LET all N devices in the set be able to act as a terminal to log in the target service.;
- LET them also be able to act as an accessory device to complete the MFA (e.g. the second factor).
- IF the legitimate owner can access the account by a device D in S which is able to provide to the owner only: a) a webview of the web browser with all the functionalities of a basic web navigation (navigate back and forward, address bar, refresh); b) a keyboard; c) a pointing device (mouse, touchscreen);
- AND IF during the MFA procedure the remaining N-1 devices in the set S, possibly involved in the procedure, DO NOT require a physical connection (USB, Ethernet, Wifi, Bluetooth, etc.) to D to successfully complete the MFA,
- THEN the account is vulnerable to the BitM/MAitM attack.

Based on such criterion and on practical experimentations, Table 1 shows which of the described MFA methods are prone to be bypassed by MAitM technique.

To conclude, we believe that the existence of the above described threat, should induce the vendors to reconsider the entire PWA installation process and the authentication methods implemented.

8 Limits and constraints

It can be easily seen that, as it is proposed here, the attack is mainly effective when addressed to a specific target. It should also be noted that the limitations and constraints of the MAitM attack are the same as those described in Section 9 of the previous BitM article [1].

9 Conclusion and future work

A trend to merge different virtual environments (desktop, mobile, cloud, browser) into a single environment, functionally indistinguishable by the final user, can be currently observed. On the other hand this very trend could encourage the development of criminal attacks exploiting the potential confusion generated among the users about what is under his/her control and what is not.

The present research work aimed at showing how a careful integration of different technologies may allow a reversal of the classic attack model: attracting the victim into a well-crafted fraudulent environment totally controlled by the attacker instead of manipulating the user or the service provider systems. It also demonstrated how the defence mechanisms of PWAs can be thwarted, together with some multifactor authentication methods and the browser's cross-origin mechanisms [46], resulting in the confusion of the victim about the system he/she is actually using.

Similar attack models can be mounted for desktop, server, mobile, industrial and automotive contexts. In conclusion we cannot but express the hope the browser, perhaps today's most important door for accessing the external world, will be further fortified.

Funding Open access funding provided by Università del Salento within the CRUI-CARE Agreement. This research received no external funding.

Declarations

Conflicts of interest The authors declare no conflict of interest.

Institutional review board statement Not applicable.

Informed consent statement Not applicable.

References

1. Tommasi, F., Catalano, C., Taurino, I.: Browser-in-the-middle (BiTM) attack. Int. J. Inf. Secur. **21**(2), 179–189 (2022)
2. Sjouwerman, S.: Anti-MFA Phishing Attacks Are Here to Stay – Businesses Need to Prepare. SC Magazine (2022)
3. Abrams, L.: Devious Phishing Method Bypasses MFA Using Remote Access Software. Bleeping Computer (2022)
4. di Corinto, A.: Tre ricercatori italiani hanno scoperto come neutralizzare l'autenticazione a due fattori. la Repubblica (2022)
5. Pirrone, G.: L'autenticazione a due fattori può essere violata? Wired (2022)
6. Doria, M.: L'autenticazione a due fattori non è più sicura, ecco perché. Tom's Hardware (2022)

7. Ahmad, A., Webb, J., Desouza, K.C., Boorman, J.: Strategically-motivated advanced persistent threat: definition, process, tactics and a disinformation model of counterattack. Comput. Secur. **86**, 402–418 (2019)
8. Alshamrani, A., Myneni, S., Chowdhary, A., Huang, D.: A survey on advanced persistent threats: techniques, solutions, challenges, and research opportunities. IEEE Commun. Surv. Tutor. **21**(2), 1851–1877 (2019)
9. Quintero-Bonilla, S., del Martín Rey, A.: A new proposal on the advanced persistent threat: a survey. Appl. Sci. **10**(11), 3874 (2020)
10. Google: Progressive web apps PWA (2022)
11. ngrok: Ngrok (2022)
12. Chiew, K.L., Yong, K.S.C., Tan, C.L.: A survey of phishing attacks: their types, vectors and technical approaches. Expert Syst. Appl. **106**, 1–20 (2018)
13. Vayansky, I., Kumar, S.: Phishing-challenges and solutions. Comput. Fraud Secur. **2018**(1), 15–20 (2018)
14. Mole, P.V.: Progressive Web Apps: A Novel Way for Cross-Platform Development. Obtenido de: https://www.researchgate.net/publication/344170769 (2020)
15. Majchrzak, T.A., Biørn-Hansen, A., Grønli, T.-M.: Progressive web apps: the definite approach to cross-platform development? (2018)
16. Liu, T., Wang, H., Li, L., Bai, G., Guo, Y., Xu, G.: Dapanda: Detecting aggressive push notifications in android apps. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 66–78. IEEE (2019)
17. Pande, N., Somani, A., Samal, S.P., Kakkirala, V.: Enhanced web application and browsing performance through service-worker infusion framework. In: 2018 IEEE International Conference on Web Services (ICWS), pp. 195–202. IEEE (2018)
18. Gambhir, A., Raj, G.: Analysis of cache in service worker and performance scoring of progressive web application. In: 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), pp. 294–299. IEEE (2018)
19. Malavolta, I., Procaccianti, G., Noorland, P., Vukmirovic, P.: Assessing the impact of service workers on the energy efficiency of progressive web apps. In: 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft), pp. 35–45. IEEE (2017)
20. Developer, M.: PWA installability requirements (2022)
21. Lee, J., Kim, H., Park, J., Shin, I., Son, S.: Pride and prejudice in progressive web apps: Abusing native app-like features in web applications. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1731–1746 (2018)
22. Břoušek, P.: Evaluation and Usage of Google Progressive Web Apps Technology. PhD thesis, Masarykova univerzita, Fakulta informatiky (2017)
23. Tommasi, F., Catalano, C., Fornaro, M., Taurino, I.: Mobile session fixation attack in micropayment systems. IEEE Access **7**, 41576–41583 (2019)
24. web.dev: Manifest (2022)
25. web.dev: Serviceworkers (2022)
26. Team, G.D.: Screen copy (2022)
27. Project, T.L.: Lineageos android distribution (2016-2022)
28. IceWreck: Lokiboard-android-keylogger (2022)
29. Microsoft: Microsoft office (2022)
30. Google: Remote debug android devices (2022)
31. Ometov, A., Bezzateev, S., Mäkitalo, N., Andreev, S., Mikkonen, T., Koucheryavy, Y.: Multi-factor authentication: a survey. Cryptography **2**(1), 1 (2018)
32. Grimes, R.A.: Hacking multifactor authentication (2020)
33. Parmar, V., Sanghvi, H.A., Patel, R.H., Pandya, A.S.: A comprehensive study on passwordless authentication. In: 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), pp. 1266–1275. IEEE (2022)
34. Velásquez, I., Caro, A., Rodríguez, A.: Authentication schemes and methods: a systematic literature review. Inf. Softw. Technol. **94**, 30–37 (2018)
35. Huang, X., Xiang, Y., Chonka, A., Zhou, J., Deng, R.H.: A generic framework for three-factor authentication: preserving security and privacy in distributed systems. IEEE Trans. Parallel Distrib. Syst. **22**(8), 1390–1397 (2010)
36. Iwuoha, O., Emmanuel, N., Ekwonwune, E.: Enhancing multi-factor authentication in modern computing (2017)
37. Jover, R.P.: Security analysis of sms as a second factor of authentication. Commun. ACM **63**(12), 46–52 (2020)
38. Kaur, N., Devgan, M.: A comparative analysis of various multistep login authentication mechanisms. Int. J. Comput. Appl. **127**(9), 20–26 (2015)
39. Dmitrienko, A., Liebchen, C., Rossow, C., Sadeghi, A.-R.: On the (in) security of mobile two-factor authentication. In: International Conference on Financial Cryptography and Data Security, pp. 365–383. Springer (2014)
40. Schneier, B.: NIST is No Longer Recommending Two-Factor Authentication Using SMS. (2016)
41. Wiefling, S., Dürmuth, M., Iacono, L.L.: Verify it's you: how users perceive risk-based authentication. IEEE Secur. Priv. **19**(6), 47–57 (2021)
42. Jubur, M., Shrestha, P., Saxena, N., Prakash, J.: Bypassing push-based second factor and passwordless authentication with human-indistinguishable notifications. In: Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security, pp. 447–461 (2021)
43. Loreti, P., Bracciale, L., Caponi, A.: Push attack: binding virtual and real identities using mobile push notifications. Future Internet **10**(2), 13 (2018)
44. Eminagaoglu, M., Cini, E., Sert, G., Zor, D.: A two-factor authentication system with qr codes for web and mobile applications. In: 2014 Fifth International Conference on Emerging Security Technologies, pp. 105–112 (2014). IEEE
45. Reynolds, J., Smith, T., Reese, K., Dickinson, L., Ruoti, S., Seamons, K.: A tale of two studies: The best and worst of yubikey usability. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 872–888. IEEE (2018)
46. Meiser, G., Laperdrix, P., Stock, B.: Careful who you trust: studying the pitfalls of cross-origin communication. In: Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security, pp. 110–122 (2021)