International Journal of Information Security (2022) 21:179–189
https://doi.org/10.1007/s10207-021-00548-5

REGULAR CONTRIBUTION

Browser-in-the-Middle (BitM) attack

Franco Tommasi[1] · **Christian Catalano**[1] · **Ivan Taurino**[1]

Published online: 17 April 2021

Abstract

Man-in-the-Middle (MitM), one of the best known attacks in the world of computer security, is among the greatest concerns for professionals in the field. Main goal of MitM is to compromise confidentiality, integrity and availability of data flowing between source and destination. However, most of its many variants involve difficulties that make it not always possible. The present paper aims at modelling and describing a new method of attack, named *Browser-in-the-Middle* (BitM) which, despite the similarities with MitM in the way it controls the data flow between a client and the service it accesses, bypasses some of MitM's typical shortcomings. It could be started by phishing techniques and in some cases coupled to the well-known *Man-in-the-Browser* (MitB) attack. It will be seen how BitM expands the range of the possible attacker's actions, at the same time making them easier to implement. Among its features, the absence of the need to install malware of any kind on the victim's machine and the total control it allows the attacker are to be emphasized.

1 Introduction

One of the best known and most used attacks in the cyberspace is the *Man-in-the-Middle* (MitM) attack. The term MitM identifies a large category of attacks whose main characteristic is the ability of the attacker to place him/herself, in many different ways, in a point of the path between the victim and the accessed service. MitM attacks have been described for practically any kind of communication technology: *LTE* (Long-Term Evolution), Bluetooth, *NFC* (Near Field Communication), *IoT*, *WiFi*, *HTTPS* protocol, operating system processes, etc.[1–8].

MitM attacks aim at compromising:

- *Confidentiality*, by eavesdropping on the communication.
- *Integrity*, by intercepting the communication and modifying messages.
- *Availability*, by intercepting and destroying messages or modifying messages to cause one of the parties to end communication.

✉ Franco Tommasi
francesco.tommasi@unisalento.it

Christian Catalano
christian.catalano@unisalento.it

Ivan Taurino
ivan.taurino@unisalento.it

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

A typical scenario for this kind of attack involves: two endpoints (the victims), a third-party (the attacker) and a communication channel. The attacker can access the communication channel and can intercept and manipulate messages sent or received by both endpoints. Figure 1 sketches the basic concept.

The wide variety of existing MitM attacks bears witness to the popularity of the category. As examples of popular MitM attacks aimed at web services, MitB[3,9,10], MitM to the HTTPS Protocol[4], DNS Hijacking[11] may be mentioned.

Despite their popularity, MitM attacks are not easy to put into practice. They can be carried out either by exploiting zero-days, known vulnerabilities and weaknesses in one of the two end-points and/or in the communication channel or by getting a physical access to the latter.

While belonging to the same category, *Browser-in-the-Middle* (BitM), the attack here presented, has no such requirements. Moreover, it is relatively easy to implement and scalable through phishing techniques. It also allows real-time monitoring of the victims' behaviour during their web navigation and tampering with exchanged data.

In its essence, it is pursued by substituting the victim's browser with a malicious transparent browser, hosted on the attack platform, that the attacker is able to control in every way, leaving the victim totally unaware of the substitution. At its root is a careful use of RFB, the protocol basic to the better known VNC (Virtual Network Control) applications.

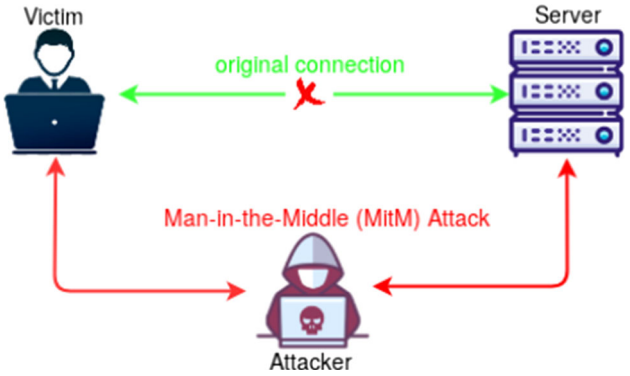

Fig. 1 Man-in-the-Middle (MitM)

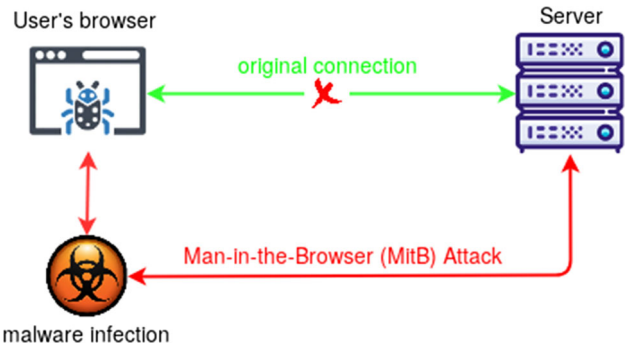

Fig. 2 Man-in-the-Browser (MitB)

2 Related work

To set the stage for the presentation of BitM, it may be useful in this section to focus attention on a family of variants of MitM, the *Man-in-the-Browser* (MitB) attack [3,9,10]. In the following section, the differences between MitB and BitM will be highlighted. In a MitB attack, a web-based Trojan malware[12,13] is injected in the victim's browser which allows to intercept any communication between the user and his/her target sites and to tamper with it. The web-based malware is programmed to become active when the user accesses specific Internet sites, e.g. online banking sites. Figure 2 sketches the idea at its base.

Once active, a MitB web-based Trojan malware is able to intercept and modify in real time all data exchanged. A number of Trojan families have been identified along the years. Zeus[14], Adrenaline, Sinowal[15], Silent Banker[16], Eurograbber[17] and EKO[18] may be quoted among them [9].

Some of them are very sophisticated and succeeded in simplifying the fraudulent activities. In fact, they can be programmed to completely automate the process beginning with the infection and ending with the money withdrawal.

To better understand the MitB attack dynamic, a brief description of a recent example of an attack engineered through the EKO malware will be provided. Initially discovered among Russian Facebook users in 2015, and later reported in France, EKO is a malware spread through the Facebook social network.

The victim receives a direct message from someone already compromised among his/her contacts. Such message contains a link pointing at a fake YouTube video. When the victim clicks on the link, a fake YouTube is presented, asking to install an extension to reproduce the video. Actually, the extension contains the EKO malware and it is installed inside the victim's browser. Once installed the extension connects to a Command & Control (C&C) server and gets from it a GitHub link from where the virus code is downloaded.

The users begin to receive undesired advertisements or, even worst, EKO is able to spy on the user's behaviour and to gather his/her personal data, home banking ones included. Eventually, the infected users become themselves a vehicle for the infection by sending malicious links to all the users in their contact list.

In short, all the MitB attacks have the target of compromising the victim browser by a Trojan malware and of intercepting, record and manipulate all communications between the user and a number of his/her contacted sites.

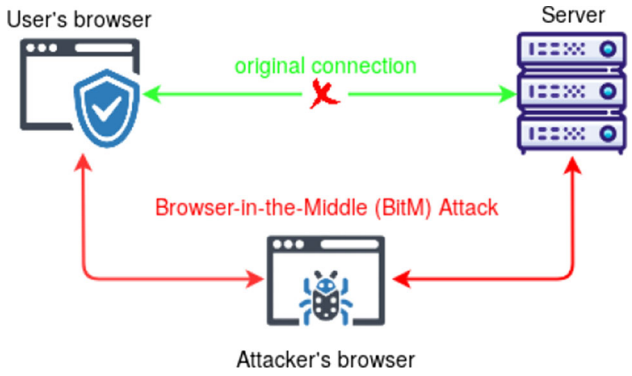

Fig. 3 Browser-in-the-Middle (BitM)

3 Browser-in-the-Middle (BitM) attack

An overview of the *Browser-in-the-Middle* (BitM) attack will be provided in the present section. The following section will provide a detailed description of the techniques used.

The idea at the base of BitM is to interpose a malicious transparent browser between the victim's browser and the web server the victim is accessing to obtain a service (e.g. a banking service, a social network, etc.). The purpose of the interposition is to intercept, to record and to manipulate any data exchange between the victim and the service provider. Figure 3 illustrates the basic concept, and Fig. 4 details a possible attack scenario.

The involved parties are:

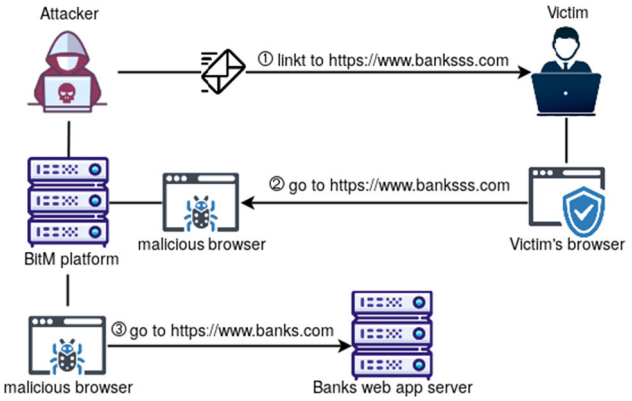

Fig. 4 Browser-in-the-Middle (BitM)

– *The Attacker*: a computer criminal aiming at violating the victim's privacy together with confidentiality, integrity and availability of the data exchanged by the two endpoints. The attacker sets up and controls a malicious server which serves a transparent web browser to the end user.
– *The Victim*: a user lured through phishing techniques[19, 20], into accessing the malicious server (hosting a transparent web browser) set up by the attacker and into using a web application.
– *The Target*: a web application able to provide sensitive and/or valuable services (e.g. banking services, mail services, etc.) accessed through an authentication mechanism.

The following steps summarize the BitM attack technique:

① The attacker sets up a forged hyperlink pointing to the malicious server. The link is sent to the victim who is lured through phishing techniques into clicking it and into authenticating in the web application;
② By clicking on the malicious hyperlink in his/her own browser, the victim will unwittingly be connected to the malicious server where a transparent web browser is residing, together with a number of programs (e.g. web proxy, sniffer, keylogger, add-ons, etc.) the attacker will use to intercept, record and manipulate the data exchanged by the victim and the target web application.
③ The malicious server will access the target web application (e.g. a banking application, a social network, etc.) through the malicious web browser a totally transparent way. At this point the victim will be able to use all the target application functionalities (plus the 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. As an example, the attacker will be able to intercept the credentials provided by the victim in some authentication form, no matter which security protocols the web application has set up. The victim will be able to navigate through all application's functionalities by a browser which, unbeknown to him/her, is connected to the malicious server.

4 BitM platform architecture and implementation

In the present section, the architecture and the implementation of the BitM platform (that is a prototypal attack platform acting as the malicious server) will be described.

The basic idea is to serve the victim a web browser encapsulating a web page, looking and behaving exactly as the desired web page of the target site. By his/her browser the victim will be able to navigate the target web application while actually unwittingly using of a transparent web browser exposed by the malicious web server.

For the attack platform, an easily customizable operating system was preferred. The malicious server was set up with a GNU/Linux distribution featuring a light window manager based on Fluxbox[21] to minimize the GUI of the window containing the malicious web browser which will be exported in the victim's web browser. The malicious web browser Chromium[22] was also customized to avoid any kind of frame.

The BitM attack platform was equipped with a VNC (Virtual Network Computing)[23] server (*vncserver* in our case, [24]). VNC is a generic name for a type of programs allowing the remote control of the GUI of a computer. Such programs are based on the use of the RFB (Remote Frame Buffer) protocol. The RFB protocol states the rules by which a client (the controlling machine) and a server (the controlled machine) can establish and manage a remote control session through a network[25].

Another important part of the picture is *noVNC*[26].

noVNC is a piece of software essentially made of two parts:

– *The client part*, which is downloaded through the malicious link and is executed by the victim browser automatically without user interactions (it requires just a browser with HTML5 support; therefore, it works on all major browsers);
– *The server part*, which must be installed on the attack platform.

The client part is made of a collection of JavaScript which interpret the HTML5 code coming from the server, open a WebSocket [27] channel to the server and send VNC binary requests through it.

The server part takes care of serving the required HTML5 code and runs *Websockify* (formerly named *wsproxy*).[28]

Websockify is a piece of software acting as an intermediary between the WebSocket channel and the actual VNC TCP socket on the machine. It begins by accepting the WebSocket handshake sent by the *noVNC* client, and from then on it translates the VNC traffic, keeping the VNC client and server in touch.

To allow the victim the access to the BitM attack platform through a web browser and to serve it the JavaScript and the HTML5 code, a minimal web server is obviously needed on the server (*Python BaseHTTPServer caldav*[29]). The served HTML5 code exploits canvas tag features in order to display remote screen content and handle keyboard/mouse events.

All in all, the above collection of software allows the victim browser to access the malicious browser running on the BitM platform. In other words, a VNC connection via a JavaScript/HTML5-based VNC viewer that does not require any additional software for the victim is established from his/her browser which allows him/her to see the malicious browser as if it were the target site without realizing, it is in fact a "browser inside a browser".

It is worth emphasizing that the proposed architecture only includes open-source technologies.

Figure 5 outlines the BitM attack platform blocks and their logical connection.

For the sake of clarity, let us concisely review the steps involved in the attack:

① The victim accesses the server's malicious URL.
② The malicious server, contacted at the malicious URL, sends JavaScript code to the client (this is what we call "the noVNC client").
③ The victim's JavaScript code starts (through the HTTP channel) a WebSocket connection.
④ The malicious web server accepts the client's request of WebSocket connection and turns the WebSocket traffic to Websockify.
⑤ The victim's JavaScript code starts a VNC dialogue (which travels on the WebSocket connection) with a VNC server on the attacker's platform.
⑥ Websockify on the attacker's platform forwards the VNC requests coming from the WebSocket channel to the actual VNC server TCP socket on the same platform.
⑦ The server sends back the VNC response through the WebSocket channel, together with HTML5 code (including the <canvas> tag). With the help of the client's JavaScript, the web server exposes a web page set up to encapsulate the page at the URL the victim originally tried to access and therefore being the legitimate one (and looking exactly the same).
⑧ The victim will now enter the credentials in such page, allowing the attacker to intercept the traffic and to steal them. At the same time it will use them in the legitimate site to allow the victim's access to it and to continue the eavesdropping (and, possibly, the manipulation of the traffic).

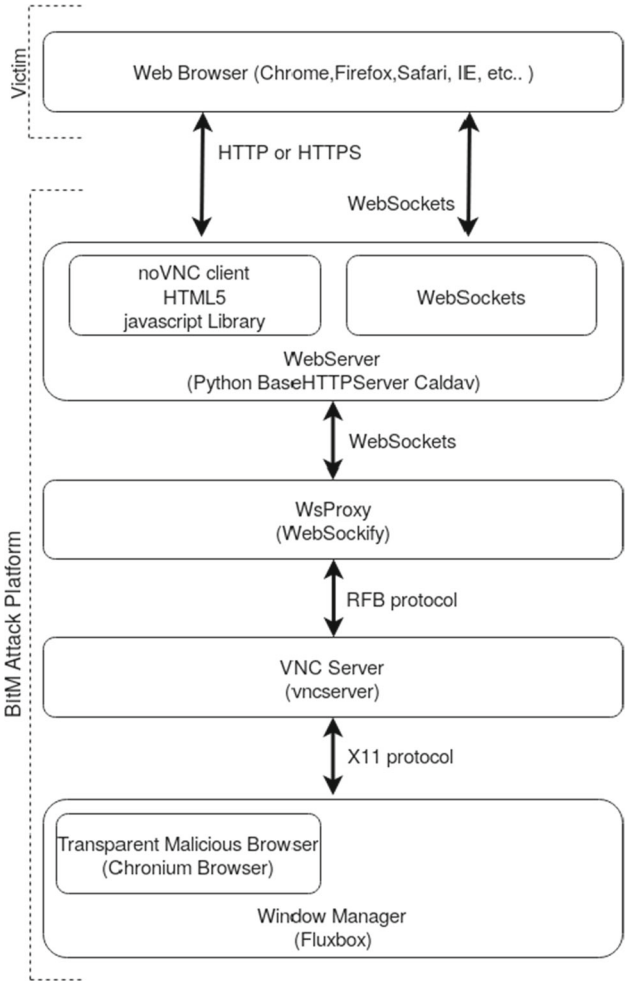

Fig. 5 Attacker's platform

5 Possible BitM attacks

By controlling the malicious server and the noVNC client served to the victim, the attacker can now start a variety of attacks. For the sake of simplicity, the attack strategies will be divided in two categories, *Passive BitM* and *Active BitM*:

– *Passive BitM*: this category includes all techniques exclusively aimed at data gathering (obviously the recorded data can be freely used later). The attacker can in fact closely examine in real time all the victim's actions during the navigation in the target web app and record the entire session. A keylogger[30] can easily be positioned in the malicious web browser to record keystrokes (e.g. for usernames, IDs and passwords). By a web-proxy[31] or suitable add-ons installed in the malicious

web browser, all the traffic (e.g. session cookies) can also be recorded.

– *Active BitM*: this category includes all techniques used to actively modify the traffic exchanged between the victim and the target web application. What the victim is supposed to see in the original target can be altered in many ways by the malicious server which is actually serving the requested page. Moreover, by the web proxy and the add-ons, all types of web-based malware exploiting known vulnerabilities for the victim's browser can be injected into it.

According to the attack goals and to the selected victim, the attacker may choose to install malicious software either on the *client side* or on the *server side*.

– *BitM client-side*: in this case the attacker may insert malicious JavaScript code or web-based malware[32] in the web page served to the victim (the end user). By such an approach, for example, an attack might be arranged along the lines of the Man-in-the-Browser (MitB) attack described in the "Related Work" section: the victim might be lead to download all kinds of malware to be installed in the web browser or, more simply, to deceptively use a Browser Exploitation Framework (BeeF)[33,34]. In practice all kinds of client-side attacks might be implemented.
– *BitM server-side*: in this case the attacker may install malicious software in the server. For example, keylogging Chrome add-ons *(e.g. Fea KeyLogger)*[35] might be used. Also add-ons able to execute JavaScript code on the browser side might be installed, in order to modify the input on the fly *(e.g. Violentmonkey)*[36]. It would also be possible to install on the malicious server a web proxy *(e.g. Burp proxy)*[37] to sniff all the traffic generated towards the target by the malicious BitM platform and to redirect when required.

To sum up, since this attack technique gives the attacker full control both over the malicious server and over the HTML and JavaScript pages served to the victim, once the victim has become entangled, the types of possible exploits are only limited by the attacker creativity.

It is worth noting the attack here described is scalable, easily replicable and hardly detected by a common user. By it criminals will no more need to mimic the target web applications. They will simply need to add software and functionalities to the malicious server, by which they will be able to directly exploit the services offered by the targets.

Figure 6 shows the possible insertions of malicious software the attacker may execute to steal useful information from the victim.

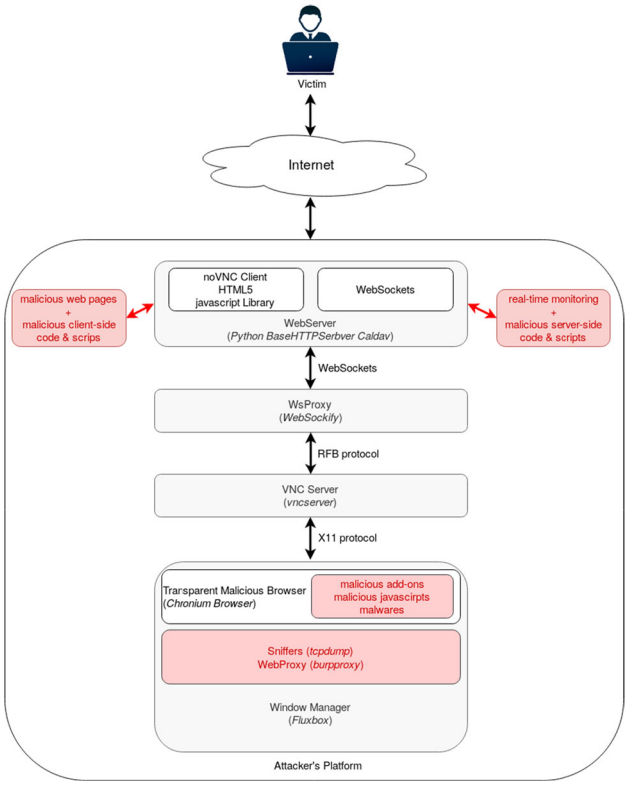

Fig. 6 Possible malicious software to profit from a BitM attack

6 BitM vs MitM

It may be worth stressing the point discussed in the following. In principle, Browser in the Middle (BitM) and Man in the Middle (MitM) attacks are similar in that they control the data flow between the victim's computer and the target service. However, a clear distinction must be drawn between the two attacks: although both may be started by a phishing technique, a MitM attack uses a proxy server which relays traffic and places itself at the application layer between the victim's browser and the legitimate target service. To make this type of attack possible, some sort of malware must be run on the victim's computer. On the contrary, a BitM attack needs no such software. The victim navigates with his/her browser as usual but, in fact, he/she is unknowingly running a remote browser transparently displayed into the browser's window, so that he/she sees exactly what expected. It is as if, so to speak, he/she were sitting in front of the attacker's computer using the attacker's keyboard. At this point, with little effort, the attacker can capture, record, alter the data exchange between the victim and the service provider. Figure 3 illustrates the basic concept.

7 BitM experimental results

In this section the results of an experiment executed by the *Telegram* web application[38] (desktop version) will be reported.

The Telegram web application was selected for the experiment because it allows authentication only by a login code sent through SMS or Telegram message to the mobile device (a sort of two factors authentication).

The same experiment may be tried with any web application requiring some form of two factors authentication (Gmail, Facebook, web banking, etc.). The scenario is the one described in the "Browser-In-The-Middle (BitM) Attack" section.

The testbed is set up as follows:

- *Victim's PC*: through social engineering techniques (e.g. by phishing techniques) the victim is led to access the Telegram web platform by the browser on a desktop computer. The victim workstation's IP address is in our case 10.0.12.32.
- *Attacker's platform*: the attack platform is set up and equipped with the software described in Sect. 4. Its IP address is 10.0.12.173. While for convenience the platform is physically located in the same subnet the victim's, it can be located everywhere, without any loss of the functionalities here described, as long the victim can access it.
- *Web application target*: the Telegram web application, accessible through a normal browser, has been selected as the target. It is assumed the victim owns an active account within the selected application. Although Telegram was selected on account of being a very popular application and allowing fast and easy tests, it must be stressed the method here introduced applies exactly in the same way to any web application based on a two-factor authentication procedure (Gmail, Facebook, etc.).

As highlighted in Fig. 7, an user receives a malicious link by any kind of phishing technique. Admittedly, using the Telegram application to send and receive the malicious link for the same (Telegram) web application, makes little sense. The choice was made anyway for two reasons:

- the way the fake link is received is irrelevant and using Telegram for that purpose does not imply any loss of generality: the attack conceptual framework is totally unaffected in case the link is sent by mail or by any type of messaging platform (WhatsApp, Facebook Messenger, etc.);
- the use of Telegram to send and receive the malicious link allowed to speed up the testing procedures.

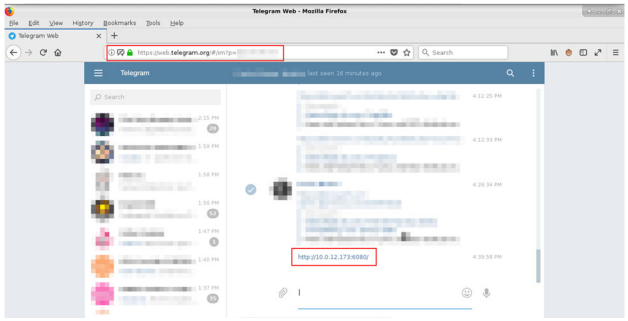

Fig. 7 Malicious link

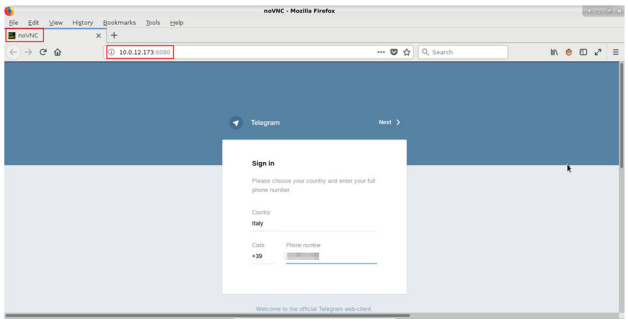

Fig. 8 Telegram authentication page step 1

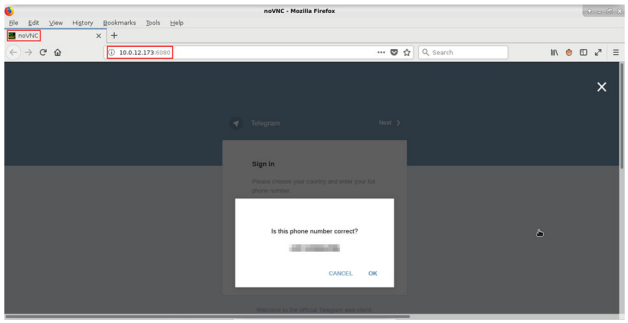

Fig. 9 Telegram authentication page step 2

After clicking on the malicious link, the victim is redirected to the attacker platform http://10.0.12.173:6080, which exposes the transparent browser and starts the web Telegram authentication process, as shown in Figs. 8 and 9.

At this point, the victim receives on the open Telegram application (wherever it runs) the login code from Telegram and completes the authentication. After that the victim is allowed the access to his/her account.

As demonstrated in Fig. 10, the victim is in fact visualizing a web page shown by the attacker platform. The word "noVNC" has intentionally been left as the window's title to demonstrate the victim is actually seeing an HTML5 canvas (containing the transparent browser set up by the attacker) and everything the victim types in is captured by the attacker.

As usual, the password provided by the victim is hidden from the view of the observer; therefore, a simple attack based

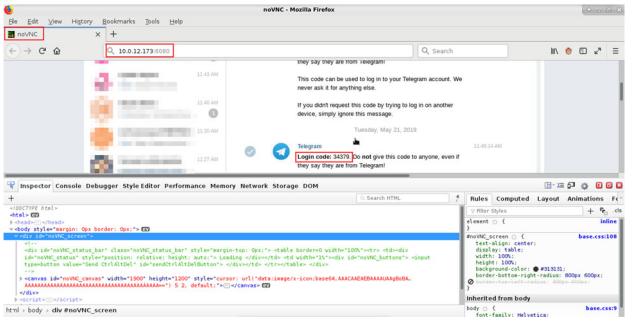

Fig. 10 Victim is visualizing a web page shown by the attacker platform

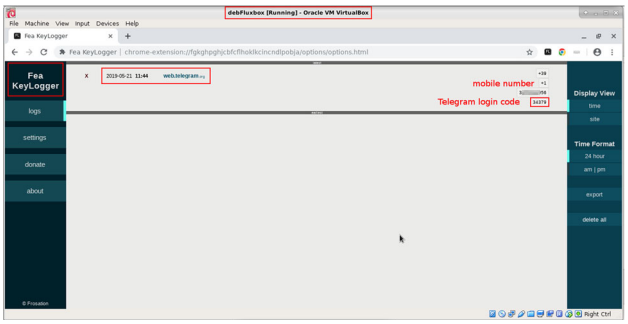

Fig. 11 Attacker is capturing every activity of the victim on his/her machine

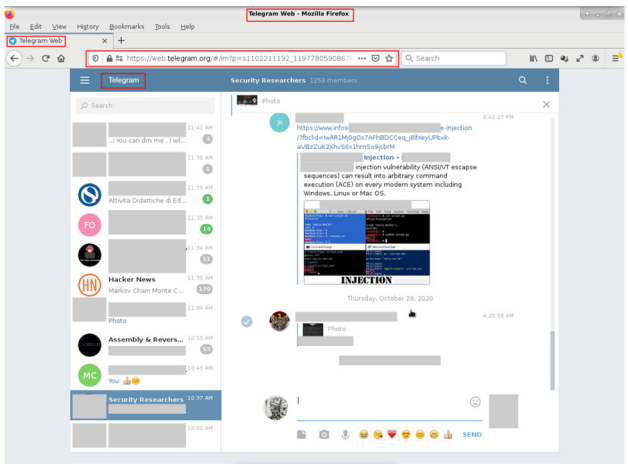

Fig. 12 User is visualizing Telegram web page

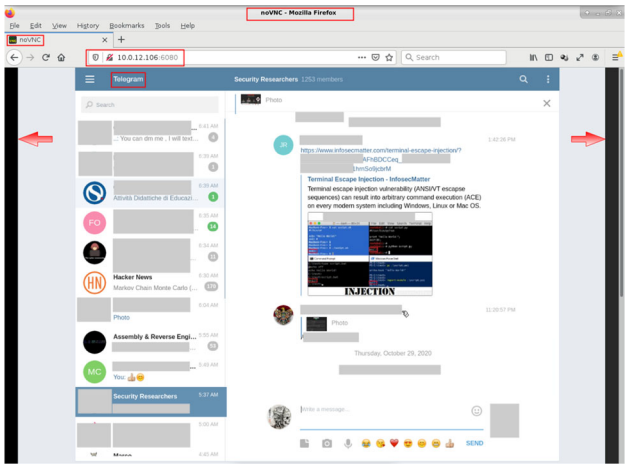

Fig. 13 User is visualizing Telegram web page shown by the attacker platform

on watching in real time the typed characters and taking note of the inserted password is not possible. However, a simple workaround is the installation of a keylogger on the attacker browser (a Chrome add-on in the test here described [32]). By that, anything the victim types (passwords, PINs, etc.) can be captured and securely stored.

Figure 11 demonstrates how the attacker succeeds in seeing and capturing every activity of the victim on his/her machine.

By controlling the malicious server and the noVNC client served to the victim, the attacker can now start any of the attacks listed in Sect. 5. In the experiment here described, a Passive BitM attack was carried on by installing a Chrome Keylogger Extension on the attacker server.

As the simple example here documented demonstrates, it was possible to carry on a successful attack without exploiting zero-day or any other known vulnerability at the two end-points (the victim PC and the Telegram web application) or in the communication channel. Also the attack was entirely conducted by a remote location, simply by an improper use of known technologies.

7.1 Genuine telegram vs fake telegram

To underline the potential of the attack, the true Telegram web page and the fake Telegram page served by the attacker are shown together in Figs. 12 and 13. An easily removable visual clue (the two black sidebars) has been intentionally left in the fake page to denote its spurious nature. The differences between the two are indicated by their inclusion in red rectangles. They are not easy to spot for an inexpert user (e.g. an IP address is shown in the address field, instead of *web.telegram.org*). Moreover, a few, probably most, of them can be fixed.

Another view of the differences between the two pages can be obtained by a browser's developer tools (Figs. 14 and 15 show them in Firefox). Although the differences are now plainly visible (note for example the canvas tag in figure 15), they are likely to be noticed only by the most experienced users (as most users are not even aware of the existence of such tools).

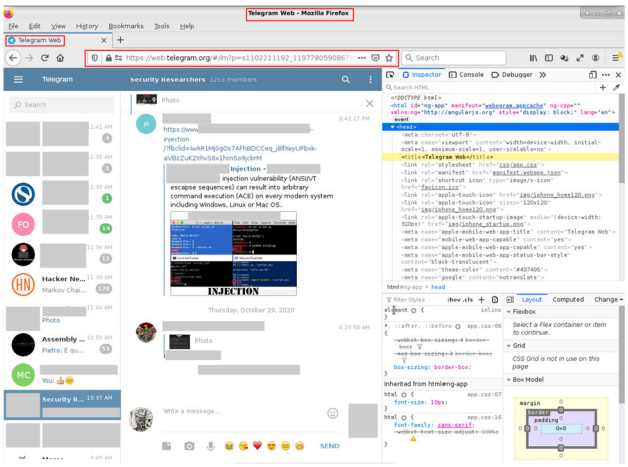

Fig. 14 User is visualizing Telegram web page with Firefox dev tool

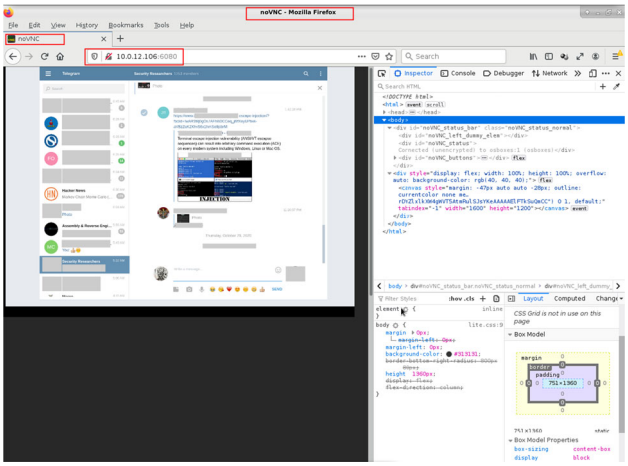

Fig. 15 User is visualizing Telegram web page shown by the attacker platform with Firefox dev tool

8 MitB and BitM

As pointed out in a previous section, *Man-in-the-Browser* type attacks (MitB)[3,9,10] are a form of Internet threat related to the well-known *Man-in-the-Middle* (MitM) attack. It combines the use of phishing approaches[39,40] with a Trojan horse technology to modify the web pages, the transactions' content and to insert additional transactions. Everything is done with both the victim and the host web application left totally unaware of the exploit.

Conversely, the BitM attack here described is conducted having the victim connecting to an attack platform hosting a browser. The browser can be set up to modify the visited web pages, the transactions content and to insert new additional transactions. In this case, however, there is no need to identify vulnerabilities of any kind to launch the attack. Also, in this case, the malicious browser is totally transparent both for the victim and the host web application.

It should be clear that the two techniques are not mutually exclusive and that they could be combined to enhance the attack's chances of success. In the following, a possible way of interaction between the two will be described. A BitM attack may be arranged in order to exactly identify the victim's browser type (e.g. the browser name and its version). Once a malware which is known to affect that particular version of the browser is found, the same BitM is used to upload the malware on the victim's host. After that, the usual MitB attack is carried on, with both the victim and the host web application unaware of it.

The definition of APT (Advanced Persistent Threat)[41] refers to "an entity that engages in a malicious, organized, and highly sophisticated long-term or reiterated network intrusion and exploitation operations to obtain information from a target organization, sabotage its operations, or both". With such description in mind, one can definitely state that a BitM infrastructure may be part of a set of tools used to mount an APT. A possible attack scenario might be the following: an entity is able to exploit some vulnerability of a specific browser or webkit used by a few browsers, which, as an example, allows to compromise the DNS cache. When a user tries to navigate to a specific URL, it might be automatically and constantly readdressed to a BitM attack platform which might harvest users' credentials. In such sense, it could be argued that a BitM attack backed by an MitB attack is in fact an APT.

9 Limits and constraints

It is most appropriate to state here clearly that the purpose of the present paper is to provide a PoC (Proof of Concept) for the BitM attack and not to design, describe and implement a weaponized attack BitM infrastructure/platform. Also, it can be easily seen that, as it is proposed here, the attack is mainly effective when addressed to a specific target. Although the attack is indeed scalable, for its use on a larger scale a deeper analysis of a few potential problems is advisable.

In the evaluation of its limits and constraints, the following points might be considered:

– *High Network Latency*: while an high network latency may potentially interfere with any kind of attack, it may be argued BitM is especially damaged by such an unfavourable condition, mainly because of the larger amount of data it needs to exchange to be carried on. To avoid the risk of launching an unsuccessful attack, the intruder might therefore silently gauge beforehand the connection speed and redirect the victim to the authentic web site when the speed is reckoned to be insufficient. It also worth considering a few studies aimed at improving

performances of VNC servers in high latency networks [42].
- *Device compatibility*: Problems could also arise from the heterogeneity of the victims' devices (e.g. the server might set up the attack assuming the victim is using a desktop computer browser while the victim is actually using a mobile device browser). Again, to minimize the risk of launching an unsuccessful attack, the intruder should check the victim's browser before launching the attack.
- *Scalability*: The attacker must beforehand decide which is the legitimate site whose credentials wishes to harvest. After that, a suitably prepared link is spread by phishing techniques. In this sense the attack is definitely scalable. That is, as usual, a percentage of the contacted users will follow the link and possibly become victims of the attack, thus leading, in the most basic example, to a massive harvesting of credentials. Because BitM allows many users to be connected to the attacker's platform at the same time, it is indeed a scalable attack. For that very reason, it must be made clear its scalability is closely related to the scalability of the attacker's platform (web and VNC servers, both hardware and software) and to the bandwidth of its connection. This said, it must be made clear the present study did not actually experiment the scalability of the attack (not an easy task anyway, for the obviously dangerous legal implications): once again, what the present paper wishes to propose is a proof of concept, that is the demonstration that the described attack is perfectly possible.

10 Ethics

In conducting the present research, we followed the guidelines of the ethics committee of the University of Salento. All experiments have been performed against our own accounts, and no other user of the involved platforms was involved in our attacks. Even though the attack technique described in this paper is completely general and not restricted to a specific web service, web application or client, according to the principle of responsible disclosure, we have reported a description of our attack and a notice about its effectiveness to the security centre of the main browsers' vendors, to the Telegram messaging service security centre, and to the CERT Coordination Center.

11 Countermeasures

The present study shows not only username/password authentication procedures are easily overcome but also some kind of two-factor authentication procedures (e.g. the usual credentials plus a code received by any other means, but only when the code is required just for the first access, e.g. in Telegram) are easily attacked through the method here described.

From the user's point of view, that is from the client side, the best practice to avoid such attacks is to put extreme care in identifying the target web page, by one of the many ways systems use to allow a preview of the address before clicking on an URL (e.g. many systems allow reading when hovering over a link with the mouse pointer) and, after that, carefully checking the URL when it appears in the address field of the browser.

On the server-side (i.e. the side of the exposed web applications) an established method, especially in the banking field, is two-step authentication with a mobile app developed on purpose by the service provider. The app is freely made available to the user, who must prove, with one of many one-time methods, his/her identity to be able to fully configure it on a mobile device. Once the app is installed, the user may access the application web site and, after a simple username/password authentication procedure the mobile application receives a confirmation request to be accepted through the mobile device, in one of many possible ways (Touch ID, Face ID, another password, etc.) Since the verification is sent through a different channel, the attacker cannot capture it and the entire transaction is secured. The only drawback of such defence is the need of a special mobile app for each needed web service. It is therefore very likely some of the services needed by each user will remain without protection.

It is the opinion of the authors that the best deterrent against attacks like the one this paper describes is a feature all browsers' developers should implement: the notification of the use of WebSocket by a web site or application and the possibility for a user to choose whether to continue the navigation or to stop it. Blocking the setup of a WebSocket channel would in fact prevent the operation of the noVNC client, thus thwarting the entire attack. Obviously a warning would not prevent the victim from authorizing the set up. Nonetheless, it could contribute to raise the his/her level of attention.

12 Future developments

The above described technique is subject to further refinements. One of the possible developments is the exposure to the view of the victim (the user of a mobile device) of a mobile application instead of the transparent browser. Such a variation may be named *Mobile-Application-in-the-Middle* (MAitM) attack.

The desired effect is that of mimicking the usual opening of some mobile apps after clicking on a link in a mobile browser (YouTube, Amazon Mobile, etc.) The attack would

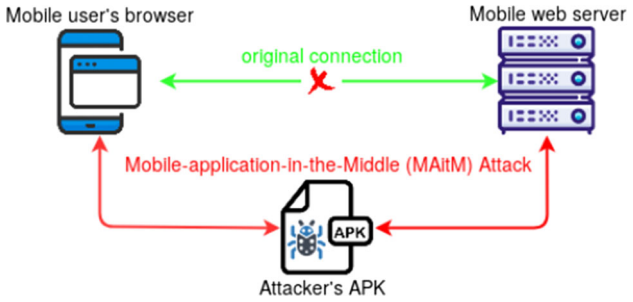

Fig. 16 Mobile-Application-in-the-Middle (MAitM)

be based on serving the victim with a web page pointing to the target site and disguised as a mobile application. The victim would then experience the familiar GUI of a mobile app.

To achieve the desired effect above described, the attack platform could be set up in two alternative and equivalent ways:

- *mobile platform attack*: a mobile device is used (e.g. an Android device) where to install and customize all the technological stack described in Sect. 4.
- *apk platform attack*: the platform described in Sect. 4 could be enhanced to run *.apk* apps [43].

To test the technique, the second method has been used. The malicious server was set up to execute *.apk* applications and, in the same way described for the basic form of the attack, to lure the victim into unwittingly executing the application within the browser of his/her mobile device.

Figure 16 illustrates the basic concept.

It could be implemented by setting up a mobile device as the malicious server and by customizing the Android system to install the mobile application to be exported in the victim's web browser.

A number of laboratory tests have been carried on by the authors with *Anbox*[44]. Anbox is a fairly new tools that acts as a layer between Linux distribution and native Android apps. It allows to use many apps as though they were running natively on your machine. While Anbox is still in the early stages of development, it has been still possible to set it up well enough to prove the concept. Further experiments are in progress to better evaluate the actual impact of such a technique on the users.

13 Conclusions

The present paper aimed at demonstrating how common two-step authentication mechanisms implemented by the overwhelming majority of web applications can easily be overcome by the BitM attack here described and at providing cyber defence professionals with new insights and incident analysis tools.

It has also been shown how combining BitM and MitB attack techniques can be used to attack a user not sufficiently aware of the dangers entailed by web navigation.

A further variation, the MAitM attack, has been hinted. The authors plan to thoroughly investigate in the near future.

Finally, an easy-to-implement countermeasure has been proposed which might help the users to better detect malicious sites.

Funding Open access funding provided by Università del Salento within the CRUI-CARE Agreement.

Declarations

Human participants and/or Animals In this research, we have designed cyber attacks on personal banking websites, Telegram and Google account. These experiments have been performed against the author's bank accounts. All the experiments were performed by intercepting confidential information belonging exclusively to the authors of the paper. This research does not involve human participants and/or animals.

References

1. Conti, M., Dragoni, N., Lesyk, V.: A survey of man in the middle attacks. IEEE Commun. Surv. Tutor. **18**(3), 2027–2051 (2016)
2. Mallik, A., Ahsan, A., Shahadat, M., Tsou, J.: Man-in-the-middle-attack: understanding in simple words. Int. J. Data Netw. Sci. **3**(2), 77–92 (2019)
3. Dougan, T., Curran, K.: Man in the browser attacks. Int. J. Ambient Comput. Intell. (IJACI) **4**(1), 29–39 (2012)
4. Callegati, F., Cerroni, W., Ramilli, M.: Man-in-the-middle attack to the HTTPS protocol. IEEE Secur. Priv. **7**(1), 78–81 (2009)
5. Sun, D.Z., Mu, Y., Susilo, W.: Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard V5. 0 and its countermeasure. Pers. Ubiquitous Comput. **22**(1), 55–67 (2018)
6. Rupprecht, D., Kohls, K., Holz, T., Pöpper, C.: Breaking LTE on layer two. In IEEE Symposium on Security and Privacy (SP) (2019)
7. Navas, R.E., Le Bouder, H., Cuppens, N., Cuppens, F., Papadopoulos, G.Z.: Do not trust your neighbors! A small IoT platform illustrating a man-in-the-middle attack. In international conference on Ad-Hoc networks and wireless (pp. 120-125). Springer, Cham (2018)
8. Bui, T., Rao, S.P., Antikainen, M., Bojan, V.M., Aura, T.: Man-in-the-machine: exploiting ill-secured communication inside the

computer. In 27th USENIX security symposium (USENIX Security 18) (pp. 1511-1525) (2018)
9. Ayyagari, K.S.A.: Man in the browser attacks (2017)
10. Rauti, S., Leppänen, V.: Browser extension-based man-in-the-browser attacks against Ajax applications with countermeasures. In proceedings of the 13th international conference on computer systems and technologies (pp. 251-258). ACM (2012)
11. Marrison, C.: Understanding the threats to DNS and how to secure it. Netw. Secur. **2015**(10), 8–10 (2015)
12. Chang, J., Venkatasubramanian, K.K., West, A.G., Lee, I.: Analyzing and defending against web-based malware. ACM Comput. Surv. (CSUR) **45**(4), 49 (2013)
13. Provos, N., McNamee, D., Mavrommatis, P., Wang, K., Modadugu, N.: The ghost in the browser: analysis of web-based malware. HotBots **7**, 4 (2007)
14. Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., Wang, L.: On the analysis of the zeus botnet crimeware toolkit. In 2010 eighth international conference on privacy, security and trust (pp. 31-38). IEEE (2010)
15. Shields, M.: Trojan virus steals banking info. http://news.bbc.co.uk/2/hi/technology/7701227.stm
16. Al-hamami, A.H., Najadat, F.A.O., Wahhab, M.S.A.: Web application security of money transfer systems. J. Emerg. Trends Comput. Inf. Sci. **3**(3), 365–372 (2012)
17. Kalige, E., Burkey, D., Director, I.P.S.: A case study of eurograbber: How 36 million euros was stolen via malware. Versafe (White paper) **35**, (2012)
18. Umawing, J.: Malware Eko affecting French Facebook users. https://blog.malwarebytes.com/cybercrime/2016/10/malware-eko-affecting-french-facebook-users/ (2019)
19. Chiew, K.L., Yong, K.S.C., Tan, C.L.: A survey of phishing attacks: their types, vectors and technical approaches. Exp. Syst. Appl. **106**, 1–20 (2018)
20. Vayansky, I., Kumar, S.: Phishing-challenges and solutions. Comput. Fraud Secur. **2018**(1), 15–20 (2018)
21. Kinnunen, H.: Windowmanager for *nix Operation Systems http://fluxbox.org/
22. https://www.chromium.org/Home
23. Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual network computing. IEEE Internet Comput. **2**(1), 33–38 (1998)
24. https://www.thegeekdiary.com/linux-os-service-vncserver/
25. Richardson, T., Wood, K.R.: The rfb protocol. ORL, Cambridge (1998)
26. noVNC Project. https://novnc.com/info.html
27. https://tools.ietf.org/html/rfc6455
28. https://github.com/novnc/websockify
29. https://github.com/python-caldav/caldav/blob/master/tests/proxy.py
30. Rahim, R., Nurdiyanto, H., Abdullah, D., Hartama, D., Napitupulu, D.: Keylogger application to monitoring users activity with exact string matching algorithm. J. Phys. Conf. Series **954**(1), 012008 (2018)
31. Moshchuk, A., Bragin, T., Deville, D., Gribble, S.D., Levy, H.M.: SpyProxy: Execution-based Detection of Malicious Web Content. In USENIX security symposium (pp. 1-16) (2007)
32. Wang, J.: Detection and analysis of web-based malware and vulnerability (Doctoral dissertation) (2018)
33. Alcorn, W.: Beef-the browser exploitation framework project (2013)
34. Sawant, H., Agaga, S.: Web browser attack using BeEF framework
35. https://chrome.google.com/webstore/detail/fea-keylogger/fgkghpghjcbfcflhoklkcincndlpobja
36. https://chrome.google.com/webstore/detail/violentmonkey/jinjaccalgkegednnccohejagnlnfdag
37. https://portswigger.net/burp
38. https://web.telegram.org
39. Utakrit, N.: Review of browser extensions, a man-in-the-browser phishing techniques targeting bank customers. In proceedings of the 7th Australian information security management conference (2010)
40. Cain, C.: Analyzing Man-in-the-Browser (MITB) Attacks. SANS Institute, Bethesda (2014)
41. 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)
42. Taylor, C., Pasquale, J.: Improving video performance in VNC under high latency conditions. In: 2010 international symposium on collaborative technologies and systems (pp. 26-35). IEEE (2010)
43. https://androidpicks.com/explain-apk/
44. https://anbox.io/