TOTP Based 2-Factor Authentication: Future of Security

Archita Kale¹ & Utkarsh Dhawan²

¹,²Department of Computer Engineering, KJ Somaiya Institute of Engineering & IT affiliated to Mumbai University

Abstract: In today's scenario with the growing risk of information being compromised security is a major issue. Only account passwords are not enough to keep you secured anymore. There are various ways in which an attacker can gain an insight into the user's account passwords and details, with the advancements in the field of hacking and system attacking, an attacker can use one of the many available tools like key loggers or develop exploits to gain personal information of the user. The development in the field of authentication has involved intermixing of various mechanisms with each other. Two factor authentication or 2FA involves layering of 2 different components to ensure better security of the user. These components may be something that the user knows, something that the user possesses or something that is inseparable from the user. This paper presents an approach to protect the user's online accounts using TOTP (Time based one time password). TOTP is a temporary pass code generated by an algorithm, this pass code is dependent on the time-stamp and changes with time.

1. Introduction

Two factor authentication is a method which allows the user to protect their data using two layer protection mechanism. This ensures the data security at higher level. Let us consider, the case of an ATM in which for any transaction we need 2FA. One is the the card and other is the pin number. Another example is the Biometric system for which we need a thumb impression which is not separable with the pin number which the user possess.

Basically it's a security system which requires two authentication means. These two means should be of different class or category and their combination should be unique to the user. But is this technology the ultimate solution or there chances of user data being compromised, this paper seeks to answer these questions. The TOTP algorithm utilizes a shared secret key and the current Unix time-stamp to generate a pass code. Generally the time-stamp increases in 30 seconds, so passwords generated close together in time from the same secret key will be equal on both the levels i.e. server and client.

Typically, In a 2FA system: the user enters his login credentials into a server, the server generates a OTP using the time-stamp running locally on the client device. The server can now run TOTP to verify the one-time password entered by the user. Synchronization between the client and the server is important and so the one-time passwords are accepted from time-stamp difference of ±1.

Later in this paper we will discuss about the Vulnerabilities, Background, Computation and Weakness and finally an Implementation of the TOTP as a 2FA tool.

1.1. Reason of concern

As times have changed so have the ways of communication and business, with the increase in utilization of the technology in these processes emphasis on cyber security has become discernible. Even a minor vulnerability could affect the complete system, Social engineering has been a reason to a plethora of compromises. Phishing is one of the major concerns in these cases, replicated fake web pages are designed to trick users to give up their personal information. An example of this was witnessed during the Veteran Administration (VA) incident on July, 2006 when the discharged veteran records of about 26 million veterans were stolen from the home of an employee who "improperly took the material home". There are thousands of such cases occurring every day at either small or large scale. Thus, full proof protection of this information cannot be guaranteed therefore it would be better to use a system which rules out this vulnerability once and for all.

2. Background

Two Factor Authentication or 2 step verification was patented in 1984. It is used where there are important information stored or where privacy cannot be risked. Protection is assured when there are two channels involved for security. The Initiative for Open Authentication (OATH) identifies standards for two-factor authentication. OATH introduced HOTP as the first open and freely available algorithm to generate event-based one-time passwords. HOTP was established in 2005. Later in 2011 TOTP came into action when the use of smartphones increased rapidly.

TOTP is a practical application of a Hash-based message authentication code (HMAC). It
is a specific type of message authentication code (MAC) involving a cryptographic hash function in combination of a secret cryptographic key. The cryptographic strength of the HMAC depends upon the cryptographic strength of the underlying hash function, the size of its hash output, and on the size and quality of the key.

There has been a rise in security failures due to increase in the awareness and quality of vulnerabilities and exploitations. Increasing the factors of authentication can lower the chances of being compromised by a great factor. TOTP is an effective mechanism to implement two-factor authentication, since TOTP generates a temporary pass key which changes with time. It is not practical to crack the key within 30 seconds, hence the technique is quite effective.

3. Computation

TOTP has its fair share for background, since it uses the HMAC-SHA-1 algorithm similar to the HOTP. We will look into the computation of the passcode using TOTP.

3.1. The HOTP Connect

RFC 4226 states that HOTP is based on an HMAC-SHA-1 algorithm. In HOTP the HMAC-SHA-1 algorithm is run with the secret key and an incrementing counter.

HMAC-SHA-1(SECRET, COUNTER) this would generate a 20-byte string. This string is used to generate a 4-byte string using Dynamic Truncation.

\[ DT(String) \rightarrow \text{String[0]}...\text{String[19]} \]

Let OffsetBits be the lower 4 bits of String[19]

\[ \text{Offset} = \text{StToNum(OffsetBits)} \]

// 0 <= Offset <= 15

Let \( P = \text{String[Offset]}...\text{String[Offset+3]} \)

Return the Last 31 bits of \( P \)

The modulus of the result is calculated with 10^digit where digit is equal to the length of the key to be produced. Different processors perform these operations differently and thus the most significant bit is masked to avoid confusion between the signed and unsigned modulo computations.

In the case of TOTP the incrementing counter is replaced by number of time-steps, i.e. (Unix Time-stamp-T) / X. Here T is the initial counter and X is the time-step. Let us consider, the time-stamp is 42, T and X are 0 and 30 respectively then the resultant output will be 1 but it would be 2 if the time-stamp is 60.

3.2. Computing a one-time code

The HMAC-SHA-1 algorithm is implemented with a secret key and a current time-step:

\[ \text{HMAC\_SHA-1(SECRET, \text{time()/30})} = \text{2e7898426e69ca166345612a7f19da8e125a879b} \]

This 160 bit message authentication code can be used directly but it would be impractical to type approximately 49 digits within 30 seconds to authenticate thus we perform dynamic truncation to reduce it the authentication code to 31 bits:

\[ \text{0x2a7f19da} \rightarrow (712972762)_{10} \]

Now the last 6 digits of this decimal are selected: 972762

This is the TOTP pass code for the next 30 seconds.

4. Real time scenario

2FA can be implemented with ease using the TOTP algorithm, available are several open source libraries. The one used here is [AG] , this library helps in producing Base32 secret keys and also generates the 6 digit standard length passcode using the combination of the secret key and the Unix time-stamp. Here we will try to show the implementation of this algorithm using an android application.
4.1. Third Party Dependencies

For the application we will utilize a few of the third party dependencies which will make our job easier.
1. Zxing ("Zebra Crossing") [ZX] : It is an open-source, multi-format 1D/2D barcode image processing library.
2. AeroGear OTP Java [AG] : It is a Java library for generating OTP according to RFC 4226.

4.2. Utilizing the dependencies

Zxing provides with methods to use the barcode scanner.

    IntentIntegrator.initiateScan(this);

Is used initiate the Zxing Scanner and the result is obtained using onActivityResult(String requestCode, String resultCode, String data).

Is used initiate the Zxing Scanner and the result is obtained using onActivityResult(String requestCode, String resultCode, String data)

    IntentResult scanResult = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
    Uri uri = Uri.parse(scanResult.getContents());
    String[] uriParts = uri.toString().split("otpauth://totp/");
    String[] fetchAccountName = uriParts[1].split("\?");
    String protocol = uri.getScheme();
    String server = uri.getAuthority();
    String path = uri.getPath();
    Set<String> args = uri.getQueryParameterNames();
    String secret = uri.getQueryParameter("secret");
    String issuer = uri.getQueryParameter("issuer");
    String accountName = fetchAccountName[0];

This code segment above shows URL parsing of otpauth address received from the QR code. This code would be common for all the email providers as the format of the uri is predefined.

    otpauth://totp/$LABEL?secret=$SECRET

Now the secret key is used to generate the one time password using TOTP.

    TOTP otp = new TOTP(secret);
    otp.now(); //541613
    Thread.sleep(50);
    otp.now(); //583839

The generated OTP can be verified with the server which would be running the same algorithm at it’s end.

5. Weakness and Vulnerabilities

Although this algorithm is quite a reliable but there are still some factors which make this vulnerable and not full-proof. The algorithm is heavily dependent upon the secrecy of the shared secret key. If this secret key gets compromised due to some server faults, then the generated pass code is an open stick. Also the clients which implement the TOTP algorithm consist of batteries that can get exhausted, there are chances of the desynchronization of time clock between the client and server. Also there are chances that the mobile phones on which the software is running is stolen. The major vulnerability lies with the various real world implementations which have methods of getting bypassed like email resets, printed codes etc.

Recently vulnerabilities like heartbleed have made the online world more insecure. Heartbleed was a recent flaw detected in the OpenSSL cryptographic software library which allowed fetching the information which is hidden under the SSL/TLS encryptions used to secure the Internet. This would allow the hacker to get hold of the secret key used to produce the pass code. Man in the middle attacks can also be some risks of major concern.

Also one of the major issues arise with the clients that display the TOTP pass code is the insecure way of displaying. Most of the times the pass code arise in notifications or written in the form of simple texts which can be easily accessed by some peer. Also some malwares can be developed to fetch the message or diverting the calls that contain the code.

6. Acknowledgements

We would like to thank Mrs. Nilambari Joshi and Harsh Daftary for their helpful contributions.

7. Conclusion

With the growing rate of security breaches and information being compromised, multi factor authentication is the need of the hour. TOTP can be considered as an effective and simple way of implementing 2 factor authentication(2FA) on a system. Though, the algorithm is quite secure in itself but there are various other factors that make it vulnerable like losing secret key etc.

8. References


