A Survey of Multi-factor Authentication Techniques

Anonymous
Removed

ABSTRACT

The abstract should be one or two paragraphs that summarize your paper. Abstracts are read independently from the rest of the paper so you cannot cite your paper or any other papers in it. Study other abstracts in the papers you are reading to understand what an abstract should really means. **Write the abstract in third person.**

The abstract should make it clear what your work is about understanding research papers and about integrated ideas from others. Don't write it as though you did the work done by the researchers who did the original research!

The abstract is not an introduction or overview, but a summary that informs the reader of the context, content and contributions of your paper.

ACM Reference format:
Anonymous. 2020. A Survey of Multi-factor Authentication Techniques. In *Proceedings of NA, NA, 2020,* 3 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The standard username and password approach to authentication is vulnerable to both technical and social attacks. Weak passwords can be brute-forced, while strong passwords can be stolen through phishing. As a result, many secure websites and services are now requiring additional evidence, or "factors", for confirming a user's identity.

Authentication evidence can be broken down into three different categories: something you have, something you know, and something you are. In the standard username and password scheme only the "something you know" category is used. Multi-factor authentication attempts to improve upon that level of security by requiring users to present evidence from two or more of those categories in order to successfully authenticate.[8] Multi-factor authentication can be further broken down into two sub-categories: two-factor authentication and two-step authentication.

Two-factor authentication combines two different pieces of authentication evidence from different categories to authenticate a user. An example of this would be withdrawing money from an ATM: a user is required to present a bank card (something they have) as well as a PIN number (something they know).

Two-step authentication requires a second factor other than something that a user is or has in addition to something that a user knows. Two-step authentication requires users to complete a secondary authentication mechanism after presenting a known secret to securely ensure the identity of the user. An example of this would be providing a secret that was sent to a user through external means, such as a secret code sent to a user via email during a login attempt.

This paper explores the trade-offs between these different forms of evidence and their use in authentication systems. A demonstration application is used to test some of the currently popular systems and understand their impact on both security and usability. The adoption of a multi-factor authentication solution is a significant change for any platform.

2 DESIGN CONSIDERATIONS

Today, there are several options available for multi-factor authentication. SMS and email communications are common ways to receive out-of-band second factors, such as one-time passwords (OTPs). Duo, an authentication solution used by large organizations for a fee, offers convenient two-step authentication at the touch of a button on a user's smartphone [5].

OTPs can also be generated via a device that a user has. Protocols such as TOTP (time-based) and HOTP (counter-based) allow users who possess an OTP hardware device or a smartphone app to log in with a periodically updating second factor that they have access to [1]. This method of additional authentication is typically considered much stronger than SMS or email second factor codes.

Hardware accessories that follow the FIDO U2F (Universal 2 Factor) standard can be used for multi-factor authentication [9]. Yubico, a popular hardware token manufacturer, produces the Yubikey security USB device. Yubikeys are affordable FIDO/U2F compliant devices that provide a second factor without the use of OTPs.

While multi-factor authentication can create very secure login experiences when implemented properly, each mainstream two-step authentication method has its downfalls and inconveniences. SMS and email communications are easily hijacked in transmission while Yubikeys and smartphones are easily physically stolen. Adding a two-step authentication method also creates frustration and misunderstanding with non-technical users. The pros and cons of each multi-factor method need to be considered before integrating them into an authentication system. Users signing into a multi-factor authentication system need to understand why the additional security mechanisms being used are necessary [1].

A secure, convenient, and easy to understand authentication system offers several two-factor authentication options and does not force users into using a particular method. Technical users who possess hardware tokens such as a Yubikey will generally prefer to use that option over the Duo authentication app. Requiring more than one secondary factor in addition to the common username and password is rarely considered as logging into an authentication host is often a time sensitive activity. Some authentication hosts have even gone as far as removing the traditional password, and instead require users to present a short pin followed by an OTP code

NA, NA
 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

to make the login experience more manageable while maintaining tighter security policies.

3 EXISTING BEST PRACTICES

Lower-risk consumer applications (such as email, online shopping, and social media) are currently best served by more user-friendly MFA solutions such as text or email based OTPs. On the other hand, higher-risk corporate applications are better off using dedicated hardware or software solutions like Duo, Yubikey, and Smart Card systems.

This dichotomy is primarily due to the fact that consumer applications are largely confined to opt-in MFA models, and as a result are limited to solutions that are familiar and readily available to their users.[1] Corporate and institutional applications on the other hand can easily mandate more secure options when accessing employee portals and company networks. This is because corporate employees are (ideally) concerned about the security of their company's systems. Psychological acceptance in MFA solutions is absolutely critical, and plays a large role in the implementation of new MFA options within a system.[8]

It is important to note that while text and email based OTPs are vulnerable to social engineering attacks[6], they still offer a significant improvement over traditional username and password authentication. While traditional passwords can be cracked via brute force, OTPs expire after a short amount of time, making long-winded brute force attacks much less likely to succeed.

4 ARCHITECTURE

The standard MFA architecture for a web application is quite similar to password-based authentication, and in fact is often built on top of it. For email or SMS OTP systems when the user attempts to login the server will generate an OTP using a secure source of randomness and save it to a database along with a timestamp. Then the OTP is transmitted to the user over their preset communication method. Once the user receives the OTP, they enter it into the application for validation against the copy in the database.[4]

TOTP and HOTP systems (of both the software and hardware varieties) tend to have a similar architecture to transmitted OTP systems. The main difference is in the case of TOTP and HOTP there is a secret that is transmitted to the user during enrollment (or burned into their hardware token) so both the server and the user have a copy of it. On the server side the secret must be stored in a secure database and protected to the same degree as a password hash. Then when the user attempts to authenticate they calculate the OTP themselves using the secret and send it to the server for validation.

Third party MFA providers such as Duo and Yubikey operate off of a slightly different model where the server that the user is attempting to authenticate to reaches out to the third party's service to collect the additional evidence. In the case of Duo this is handled by redirecting the user's browser. Yubikey OTPs on the other hand are submitted to the application's server and then sent on to the Yubikey service for validation. In either case, the result of that validation is provided to the application's server by the third party. This offloads much of the complexity of implementing a secure MFA system but makes the security of the application dependent on the third party.

The architecture for non-web applications takes a slightly different approach. In most cases (SSH, VPN login, building entry) the user is still prompted for the secondary evidence which is then passed off to a server for validation against a stored secret. The only time this model changes is in the case of decentralized MFA[10]. In that case the secrets have to be shared by all parties so they can mutually generate and verify tokens.

5 IMPLEMENTATION

All of these specifications were reviewed in order to properly implement our code. The TOTP[3] and HOTP[2] specifications were crucial for ensuring that the provided TOTP and HOTP functions work with standard OTP client code generators, such as Google Authenticator.

The pyqrcode python code library was used to generate TOTP and HOTP QR codes. These QR codes contain a TOTP or HOTP compliant URL that share the OTP secret with the client OTP generator. The python secrets standard library was used instead of the infamous python random library for creating OTP secrets and messages. The secrets library is crytopgrahically strong, while the random library is a very weak pseudo-random entropy generator.

For sending OTPs via email, the standard python email and SMTP libraries were used. This decision was made to demonstrate that email OTP systems are not hard to implement and might not require external libraries.

TODO: Discuss secure password generator code for use in secrets and OTP sending code. FIDO Spec for when we implement yubikey support: [9] Attacks to design around: [7]

6 LESSONS LEARNED

6.1 Response to Investigating Teams

Discuss how you addressed the issues, if any, that were identified by the investigating teams. Be specific.

6.2 Other Lessons

Use this section to describe mistakes you made and corrected (or did not get a chance to correct including why you didn't). Also describe what all you learned during the course of this effort; this section, like the others, plays a critical component in determining your final grade.

7 ETHICAL AND LEGAL ISSUES

Topics to cover: Government may require hardware / software tokens to be turned over. Government may require authentication hosts to bypass login requirements, or otherwise present information without any login information. Purchasers of Yubikeys and other MFA accessories may be viewed as "having something to hide". Local governments could outlaw the sale of Yubikeys and other devices. Given enough money, a Yubikey could be deconstructed and examined on an atomic level by a government agency to determine the secret bits it holds [10].

Any biometric login factor is sensitive to the owner and should not be shared.

Any broken MFA system is inherently doing damage to it's users with or without their knowledge. Just because 2FA is enabled, exploits or bugs are still possible, and sometimes 2FA gives the illusion of security to users.

A broken MFA system could also potentially refuse to authenticate someone, thus temporarily banning them from accessing the given system.

8 CONCLUSIONS

In this section we'll discuss what we see as the correct response to our lessons learned and any changes we suggest to the existing best practices or status quo.

9 CURRENT STATUS & FUTURE WORK

Use this section to describe the current status of your work and what else should be done. Also, discuss what further directions your work can be taken by others.

REFERENCES

[1] Emiliano De Cristofaro, Honglu Du, Julien Freudiger, and Greg Norcie. 2013. Two-Factor or not Two-Factor? A Comparative Usability Study of Two-Factor Authentication. *USEC* (09 2013). DOI : http://dx.doi.org/10.14722/usec.2014.23025

[2] M'Raihi et al. 2005. HOTP: An HMAC-Based One-Time Password Algorithm. (2005). https://tools.ietf.org/html/rfc4226

[3] M'Raihi et al. 2011. TOTP: Time-Based One-Time Password Algorithm. (2011). https://tools.ietf.org/html/rfc6238

[4] Kathleen Garska. 2018. Two-Factor Authentication (2FA) Explained: Email and SMS OTPs. (September 27 2018). https://blog.identityautomation.com/two-factor-authentication-2fa-explained-email-and-sms-otps

[5] Tom Henderson. 2015. For Duo Security, data security is job security. *Crain's Detroit Business* 31, 30 (Jul 27 2015), 12. https://ezproxy.rit.edu/login?url=https://search-proquest-com.ezproxy.rit.edu/docview/1700196757?accountid=108 Copyright - Copyright 2015 Crain Communications Inc. All Rights Reserved; Last updated - 2015-07-30.

[6] Markus Jakobsson. 2020. Social Engineering Resistant 2FA. (2020). arXiv:cs.CR/2001.06075

[7] Collin Mulliner, Ravishankar Borgaonkar, Patrick Stewin, and Jean-Pierre Seifert. 2013. SMS-Based One-Time Passwords: Attacks and Defense. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, Konrad Rieck, Patrick Stewin, and Jean-Pierre Seifert (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 150–159.

[8] Ken Reese, Trevor Smith, Jonathan Dutson, Jonathan Armknecht, Jacob Cameron, and Kent Seamons. 2019. A Usability Study of Five Two-Factor Authentication Methods. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*. USENIX Association, Santa Clara, CA. https://www.usenix.org/conference/soups2019/presentation/reese

[9] Sampath Srinivas, Dirk Balfanz, Eric Tiffany, and Alexei Czeskis. 2017. Universal 2nd Factor (U2F) Overview. (Apr 2017). fidoalliance.org

[10] Guomin Yang, Duncan S. Wong, Huaxiong Wang, and Xiaotie Deng. 2008. Two-factor mutual authentication based on smart cards and passwords. *J. Comput. System Sci.* 74, 7 (2008), 1160 – 1172. DOI : http://dx.doi.org/https://doi.org/10.1016/j.jcss.2008.04.002