A Survey on Countermeasures against Man-in-the-browser Attacks

Sampsa Rauti

University of Turku, Finland
sampsa.rauti@utu.fi

Abstract. Man-in-the-browser (MitB) attacks can modify the contents of a web page or alter data in messages exchanged over the network without the communicating parties (the user and the web service) noticing anything out of ordinary. In this paper, we present a systematic survey of countermeasures against man-in-the-browser attacks. While no countermeasure seems to be completely foolproof (and still usable) against these attacks, combining a set of solutions and more effectively enforcing them in real-world systems should greatly mitigate this threat in the future.

Keywords: Man-in-the-browser attacks, Web browser security, Malicious browser extensions

1 Introduction

The man-in-the-browser attack was first discussed in 2005 by Augusto Paes de Barros in his presentation about emerging backdoor trends. Philipp Gühring later coined the name man-in-the-browser and provided a more detailed description of the threat and its countermeasures [13]. Now, over 10 years later, man-in-the-browser attacks are still a serious threat for many web-based services. For example, many security experts have stated that this type of malware is the most serious threat against online banking.

In man-in-the-browser attacks, the malware can modify the contents of the displayed web page or alter data in HTTP requests and responses without the communicating parties (the user and the web service) noticing anything suspicious. Real-life examples of man-in-the-browser malware include for instance Zeus, SpyEye, URLZone, Torpig and Silentbanker [7, 8].

This study surveys the literature related to countermeasures against man-in-the-browser attacks. In doing so, it can be though as both an extension and an update to the report written by Gühring back in 2006. To the best of our knowledge, this paper is also the first systematic survey on man-in-the-browser attacks and countermeasures proposed against them.

The rest of the paper is structured as follows. Section 2 gives a more detailed description of the man-in-the-browser attack. Section 3 explains the method of the study. Section 4 presents the results of the survey: the proposed countermeasures against man-in-the-browser attacks found in different literature sources are explained and their benefits and drawbacks are assessed. Finally, Section 5 contains the discussion and Section 6 concludes the paper.

2 Man-in-the-browser attacks

Man-in-the-browser (MitB) is a type of software security threat that infects a web browser by exploiting the security holes in the browser. The malware can make changes to web pages before they are displayed to the user, modify the contents of incoming and outgoing communications, and issue additional HTTP requests [13]. MitB malware can also steal data and send it back to the command and control server [31]. All this functionality happens stealthily without the user, the targeted web application or the server noticing anything suspicious. The man-in-the-browser malware can be though of as a dishonest proxy that operates between the client and the server without these parties observing anything out of the ordinary. In this sense, MitB is a type of man-in-the-middle attack, where the adversary manages to capture and possibly modify the messages between two communicating parties (e.g. a client and a server). However, instead of intercepting messages in the network, "the middle" is now located at the endpoint, that is, the malware resides on the infected client machine.

A MitB attack can proceeds as follows:

1. The malware infects the user's computer. It may infect the browser (e.g. as a malicious browser extension) or operate as a malicious program separate from the browser.
2. The malware waits for the user to navigate to a URL that is on the list of web pages to be targeted.
3. The malware then monitors the user's actions, and when the user has logged into a service and initiates a transaction (e.g. transfers money from a bank account), the malware intercepts this request and modifies the values (e.g. alters the receiver's bank account number through the DOM interface of the browser).
4. After modification of submitted values, the malware allows the browser to continue with the operation as usual.
5. The browser sends the request with modified values to the server. The server accepts the request, because it has no way of knowing this request is not what the user had originally intended to send in.
6. The server then sends an acknowledgement of a successful transaction to the browser. For example, in many online banking applications, a confirmation of the details of a bank transfer is shown to the user.
7. The malicious program or browser extension scans the displayed HTML page, and changes any details (e.g. the receiver's bank account number) to correspond to the original request made by the user. This way, the user has no way of knowing that the transaction has not gone as intended.

The involved parties (e.g. the user and the bank) have been deceived, and when they learn that the transaction by the user was in fact modified, it is usually too late to do anything about it.

It is noteworthy that many of the traditional authentication mechanisms do not work against man-in-the-browser attack. The attack modifies the web page or HTTP requests after authentication has taken place. Thus, the attack takes place on a level different from authentication mechanisms and simply bypasses them. Traditional authentication mechanisms such as username-password pairs, one time pad tokens or smartcards do not protect from MitB. Multifactor authentication is also easily bypassed in many cases.

The same is true for many traditional security mechanisms such as encryption and firewalls. For example, TLS encryption takes place on the transport layer, but the malware operates in the browser, where it can get access to unencrypted data – before it is encrypted and is sent over the network (or, in the case of incoming data, after it is received and decrypted). Stealthiness and immunity to almost all traditional security measures is what makes man-in-the-browser attacks so dangerous.

There are several methods to implement the man-in the browser attack. Browser extensions (or user scripts) can be used to modify web pages and values in their data fields (by using Document Object Model API), grab information in the forms, and to intercept and alter HTTP request and responses [19, 32]. The web page and data can also be edited on lower level, for instance by setting up hooks in the network libraries [26]. In some cases, man-in-the-browser attacks can also be carried out by poisoning the browser cache [14]. Some countermeasures discussed in Section 4 work for some of these implementations of MitB attacks, and some of them work in all cases.

3 The Method of the Study

The study was implemented as a systematic survey. Relevant papers were searched from the Scopus database. We used the search term "man-in-the-browser" to search from publication titles and abstracts. Also, the all the publications in the references of these papers were included in the initial set of publications. This helped us to get a more extensive collection of related papers and also to include a few relevant white papers from software security vendors that would have otherwise been missed.

The papers were then assessed first based on their titles, then based on the abstract and finally based on the full paper. Papers that presented either practical or conceptual countermeasures against man-in-the-browser attacks were included in the study. Papers that did not discuss solutions to thwart MitB attacks were excluded.

4 Results: Countermeasures against MitB

Monitoring browser extensions Traditionally, web browser extensions, used to add extra functionality to the browser, have been a very powerful and stealthy way to implement man-in-the-browser functionality. While modern browsers (such as Chrome) have recently been trying to add mechanisms to enforce more

fine grained extension permissions and restrict what browser extensions are allowed to do, malicious extensions are still a significant problem for privacy and security today.

Ter Louw et al. [27] use extension integrity checking and discuss monitoring the runtime behavior of extensions to solve the problem. Marouf and Shehab [17] present a framework for browser extension permission management in Chrome, while Wang et al. discuss access-control of extensions for Firefox [33]. Guha et al. [12] propose a framework for specifying fine-grained access control and dataflow policies for extensions. Liu et al. [16] propose micro-privilege management for extensions and different sensitive levels for DOM (Document Object Model) elements on a web page.

Restricting the power of extensions is an effective countermeasure for man-in-the-browser attacks that have been implemented by the means of browser extensions, and simple policies such as turning the extensions off completely on certain critical web pages such as online banks can be very successful in fraud prevention. However, monitoring extensions and restricting their permissions does not help of the man-in-the-browser malware has been implemented with some other method such as setting up hooks in the network libraries.

Out-of-band verification One of the most effective ways to prevent man-in-the-browser attacks is out-of-band (OOB) verification. In other words, a second channel separate from the connection between the client and server is created and the intended transaction (such as money transfer) is verified using this secure channel. For example, the user can receive a verification message via SMS and can verify that the transaction has not been tampered with [8, 35]. However, NIST (the US National Institute of Standards and Technology) has stated in 2016 that it no longer considers this practice safe: *"SMS messages may be intercepted or redirected, implementers of new systems should carefully consider alternative authenticators."* [29].

Authentication or verification using OOB push notifications is considered a safer practice. When the user authenticates or verifies a transaction, a push notification is sent to the user's mobile phone. The user approves the login process or the transaction using an app installed on his or her mobile device. In this process, the user can be required to give a PIN code or a token or a cryptographic key can be readily stored on the users mobile phone. Even QR codes containing encrypted information can be used together with mobile devices to verify transactions [5, 1].

The potential problem with OOB verification is that today, people use the internet more and more from their mobile devices, which means "the second channel" of communication is actually on the same device as the connection between the client and the server [15]. This makes man-in-the-browser attacks possible on the mobile device (so called man-in-the-mobile attack). The adversary may also be able to subvert both the user's PC and mobile device, in which case OOB is defeated [6].

This problem can be alleviated by using separate devices with a display. With this kind of external device, the user can verify and accept authentication or transaction [24]. The risk of this device getting infected is much smaller than the risk with the mobile device (which is used for many purposes, contains several apps and is exposed to various websites), but the usability of a separate device is worse, especially when one needs to be able to verify transactions anywhere. Moreover, in some cases the devices have been vulnerable to attacks or reverse-engineered completely [3].

OOB also has the problem of taking online banking out of internet, as one needs a separate device for verification. In addition, a concern with many OOB schemes that is not really addressed often enough in our opinion is the fact the whole verification process may become a routine after a while; do the users really pay attention to the data that should be verified (for example, do they always carefully check that the name of the receiver or the account number displayed on their phone is correct?). This concern could be solved with the systems that make the user verify the transaction details by inputting them on both in the browser and on the mobile device [25], but this makes the process significantly more cumbersome.

Inspecting intercepted transactions securely A little bit different method to prevent MitB that could still be considered as a type of out-of-band verification is to use a proxy (for example an USB device) to intercept a HTTP request before it leaves the potentially infected PC. This way, the user can verify that the transaction is still intact when it is sent over the network. The model introduced by Rautila and Suomalainen [22], for example, allows the user the inspect and verify transactions by intercepting them before they leave the machine. Similarly, the Zurich Trusted Information Channel [34] uses a USB device to intercept critical transactions and lets the user review and verify them.

Creating a secondary channel using images Images can be used to avert man-in-the-browser attacks to some extent. Goyal et al. [11] propose embedding the data to be verified into a personal image that has been previously supplied by the user. The challenge here is that the user has to securely deliver a set of personal images to every trusted party he or she wants to communicate with. It is also not completely guaranteed that a malware could not forge a text with these images given enough different image-text combinations and applying machine learning.

CAPTCHA can also be used to construct a secondary channel and defend against MitB if we send a challenge that malware cannot understand, and use a human as a computational resource [30]. However, today artificial intelligence has been proven quite capable of solving different image-based tasks while many humans might not be able to solve them easily.

Monitoring web page integrity Some man-in-the-browser attacks can stealthily modify the DOM to reach the adversary's objectives. For example, in a banking

application the malware might add a fake input field on top of the real one. The user will input a value to this fake field but the value in the invisible real field, set by the malware, will be sent to the server. These kind of attacks can be detected by monitoring the integrity of the DOM [19]. Toreini et al. [28] present DOMtegrity, a cryptographic protocol protecting the integrity of a web page. This kind of solution prevents many man-in-the-browser attacks. However, MitB attacks can also be carried out by not touching the DOM, if the adversary intercepts the network traffic and only changes some data values.

Client-side obfuscation Man-in-the-browser attacks can also be mitigated by using obfuscation [19–21]. Obfuscation refers to the act of generating source or machine code that is difficult to understand. If the JavaScript code and the DOM structure of a web page is obfuscated (e.g. by changing names of functions and variables, and reordering the code), it will become more difficult for the malicious extension to modify the web page and the data values on the page. Obfuscation should be different for each user session and it can even dynamically change on the fly. As in the case of DOM integrity based solutions, this does not really protect from the threat of network packet modification. The web application could encrypt the data it sends over the network to alleviate this problem.

Hardened browser Hardening software refers to the process of securing a system by limiting its attack surface. The browser can be hardened by running it from an external tamper proof hardware device, and by making it capable of establishing a mutually authenticated TLS session with the server [23]. This way, the malware cannot infect the browser and modify the transactions or web pages. Different hardening techniques such as disallowing extensions, using a minimal code build, applying anti-reverse engineering methods and employing obfuscation techniques can be used to further harden the client software. The downside of this approach is that to be totally tamper-proof, it needs hardware support. In a similar manner, trusted execution technology can be used to confirm that an authentic, non-infected client software is run on the system [13]. This approach also requires hardware support. Nor et al. propose a remote attestation scheme based on trusted execution technology in order to verify integrity of the client platform [18].

Virtualization Technologies Virtualization technologies can also be used to counter man-in-the-browser attacks. For example, the user could use a trusted cloud-based environment, the integrity of which can be verified with trusted execution technology, to carry out critical transactions. A virtualized environment is also easy to monitor, we can for example monitor and analyze the memory and network traffic of a virtual machine with machine learning techniques in order to find out whether there is any abnormal behavior going on in the system [2].

Detect anomalies in binaries and program behavior García-Cervigon and Llinàs [9] propose behavior-based detection of man-in-the-browser attacks

and analyze browser function calls in order to find profiles typical for infected browser. Buescher et al. [4] introduce a solution that detects the malicious attempts to manipulate (e.g. setting up hooks in network libraries) the browser's networking libraries. This approach is effective against many Trojans that steal or modify data on network level. On the other hand, data manipulation on the higher level (e.g. using DOM) still goes undetected.

Network traffic analysis Man-in-the-browser attacks can be detected by analyzing network traffic. Gezer et al. [10] propose using machine learning and looking at properties of network packets such as inter-arrival times and packet lengths. This approach can be successfully used to identify abnormal behavior patterns of a specific malware. The challenge of course is that different types of malware have different profiles and a malicious program often keeps devolving when new variants appear. Moreover, not all man-in-the-browser attacks need to contact command and control center (for example, consider the program in our earlier example that only changes the receiver's bank account number in the transaction) and do not really generate suspicious network traffic.

The findings of this survey are summarized in Table 1.

Table 1. The countermeasures for the MitB attack along with their pros and cons.

Countermeasure	**Pros**	**Cons**
Monitoring browser extensions	- useful in countering MitB extensions	- fine-gained rules are hard to maintain
Out-of-band verification	- prevents MitB when used correctly	- mobile device can be infected - transaction not always shown
Inspecting intercepted transactions securely	- prevents MitB when used correctly	- requires a separate device - may become a routine
Creating a secondary channel using images	- a creative idea that mitigates MitB	- not very usable - images might be forged
Monitoring web page integrity	- mitigates all MitB attacks	- monitor may be removed or deceived
Client-side obfuscation	- mitigates MitB attacks	- obfuscation can be undone
Hardened browser	- works when tamper-proof	- not very usable
Virtualization Technologies	- mitigates MitB considerably	- attack might occur between user's machine and cloud
Monitor binaries and program behavior	- useful against some attacks	- may not work for extensions
Network traffic analysis	- works when data is stolen	- does not prevent modification

5 Discussion

As the traditional security measures such anti-virus programs, firewalls and TLS encryption are mostly ineffective against man-in-the-browser attacks, both the academia and security companies have been trying to develop countermeasures that would be usable and effective. There is still no silver bullet against man-in-the-browser attacks, but the growing popularity of mobile devices and out-of-band verification solutions have alleviated the problem. At the same time, the growing number of malware on mobile platforms and the general trend of using various applications on mobile environment as well as on a traditional PC has put OOB verification in jeopardy. We have also noted that negligence of users may be a problem when inspecting the correctness of transactions, as the verification process may become routine, carried out almost automatically by the user.

Today's complex threat scenery requires advanced, multilayered countermeasures. It is worth noting that many of the countermeasures presented in the literature are orthogonal. In other words, they can often be used together without any problems. For instance, monitoring the behavior of extensions and restricting their permissions, protecting the integrity of web pages, detecting hooks in network libraries, anti-virus software, observing network traffic and out-of-band verification could all be used together. Additionally, although we have been focusing on client-side solutions here (for the obvious reason man-in-the-browser is a client-side threat), financial institutions and other service providers should also look for abnormalities in transaction on the server side and notify the user if necessary.

We believe man-in-the-browser attacks are just a part of more general trend: when applications and users increasingly move to the web environment, threats will also be there. In a sense, the web browser is becoming the new operating system where applications are executed. Just as operating systems have protection mechanisms against malicious attacks, browsers should also be strongly protected. Anti-virus vendors also need to better take notice what is happening inside the web browsers and what kind of extensions are executed there. With the co-operation of the academic community, security software vendors and service providers, we hope to see practical, multilayered countermeasures against man-in-the-browser attacks rolled out more effectively in the future.

6 Conclusion

We have presented a survey of countermeasures against man-in-the-browser attacks. These attacks, introduced over ten years ago, are still a serious threat to preserving users' privacy online, safe online banking and other online transactions. While no countermeasure seems to be completely foolproof (and still usable) against man-in-the-browser attacks, combining some of these solutions and more effectively enforcing them in real-world systems should greatly mitigate this threat in the future.

References

1. Almeshekah, M.H., Atallah, M.J., Spafford, E.H.: Enhancing passwords security using deceptive covert communication. In: IFIP International Information Security and Privacy Conference, Springer (2015) 159–173
2. Biedermann, S., Ruppenthal, T., Katzenbeisser, S.: Data-centric phishing detection based on transparent virtualization technologies. In: 2014 Twelfth Annual International Conference on Privacy, Security and Trust, IEEE (2014) 215–223
3. Blom, A., de Koning Gans, G., Poll, E., De Ruiter, J., Verdult, R.: Designed to fail: A usb-connected reader for online banking. In: Nordic Conference on Secure IT Systems, Springer (2012) 1–16
4. Buescher, A., Leder, F., Siebert, T.: Banksafe information stealer detection inside the web browser. In: International Workshop on Recent Advances in Intrusion Detection, Springer (2011) 262–280
5. Chow, Y.W., Susilo, W., Yang, G., Au, M.H., Wang, C.: Authentication and transaction verification using qr codes with a mobile device. In: International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, Springer (2016) 437–451
6. Dmitrienko, A., Liebchen, C., Rossow, C., Sadeghi, A.R.: On the (in)security of mobile two-factor authentication. In Christin, N., Safavi-Naini, R., eds.: Financial Cryptography and Data Security, Springer Berlin Heidelberg (2014) 365–383
7. Dougan, T., Curran, K.: Man in the browser attacks. International Journal of Ambient Computing and Intelligence (IJACI) **4**(1) (2012) 29–39
8. Entrust: Defeating man-in-the-browser malware – how to prevent the latest malware attacks against consumer and corporate banking. White paper. (2014)
9. Garcia-Cervigon, M., Llinàs, M.M.: Browser function calls modeling for banking malware detection. In: 2012 7th International Conference on Risks and Security of Internet and Systems (CRiSIS), IEEE (2012) 1–7
10. Gezer, A., Warner, G., Wilson, C., Shrestha, P.: A flow-based approach for trickbot banking trojan detection. Computers & Security **84** (2019) 179–192
11. Goyal, P., Bansal, N., Gupta, N.: Averting man in the browser attack using user-specific personal images. In: 2013 3rd IEEE International Advance Computing Conference (IACC), IEEE (2013) 1283–1286
12. Guha, A., Fredrikson, M., Livshits, B., Swamy, N.: Verified security for browser extensions. In: 2011 IEEE symposium on security and privacy, IEEE (2011) 115–130
13. Gühring, P.: Concepts against man-in-the-browser attacks. Technical Report. (2006)
14. Jia, Y., Chen, Y., Dong, X., Saxena, P., Mao, J., Liang, Z.: Man-in-the-browser-cache: Persisting https attacks via browser cache poisoning. computers & security **55** (2015) 62–80
15. Konoth, R.K., van der Veen, V., Bos, H.: How anywhere computing just killed your phone-based two-factor authentication. In: International Conference on Financial Cryptography and Data Security, Springer (2016) 405–421
16. Liu, L., Zhang, X., Yan, G., Chen, S., et al.: Chrome extensions: Threat analysis and countermeasures. In: NDSS. (2012)
17. Marouf, S., Shehab, M.: Towards improving browser extension permission management and user awareness. In: 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), IEEE (2012) 695–702

18. Nor, F.B.M., Jalil, K.A., et al.: An enhanced remote authentication scheme to mitigate man-in-the-browser attacks. In: Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), IEEE (2012) 271–276
19. Rauti, S., Leppänen, V.: Man-in-the-browser attacks in modern web browsers. In: Emerging Trends in ICT Security. Elsevier (2014) 469–480
20. Rauti, S., Leppänen, V.: Man-in-the-browser attacks in modern web browsers. In: Emerging Trends in ICT Security. Elsevier (2014) 469–480
21. Rauti, S., Parisod, H., Aromaa, M., Salanterä, S., Hyrynsalmi, S., Lahtiranta, J., Smed, J., Leppänen, V.: A proxy-based security solution for web-based online ehealth services. In: International Conference on Well-Being in the Information Society, Springer (2014) 168–176
22. Rautila, M., Suomalainen, J.: Secure inspection of web transactions. International Journal of Internet Technology and Secured Transactions **4**(4) (2012) 253–271
23. Ronchi, C., Zakhidov, S.: Hardened client platforms for secure internet banking. In: ISSE 2008 Securing Electronic Business Processes. Springer (2009) 367–379
24. SafeNet: Safenet etoken 3500. https://www.pronew.com.tw/download/doc/eToken3500_PB_(EN)_web.pdf (2011)
25. Saisudheer, A., Tech, M.: Smart phone as software token for generating digital signature code for signing in online banking transaction. International Journal of Computer Engineering Science **3**(12) (2013) 1–4
26. Ståhlberg, M.: The trojan money spinner. In: Virus bulletin conference. Volume 4. (2007)
27. Ter Louw, M., Lim, J.S., Venkatakrishnan, V.N.: Enhancing web browser security against malware extensions. Journal in Computer Virology **4**(3) (2008) 179–195
28. Toreini, E., Shahandashti, S.F., Mehrnezhad, M., Hao, F.: Domtegrity: ensuring web page integrity against malicious browser extensions. International Journal of Information Security (2019) 1–14
29. Tsai, K.: Addressing new nist authentication guidelines with symantec vip. https://www.symantec.com/connect/blogs/addressing-new-nist-authentication-guidelines-symantec-vip (2016)
30. Tsuchiya, T., Fujita, M., Takahashi, K., Kato, T., Magata, F., Teshigawara, Y., Sasaki, R., Nishigaki, M.: Secure communication protocol between a human and a bank server for preventing man-in-the-browser attacks. In: International Conference on Human Aspects of Information Security, Privacy, and Trust, Springer (2016) 77–88
31. Utakrit, N.: Review of browser extensions, a man-in-the-browser phishing techniques targeting bank customers. (2009)
32. Van Acker, S., Nikiforakis, N., Desmet, L., Piessens, F., Joosen, W.: Monkey-in-the-browser: malware and vulnerabilities in augmented browsing script markets. In: Proceedings of the 9th ACM symposium on Information, computer and communications security, ACM (2014) 525–530
33. Wang, L., Xiang, J., Jing, J., Zhang, L.: Towards fine-grained access control on browser extensions. In: International Conference on Information Security Practice and Experience, Springer (2012) 158–169
34. Weigold, T., Kramp, T., Hermann, R., Höring, F., Buhler, P., Baentsch, M.: The zurich trusted information channel–an efficient defence against man-in-the-middle and malicious software attacks. In: International Conference on Trusted Computing, Springer (2008) 75–91
35. Zhang, P., He, Y., Chow, K.: Fraud track on secure electronic check system. International Journal of Digital Crime and Forensics **10**(2) (2018) 137–144