Student Seminar: Exploiting Two Factor Authentication of Android and IOS

Robin Solignac 235020

March 12, 2017

1 Introduction: The 2 Factors Authentication

Two factor authentication (2FA) is a combination of 2 access control in order to make the authentication more robust to attacker. The general model is: to authenticate yourself to an online service you need to provide both an information you know (i.e password) and use a physical object you have (i.e [7]). One of the most most used "physical object" is a phone or a smartphone as large majority of people in today's world has one.

More specifically, it's the fact that your smartphone has out of band channels with respect to your PC which is used. it assume that if only the PC is compromised, the channels used to communicate with the smartphone are still secure

While other 2FA schemes exist [7, 6] we will study here the ones using smartphones as they are the most frequent in practice. Moreover, the attack focus on SMS based scheme. This is motivated by the fact that, most of the time, other 2FA solutions propose also SMS 2FA as a backup solution.

Let's assume that the scheme is the following: We want to authenticate to a service on a PC. first the user is require to enter a password, if it's correct the user receive a One-Time Password (OTP) by SMS. Second the user enter the OTP, if its correct the user accesses to the service

This paper will present 2 attacks to break this type of 2FA assuming a compromise PC and a sane smartphone. The former is to be used with android, the later on iOS

These attack were discovered by Konoth et al. [1]

2 Key concepts

Before describing the attacks, this section will explain and summarize the key concept of the attacks. We first describe what is synchronization and why it's a threat to the 2FA concept. The second describe the setup in which the type attack performed: a Man in the Browser attack.

2.1 Synchronization

Our use of informatics is mostly divided between our personal computers and smartphones. For this reason software manufacturer have decided to implement synchronization processes between the two devices to make transition and general use smoother. Thus they blur the line between these two. This means that the smartphone is less and less out of band with respect to the PC, which breaks the main assumption done by 2FA scheme using smartphone.

Three examples of such synchronization

which are used in the attacks are the following:

Google play remote install It's possible from a PC via the Google play website to remotely install an application on an Android smartphone if both are logged on the same Google account. The only acknowledgment appearing on the phone afterward is the application icon in the app tray and a notification saying "<app_name> has been successfully installed".

Apple Continuity On recent version of iOS and macOs you can enable this Continuity option to synchronize, read and send your SMS from your mac.

Browser synchronization Almost all of today's most popular Internet browser (including Firefox, Chrome and Safari) propose synchronization between the mobile and desktop versions logged under the same user account. it can synchronize history, bookmarks and currently open tabs.

2.2 MitB: Man in the browser attack

MitB is a type of attack resulting in the attacker having an entire control and view on the PC browser of the victim. Like a man in the middle attack, the attacker can see all data exchanged between the browser and server and can modify them (on the fly). It also can send and receive data in the name of the user. But unlike the former, it has access to these data *before they are encrypted and after they are decrypted*. It can also modify browser related settings like bookmarks, history and current open tabs. In short, a powerful man in the browser attack can remotely perform the same actions has someone getting physical access to the browser as well as modify content of the requests and responses.

There are different ways to do a MitB attack, using malware infecting the whole system, by API hooking or via malicious plugin [3, 5].

3 2FA Attack

This section describes the attacks to defeat 2FA under the following model: we assume that the PC of the customer is compromised and can perform man in the browser attack but the smartphone is still sane. The attacker tries to authenticate on a service from the PC. It gets the password from the PC infection but the service requests a second factor authentication by sending a one time password (OTP) with SMS to the victim's smartphone who runs either under Android or iOS. This choice of model is motivated by the fact that 2FA is specially designed to prevent authentication to services because of only an infected PC.

3.1 Android

The principle to the attack is: The attacker install an application on the victims smartphone who has authorization to read SMS. When the OTP SMS is received, it is forward via Internet to the attacker, who can successfully authenticate. The application is installed remotely from MitB who has hijacked a victim Google session do the install and authorization through the playstore web interface. The hijack is also done by the MitB via password, cookie stealing or else.

In order to succeed the attacker need to pass two defense setup by Google.

Bypassing Google bouncer Google remote install only allows to install application published on the Google play store. So the attacker needs to publish an SMS stealing app on the store. In order to do this it must bypass Google Bouncer which is an automated malware analysis tool deployed by Google on its store. Whenever an app is uploaded to the store, it's analyzed by Bouncer, which performs static analysis (e.g code inspection, ...) as well as dynamic analysis (i.e, sending request to the app and analyzing the resulting behavior)

Recent work [4, 2] show that this defense is easy to bypass in various ways. [1] is using another clever way. the app opens a poorly protected of web window, not visible on the smartphone screen. From which it's possible to remotely execute malicious (Javascript) code on the phone. As this code is invisible to Google boncer the app will not be detected as malicious.

Let us note that this possibility to load web content and let it execute some code on the app is another intended feature made by Google called Web-View.

Activate the app When installed, an Android app can't be triggered by an external event (such as RECEIVE_SMS) until it has been explicitly opened for the first time. So, we need to trick the user to open the app from the phone.

The first way is to give a clickbait title to the app so that the user will want to open it when he sees the installation notification.

The second is to trigger its opening from the mobile browser by clicking a link. with browser sync activated (by default on chrome) we can modify with MitB all open tabs, bookmarks, and history URL to be this particular link, making it eventually clicked on. We can make the app so that it will then redirect the user to the normal URL immediately after this opening, making the move difficult to notice.

In both case just after this opening the app can make itself disappear from homescreen and app tray, leaving little trace of its existence of the smartphone(only in the application of the parameters men). While still permanently running in the background.

3.2 iOS

Since 2015 iOS has forbid applications to read all notifications or SMS without explicit authorization (or they will be rejected from entering the Appstore). Since then, the previous attack does not work.

However, if the infected browser is on a Mac and Continuity is activated on the Iphone, the browser can still has access to SMS in clear on the mac as soon as both are on the same LAN, which is likely to eventually occur as they belong to the same person.

So, under these pretty likely to appears conditions, the 2FA authentication can be very easily bypassed on 2FA by an MitB attack.

4 Discussions

This section will discuss practical feasibility of these attacks and potential solution to defeat them on both platforms.

4.1 It's not a exploit, it's a feature

Before talking about feasibility we must emphase a particularity of these two attacks: After the initial infection of the PC, no exploit, bug or hack are used. Every tool we used as been purposely designed for how we use them. Service providers (i.e, Google, Apple, Mozilla, ...) try to reduce the air gap between the PC and the smartphone and at the same time security engineer rely on it too construct secure 2FA.

And that's why defeat these attacks are complicated, there's nothing fix. Either developers should modify the offered synchronization possibilities, And most of the time it would mean reducing them. Or security engineers should modify the assumption they make on 2FA, which mean finding new ones.

4.2 Android

This attack is largely feasible because its nearly impossible to use an Android phone without Google account and if you have one it's extremely likely that you will also use it on your PC. And a Google account gives both the possibility the remotely install app and modify Google chrome sync settings

However, the truth is that a the solution against to this attack already exists but can still be bypassed. The solution is the following: Since Android Marshmallow, deployed in October 2015, the authorization model has changed. All authorizations are now granted individually at run time when needed. With this new model, authorization to read SMS can't be given from a remote browser because it require text box confirmation on the phone screen. The only problem is that if the app is compiled for older version of Android the authorization scheme will be the old one while still running on newer versions. But it is likely that in the future, it will be forbidden for apps to be compile for versions older than Marshmallow.

It's also likely that the pure engineering challenge Google bouncer will become more and more powerfull in the future.

4.3 iOS

The iOS version of the attack is a even bigger example of the problem describe previously. The attack only use one feature made by Apple itself to defeat the whole 2FA concept. The only solution to this is a modification of ether the Apple continuity feature or the 2FA scheme used.

On the feasibility side, this attack needs more prerequisites: user needs to use a Mac, have Continuity enable (off by default) and the 2 devices must be on the same LAN. This do not reduce the feasibility of the attack as the conditions are still likely to happened. But, it reduces scalability of a such attack comparing to the Android one.

A solution to this attack on Apple side with light impact of user experience would be to only sync SMS whose sender is in contact list of the user.

4.4 Other 2FA authentication on smartphone

SMS OTP are not the only existing schemes of 2FA on smartphones. Majority of services also propose 2FA dedicated app (such as Google authenticator or Azure authenticator). As on both iOS and Android application can't access each other data's (Application sandboxing technique) both attacks are defeated. But most of the service which implement 2FA this way also include SMS 2FA as a backup solution. If for some reason the user can't use or access the app (i.e lack of mobile network or buggy phone) it can still authenticate through SMS. So, if the user has provide its phone number to the service, 2FA using dedicated app can still be defeat by these attacks by asking, from the PC, to use the backup solution to authenticate.

5 Conclusion

The very soul of 2FA using smartphone is the assumption that there's an air barrier between your smartphone and your channel and is has been true for many years. But except for 2FA this barrier is not seen as a desirable feature anymore, and so it's been reduced over the year. This needs to be taken into account while implementing 2FA on smartphone. Services must make sure that the channel they are using is really out of band with the user's PC. What's sure is that today 2FA through SMS is not secure anymore on Android and iOS and should be avoided, even as a backup solution.

References

[1] Victor van der Veen Konoth Radhesh Krishnan and Herbert Bos. *How Anywhere Computing Just Killed Your Phone-Based Two-Factor Authentication*. VU University Amsterdam, The Netherlands. URL: fc16.ifca.ai/preproceedings/24_Konoth.pdf.

[2] Miller C. Oberheide J. *Dissecting the Android Bouncer*. Jun 2012.

[3] Guhring P. *Concepts against Man-in-the-Browser Attacks*. September 2006.

[4] Vigna G. Poeplau S. Fratantonio Y. Bianchi A. Kruegel C. *Execute This! Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications*. 2014.

[5] R.J. Bansal R. Sood A.K. Enbody. *The art of stealing banking information form grabbing on fire*. November 2011.

[6] Wikipedia. *Automated teller machine*. URL: https://en.wikipedia.org/wiki/Automated_teller_machine.

[7] Wikipedia. *SecurID*. URL: https://fr.wikipedia.org/wiki/SecurID.

Other references can be fond in the original paper [1]