به‌نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

تحقیق برنامه‌نویسی وب
احراز هویت با Google

نگارندگان:

مهراد میلانلو، یاسمین گلزار

۹۹۱۰۵۷۷۵ - ۹۸۱۷۱۰۶۴

فهرست مطالب

۱ مقدمه

احراز هویت یا احراز اصالت کاربر [۱] اولین لایه‌ی دفاعی در بسیاری از سیستم‌هاست. این عملیات پیش‌نیاز بسیاری از مکانیزم‌های امنیتی از جمله مجازشماری [۲]، کنترل دسترسی [۳] و حسابرسی [۴] است. بنابراین احراز هویت یکی از مهم‌ترین مسائلی است که در طراحی سیستم‌های اطلاعاتی باید به آن توجه شود.

۱.۱ احراز اصالت کاربر چیست؟

طبق تعریف استاندارد RFC 4949، احراز اصالت کاربر فرایند وارسی هویت یا شناسه‌ی مورد ادعای یک کاربر است. بنابراین احراز هویت یعنی اطمینان از اینکه فردی که ادعا می‌کند فردی خاص است، واقعاً آن فرد است. در واقع بعد از احراز هویت، می‌توانیم به اطلاعاتی که فرد ارائه می‌دهد اعتماد کنیم.

فرایند احراز هویت کاربر شامل دو گام است:

۱. **گام شناسایی [۵]:** فراهم کردن هویت یا شناسه به سیستم.
برای مثال: ارائه‌ی نام کاربری در بسیاری از سیستم‌ها

۲. **گام وارسی [۶]:** فراهم کردن اطلاعاتی جهت اثبات تعلق هویت/شناسه‌ی ارائه‌شده به کاربر مدعی
برای مثال: ارائه‌ی گذرواژه جهت اثبات ادعا

۲.۱ روش‌های احراز اصالت کاربر

احراز هویت کاربر می‌تواند از روش‌های مختلف فیزیکی، دانشی، مالکیتی و یا دیگر روش‌های احراز هویت انجام شود.

- **بر اساس دانسته‌های کاربر**
مثال: گذرواژه، PIN
مشکلات و خطرات: حطر افشا، حدس زدن، فراموشی

- **بر اساس داشته‌های کاربر**

  - **توکن‌های سخت‌افزاری متصل**
مثال: انواع کارت‌های مغناطیسی یا هوشمند و یا توکن سخت‌افزاری که به دستگاه متصل می‌شود.

  - **توکن‌های سخت‌افزاری غیرمتصل**
مثال: توکن گذرواژه یک‌بار مصرف [۷] که به دستگاه متصل نیست و اطلاعات احراز را نمایش می‌دهد.

مشکلات و خطرات: خطر سرقت، مفقود شدن، کپی کردن

- **بر اساس مشخصات بیولوژیکی کاربر**

[1] Authentication
[2] Authorization
[3] Access Control
[4] Accountability
[5] Identification
[6] Verification
[7] OTP

- **آن‌چه که هست**

  مثال: اثر انگشت، چهره، شبکیه‌ی چشم

- **آن‌چه که انجام می‌دهد**

  مثال: ریتم تایپ‌کردن، نحوه‌ی صحبت کردن، دست‌خط

مشکلات و خطرات: خطر خطا در تشخیص، هزینه‌ی بالا

۳.۱ احراز اصالت چندعاملی

همان‌طور که در توضیحات و مقایسه‌ی بخش قبلی مشاهده می‌شود، هر کدام از روش‌های احراز هویت می‌تواند جداگانه مشکلات غیرقابل چشم‌پوشی داشته باشد. به همین دلیل بسیاری از سیستم‌ها از ترکیب چندین روش برای احراز هویت استفاده می‌کنند. این دسته از روش‌های مبتنی بر ترکیب چند روش احراز اصالت (حداقل دو روش)، به عنوان احراز هویت دو عاملی را **احراز اصالت قوی** یا **احراز اصالت چندعاملی** [۸] می‌نامند.

۲ اهمیت احراز هویت دو عاملی

به‌طور کلی، احراز هویت دو عاملی [۹] به عنوان یک گزینه‌ی بهتر از احراز هویت تک‌عاملی [۱۰] مورد توجه قرار می‌گیرد که معمولا شامل نام کاربری و رمز عبور است. احراز هویت دو عاملی احتمال تقلب یک حمله‌کننده به عنوان یک کاربر مجاز را به‌گونه‌ی مشهودی کاهش می‌دهد.

اهمیت احراز هویت دو عاملی در یک پست وبلاگ در فوریه سال ۲۰۲۲ میلادی توسط Guemmy Kim، مدیر امنیت حساب و ایمنی وقت در گوگل، به روشنی آشکار شد. طبق گفته‌ی خانم Kim، گوگل سال پیش از آن، کاربران را به صورت خودکار در احراز هویت دو عاملی شرکت داده بود. پس از این‌که بیش از ۱۵۰ میلیون نفر با احراز هویت دو عاملی ثبت نام شدند، گوگل کاهش ۵۰ درصدی در تعداد حساب‌های مورد نفوذ را مشاهده کرد.

احراز هویت دو عاملی نیازمند ارائه‌ی دو نوع تایید هنگام ورود به یک وب‌سایت یا سیستم است. به عنوان مثال، ورود با نام کاربری/رمز عبور یک نوع است و یک رمز عبور ارسال شده به کاربر نشان‌دهنده‌ی نوع دوم است. این انواع احراز هویت ــ که می‌دانیم به آن‌ها فاکتورها گفته می‌شود.ــ به تایید هویت کاربری که تلاش می‌کند به یک سایت یا خدمات امن وصل شود، کمک می‌کند و نشان می‌دهد واقعاً آن فردی است که ادعا می‌کند.

متاسفانه، رمزهای عبور معمولا بزرگ‌ترین ضعف در حفاظت شبکه و داده‌ها هستند؛ به خصوص زمانی که به انتخاب آن‌ها توجه کافی نمی‌شود و برخی از آن‌ها به اندازه‌ی موردنیاز قوی نیستند. یک وب‌سایت یا خدمات آنلاین که تنها نام کاربری و رمز عبور را نیاز دارد، از احراز هویت تک‌عاملی استفاده می‌کند زیرا فقط به فاکتورهای دانش برای احراز هویت کاربران وابسته است. با این حال، با پیچیدگی حملات سایبری، تنها نام کاربری و رمز عبور کافی نخواهد بود. برای حفاظت کامل از منابع و بسیاری از سامانه‌های آنلاین اکنون به صورت اجباری احراز هویت دو عاملی قرار داده شده‌است یا حداقل به‌عنوان یک گزینه‌ی اختیاری پیشنهاد داده می‌شود.

در حال حاضر، بسیاری از پروژه‌های پیاده‌سازی شده از احراز هویت دو عاملی از ترکیب ورود با نام کاربری/رمز عبور به عنوان فاکتور اول استفاده می‌کنند و فاکتور دوم یک رمز عبور موقت است که کاربر باید همراه با اطلاعات ورود وارد کند. کاربر رمز عبور موقت را روی دستگاه خود دریافت می‌کند. این پیام به‌طور معمول به صورت پیام متنیست یا از طریق یک برنامه‌ی تایید هویت مانند Google Authenticator برای

[8] Multi-factor Authentication

[9] Two-factor Authentication or 2FA

[10] Single-factor Authentication or SFA

مدت زمان محدود قابل دسترسی است. دستگاهی که پیام متنی را دریافت می‌کند، معمولا دستگاهیست که کاربر روزمره در اختیار دارد. بنابراین معمولا تلفن هوشمند به عنوان یک فاکتور دارایی در احراز هویت دو عاملی قابل قبول است.

۳ Google Authenticator

نرم‌افزار Google Authenticator یک نرم‌افزار احرازکننده‌ی هویت است که روی دستگاه‌های موبایل می‌تواند نصب شود. این نرم‌افزار متن‌باز [۱۱] ، اولین بار در ماه سپتامبر سال ۲۰۱۰ میلادی توسط شرکت گوگل معرفی و منتشر شد و هم‌چنان در حال توسعه می‌باشد. در حال حاضر این برنامه برای سیستم‌عامل‌های `Android`، `iOS`، `Blackberry` و دیگر سیستم‌عامل‌های مربوط به لوازم جانبی پوشیدنی، به زبان‌های `Java` و `Objective-C` نوشته شده‌است. Google Authenticator از الگوریتم‌های `TOTP` [۱۲] و `HOTP` [۱۳] برای تولید عبارت رمز مشترک استفاده می‌کند. زمانی که به یک وب‌سایت یا سیستمی که احراز هویت را پشتیبانی می‌کند مراجعه می‌کنید، این نرم‌افزار که بر روشی گوشی همراه شما قرار دارد، رمز یک‌بارمصرفی [۱۴] را تولید می‌کند و شما برای احراز هویت دو عاملی، علاوه بر نام کاربری و رمزعبور به این رمز ۶ یا ۸ رقمی نیاز خواهید داشت.

۱.۳ کارکرد Google Authenticator از دیدگاه فنی

برای آن‌که سرور یک سیستم بتواند از احراز هویت دو عاملی پشتیبانی کند، باید در سمت Backend خودش الگوریتمی مانند `TOTP` را پیاده‌سازی کند. این الگوریتم در نهایت یک کد چند رقمی به‌عنوان خروجی می‌دهد. اکنون سرور می‌تواند به‌طور سنتی این کد را برای کاربر از طریق ایمیل یا پیامک ارسال کند و یا از `API` نرم‌افزار Google Authenticator برای به اشتراک گذاشتن کلید استفاده شده در تولید رمز استفاده کند. در این صورت، در هر ورود، کاربر رمزی که از نرم‌افزار احراز هویت گوگل دریافت می‌کند را در اختیار سرور میزبان قرار می‌دهد. در ادامه تمامی مراحلی را که برای پیاده‌سازی این شیوه‌ی احراز هویت انجام می‌شوند را بیان می‌کنیم.

۱.۱.۳ تولید کلید مخفی

سرور میزبان به‌جای ارسال مستقیم کد تولید شده برای کاربر، از گوشی همراه وی برای ارسال کلید مخفی به Google Authenticator استفاده می‌کند. بنابراین سرور باید به‌ازای هر کلاینتی که می‌خواهد وارد شود، یک کلید مخفی تولید کند. طول این کلید بسته به پروتکل‌های مختلف می‌تواند متنوع (۸۰بیت، ۶۴بیت و یا ۱۲۸ بیت) باشد و همچنین در صورت نیاز می‌توان از الگوریتم‌های لایه‌زنی [۱۵] برای افزایش طول آن استفاده کرد.

۲.۱.۳ تولید QR Code

سرور همچنین یک QR Code برای کاربر تولید می‌کند که فرمت محتوای `URI` آن به صورت زیر است:

```
otpauth://{type}/{app}:{accountName}?secret={secret}{query}
```

- `type`: می‌تواند `totp` یا `hotp` باشد که نشان‌دهنده‌ی الگوریتم درهم‌سازی است.
- `app`: نام اپلیکیشن یا وب‌سایتی است که احراز هویت دو عاملی برای آن قرار است انجام شود.

[11] Open-Source
[12] Time-based One-Time Password
[13] HMAC-based One-Time Password
[14] OTP or One-Time Password
[15] Padding

- `accountName`: نام کاربری یا ایمیل کاربر که به واسطه‌ی آن می‌خواهد به برنامه وارد شود.
- `secret`: همان کلیدی است که سرور برای کاربر تولید کرده‌است.
- `query`: سرور می‌تواند در این پارامتر موارد بیش‌تری مانند تعداد ارقام کد TOTP را ارسال کند تا کد تولیدشده شخصی‌سازی‌شده‌تر باشد.

۳.۱.۳ اسکن QR Code

در این مرحله کاربر نرم‌افزار Google Authenticator را باز کرده و برای وارد کردن اطلاعات برنامه‌ای که می‌خواهد احراز هویتش را انجام بدهد، QR Codeای که برنامه در مرحله‌ی قبل تولید کرده‌است را اسکن می‌کند. بدین ترتیب اطلاعات از سمت سرور میزبان به Google Authenticator می‌رسد و کلید مخفی بین این دو به اشتراک گذاشته‌شده است. اکنون Google Authenticator هر ۳۰ ثانیه یک کد چند رقمی تولید می‌کند. (تا ابد این کار را ادامه می‌دهد!)

۴.۱.۳ تایید کد و کلید مخفی

اکنون سرور میزبان باید یک محل برای وارد کردن کد دریافت شده از Google Authenticator را در اختیار کاربر قرار بدهد. اکنون سرور میزبان و نرم‌افزار Google Authenticator کلید یکسانی در اختیار دارند. سرور بعد از وارد کردن کد توسط کاربر، الگوریتم درهم‌سازی را اجرا می‌کند و به یک کد می‌رسد. اگر این کد با کدی که کاربر وارد کرده‌است یکسان باشد، پس هویت کاربر برای اولین بار تایید شده و سرور در سمت خودش کلید مخفی را در پایگاه داده‌اش ذخیره می‌کند. از این لحظه به بعد، دیگر نیازی به به‌اشتراک‌گذاری کلید نیست و کاربر برای ورود می‌تواند به راحتی کد یک‌بار مصرف و موقتی که در Google Authenticator تولید می‌شود را برای ورود در سمت برنامه‌ی مورد استفاده‌اش وارد کند و وارد سیستم شود.

۲.۳ TOTP

برای درک بهتر چگونگی تولید کدها در عملیات‌هایی که شرح دادیم، الگوریتم TOTP را معرفی اجمالی و کوتاهی می‌کنیم.

۱.۲.۳ پیشینه: HOTP

همان‌طور که در RFC4226 تعریف شده‌است، الگوریتم HOTP برپایه‌ی الگوریتم درهم‌سازی HMAC-SHA-1 (که در RFC2104 تشریح شده‌است.) است و روی مقدار یک شمارنده‌ی در حال افزایش به‌عنوان پیام ورودی HMAC اعمال می‌شود.

در حقیقت، خروجی محاسبات HMAC-SHA-1 کوتاه می‌شود تا یک مقدار متناسب با نیاز کاربر به‌دست بیاید:

```
HOTP(K,C) = Truncate(HMAC-SHA-1(K,C))
```

که Truncate تابع کوتاه‌ساز کد و K و C به‌ترتیب کلید و پیام (مقدار شمارنده) هستند.

۲.۲.۳ الگوریتم TOTP:

TOTP تعمیم HOTP بر پایه‌ی زمان است. به این‌گونه که پارامتر T که از یک منبع زمانی و یک گام زمانی به‌دست می‌آید جایگزین C می‌شود.

```
TOTP = HOTP(K, T)
T = (Current Unix time - T0) / X
```

- `X`: نمایانگر واحدهای زمانی به ثانیه است. (مقدار پیش‌فرض آن ۳۰ ثانیه است.)
- `T0`: Unix timeای است که از لحظه‌ی شمردن شروع شده‌است. بنابراین یک پارامتر سیستمیست.
- `Current Unix time`: مفهوم Unix time، تعداد ثانیه‌هاییست که از شروع روز ۱ ژانویه‌ی سال ۱۹۷۰ میلادی می‌گذرد. این مقدار در تمام سیستم‌عامل‌های برپایه‌ی Unix به‌عنوان یک شمارنده‌ی ۳۲ بیتی قرار دارد. (به‌زودی سرریز خواهد کرد!) بنابراین این پارامتر، Unix time همان لحظه را نشان می‌دهد. برای مثال در لحظه‌ی نوشتن این جمله، این مقدار 1720265627 است!

۳.۲.۳ پیاده‌سازی

یک نمونه از پیاده‌سازی الگوریتم TOTP در زبان جاوا را در ادامه آورده‌ایم. برای مشاهده‌ی بیش‌تر می‌توانید به www.openauthentication.org مراجعه کنید.

```
/**
Copyright (c) 2011 IETF Trust and the persons identified as
authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, is permitted pursuant to, and subject to the license
terms contained in, the Simplified BSD License set forth in Section
4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info).
*/

import java.lang.reflect.UndeclaredThrowableException;
import java.security.GeneralSecurityException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.math.BigInteger;
import java.util.TimeZone;

/**
 * @author Johan Rydell, PortWise, Inc.
 */

public class TOTP {

    private TOTP() {}

    private static byte[] hmac_sha(String crypto, byte[] keyBytes,
            byte[] text){
        try {
```

```
            Mac hmac;
            hmac = Mac.getInstance(crypto);
            SecretKeySpec macKey =
                new SecretKeySpec(keyBytes, "RAW");
            hmac.init(macKey);
            return hmac.doFinal(text);
        } catch (GeneralSecurityException gse) {
            throw new UndeclaredThrowableException(gse);
        }
    }

    private static byte[] hexStr2Bytes(String hex){
        // Adding one byte to get the right conversion
        // Values starting with "0" can be converted
        byte[] bArray = new BigInteger("10" + hex,16).toByteArray();

        // Copy all the REAL bytes, not the "first"
        byte[] ret = new byte[bArray.length - 1];
        for (int i = 0; i < ret.length; i++)
            ret[i] = bArray[i+1];
        return ret;
    }

    private static final int[] DIGITS_POWER
    // 0 1  2   3    4     5      6       7        8
    = {1,10,100,1000,10000,100000,1000000,10000000,100000000 };

    public static String generateTOTP(String key,
            String time,
            String returnDigits){
        return generateTOTP(key, time, returnDigits, "HmacSHA1");
    }

    public static String generateTOTP256(String key,
            String time,
            String returnDigits){
        return generateTOTP(key, time, returnDigits, "HmacSHA256");
    }

    public static String generateTOTP512(String key,
            String time,
            String returnDigits){
        return generateTOTP(key, time, returnDigits, "HmacSHA512");
    }

```

```
    public static String generateTOTP(String key,
            String time,
            String returnDigits,
            String crypto){
        int codeDigits = Integer.decode(returnDigits).intValue();
        String result = null;

        // Using the counter
        // First 8 bytes are for the movingFactor
        // Compliant with base RFC 4226 (HOTP)
        while (time.length() < 16 )
            time = "0" + time;

        // Get the HEX in a Byte[]
        byte[] msg = hexStr2Bytes(time);
        byte[] k = hexStr2Bytes(key);

        byte[] hash = hmac_sha(crypto, k, msg);

        // put selected bytes into result int
        int offset = hash[hash.length - 1] & 0xf;

        int binary =
            ((hash[offset] & 0x7f) << 24) |
            ((hash[offset + 1] & 0xff) << 16) |
            ((hash[offset + 2] & 0xff) << 8) |
            (hash[offset + 3] & 0xff);

        int otp = binary % DIGITS_POWER[codeDigits];

        result = Integer.toString(otp);
        while (result.length() < codeDigits) {
            result = "0" + result;
        }
        return result;
    }

}
```

۴ مزایای احراز هویت با گوگل

معمولا ابزارهای نرم‌افزاری که معرفی می‌شوند و مورد استقبال عمومی قرار می‌گیرند و به توسعه ادامه می‌دهند، نیاز مهمی را در دنیای فناوری برآورده می‌کنند. نرم‌افزار Google Authenticator نیز از این قاعده مستثنی نیست و نیازهای مهم و متنوعی را برآورده می‌کند که در ادامه به چند مورد از آن‌ها اشاره‌ی کوتاهی می‌کنیم.

۱.۴ دسترسی سریع و یکپارچگی

پیش از مطرح شدن ایده‌ی احراز هویت با گوگل، هر سامانه‌ای باید به‌طور جداگانه احراز هویت چند عاملی را پیاده‌سازی می‌کرد و مستقیما با کاربر ارتباط برقرار می‌کرد. با حضور Google Authenticator، پروتکل‌های این نوع از احراز هویت پر صورت پشتیبانی سیستم یکپارچه می‌شوند و استفاده از آن بسیار ساده و در دسترس است و نیازی به پیاده‌سازی مجدد کدهای موجود نیست! همچنین از دیدگاه کاربر، تمامی رمزهای یک‌بار مصرفی که نیاز دارد در یک برنامه‌ی واحد جمع‌آوری شده‌است و مدیریت و استفاده از آن‌ها راحت‌تر است.

۲.۴ استفاده‌ی برون‌خط

یکی از مهم‌ترین فواید این نرم‌افزار این است که در هر شرایطی و تنها با دسترسی به یک گوشی همراه قابل استفاده است. حتی اگر دستگاه شما به اینترنت متصل نباشد یا در حالت هواپیما قرار داشته باشد می‌توانید به‌راحتی از این برنامه استفاده کنید و اختلالی به‌وجود نمی‌آید. کاربر فقط برای اولین ورود نیاز به اینترنت دارد و این نوآوری باعث افزایش راحتی، دسترس‌پذیری و امنیت بیش‌تر احراز هویت شده‌است.

۳.۴ حمله‌ی مرد میانی

یکی از حملات رایجی که در روش سنتی احراز هویت چند عاملی می‌تواند انجام شود، حمله‌ی مرد میانی [۱۶] است. به این شکل که حمله‌کننده در میانه‌ی ارتباط سرور و کلاینت قرار می‌گیرد و هویت خودش را به‌جای هر کدام برای دیگری جعل می‌کند. به‌همین دلیل اگر کد احراز هویت مستقیما از سرور به کلاینت ارسال شود، می‌تواند این ارتباط را شنود کند و کد را به‌همراه فاکتورهای دیگر احراز هویت برای سرور ارسال کند و به اطلاعات کاربر دسترسی داشته باشد. اگرچه در هر ارتباطی در شبکه لازم است امنیت کانال‌های ارتباطی تامین شود، اما در طراحی سیستم باید انواع نفوذپذیری‌های ممکن را در نظر گرفت. با توجه به این‌که احراز هویت با گوگل به‌صورت برون‌خط [۱۷] انجام می‌شود، به‌جز مرحله‌ی اولیه‌ی ورود، دیگر نیازی به حفاظت در برابر حمله‌های میانه‌ی شبکه نیست.

۴.۴ حمله‌ی فیشینگ

حمله‌ی فیشینگ، [۱۸] با ایجاد یک سامانه‌ی جعلی کاملا مشابه از لحاظ ظاهری با سامانه‌ی وقعی از سوی مهاجم انجام می‌شود. اگر کاربر متوجه غیرواقعی بودن وب‌سایت نباشد، می‌تواند ناخواسته مراحلی را طی کند که مهاجم قصدشان را دارد و منجر به آسیب‌های بزرگی می‌شود. ارسال کدهای احراز هویت دو عاملی به روش سنتی هم می‌تواند ظاهری مشابه با سیستم واقعی داشته باشد و همچنان کاربر متوجه این حمله نشود. بنابراین یک راه‌حل مطمئن می‌تواند استفاده از احراز هویت گوگل باشد و بعد از تبادل کلید مخفی با سایت اصلی دیگر نگرانی رخ دادن حمله‌ی فیشینگ وجود نداشته باشد.

۵ ملاحظات امنیتی

در طراحی تمام سیستم‌های نرم‌افزاری، همواره یک داد و ستد میان افزایش امنیت سیستم و راحتی و دسترس‌پذیری استفاده از آن وجود دارد. در استفاده و پیاده‌سازی احراز هویت گوگل نیز آسیب‌پذیری‌هایی وجود دارند که در طراحی سامانه‌های خود باید آن‌ها را در نظر داشته باشیم.

[16] Man-in-the-middle attack
[17] Offline
[18] Phishing

۱.۵ خطر افشای کلید

می‌توان گفت مرحله‌ی تبادل کلید مخفی میان سرور میزبان و نرم‌افزار کلاینت، مهم‌ترین گام ایجاد ارتباط میان این دو است. اگر کلید مخفی‌ای که ارسال می‌شود را یک شنودگر به‌طور غیرمجاز به‌دست بیاورد، ممکن است با ترکیب آسیب‌پذیری‌های دیگر بتواند به‌جای کاربر همواره وارد سیستم شود و سرور و کلاینت هیچ‌یک متوجه این موضوع نشوند. بنابراین خطر افشای کلید یکی از مهم‌ترین نفوذهای امنیتی برنامه است.

۲.۵ حمله‌ی آزمون فراگیر

از آن‌جایی که تعداد ارقامی که کد احراز هویت تولید می‌کند کوتاه و محدود است، بنابراین فضای حالت حمله‌ی آزمون فراگیر [۱۹] بسیار کوچک است. اگر در پیاده‌سازی سیستم میزبان، امکان وقوع این حمله را در نظر نگیریم و پیش‌نیازهای بسیار ساده برای جلوگیری از آن را فراهم نکنیم، (مثلا محدود کردن تعداد درخواست‌ها یا افزایش زمان مجاز میان درخواست‌ها) ممکن است عواقب سنگینی برای صاحب سامانه داشته باشد.

۳.۵ حمله‌ی تکرار

یکی دیگر از حملات رایجی که در ارتباطات شبکه‌ای انجام می‌شود، حمله‌ی تکرار [۲۰] است. به بیان ساده این حمله این‌گونه است که مهاجم در میانه‌ی راه کاربر و میزبان، اطلاعاتی که کاربر ارسال می‌کند را دریافت و ذخیره می‌کند. بعد از گذشتن زمان مناسب، دقیقا همان داده‌ها را از سوی خودش به میزبان ارسال می‌کند. بنابراین اگر با تمام شدن ارتباط و نشست [۲۱] میان میزبان و کاربر، اطلاعات اعتبارسنجی و تایید هویت کاربر را منقضی نکنیم، می‌تواند آسیب‌پذیری برای این حمله را فراهم کند.

۶ به‌کارگیری عملی Google Authenticator

برای آن‌که بخش عملی و کاربردی استفاده از ابزار احراز هویت گوگل را در کنار تئوری نحوه‌ی عملکرد آن داشته باشیم، در ادامه راهنمای گام‌به‌گام فعالسازی این ابزار برای سیستم حساب‌های گوگل را نشان داده‌ایم. دقت کنید که ممکن است با به‌روزرسانی برنامه‌ها، رابط کاربری آن‌ها نیز عوض شود و تغییراتی در مراحل ذکرشده به‌وجود بیاید. این ابزار را می‌توان در هر سیستمی که از آن پشتیبانی کند (که امروزه اکثر سایت‌های معتبر این پشتیبانی را انجام می‌دهند) می‌توان به کار برد. ما در تصاویر و توضیحات زیر آن را برای تمام ابزارهای متصل به حساب گوگل فعال می‌کنیم.

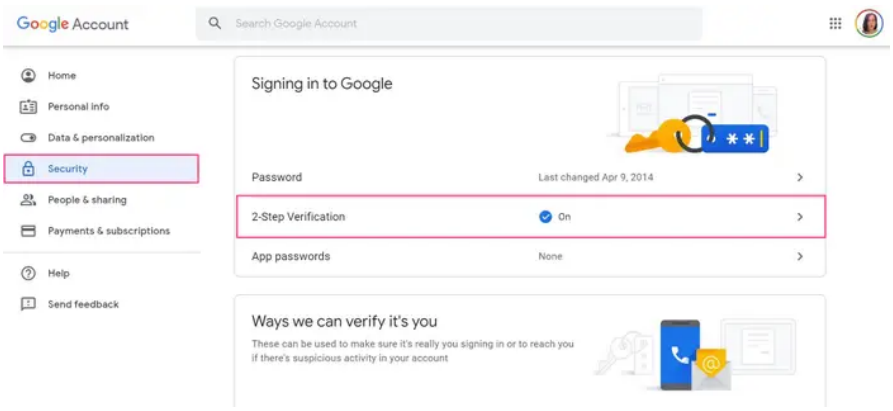
۱. نرم‌افزار Google Authenticator را بر روی گوشی همراه خود نصب کنید.

۲. در حالی که در حساب گوگل خود روی کامپیوتر شخصی‌تان وارد شده‌اید، روی آیکون کوچکی که عکستان را نشان می‌دهد کلیک کرده و سپس گزینه‌ی Manage your Google account را انتخاب کنید.

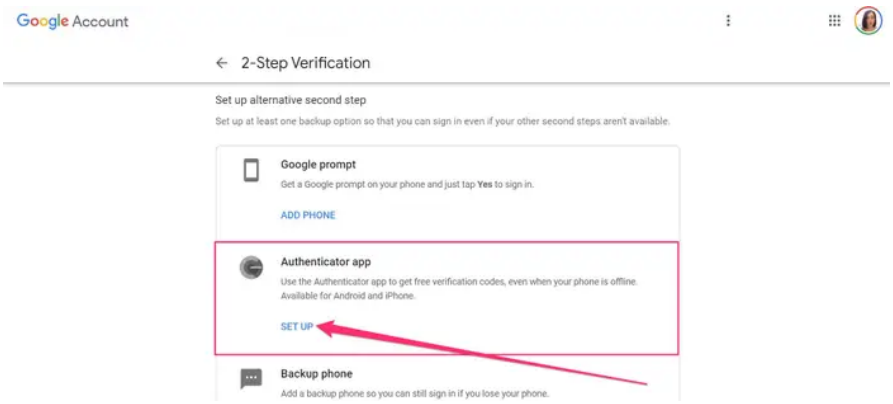
۳. از منوی سمت چپ صفحه، روی گزینه‌ی Security کلیک کنید و تا بخش Signing in to Google پایین بیایید.

۴. گزینه‌ی 2-Step Verification را روشن کنید. از شما خواسته می‌شود تا رمز خود را مجددا وارد کنید تا بتوانید ادامه دهید.

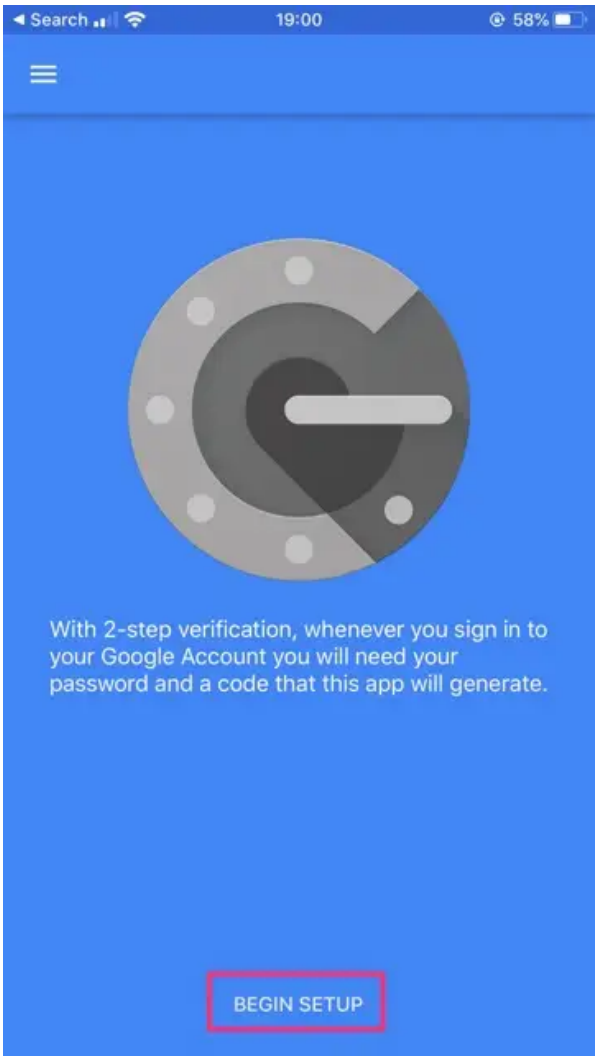
[19] Brute-force attack
[20] Replay attack
[21] Session



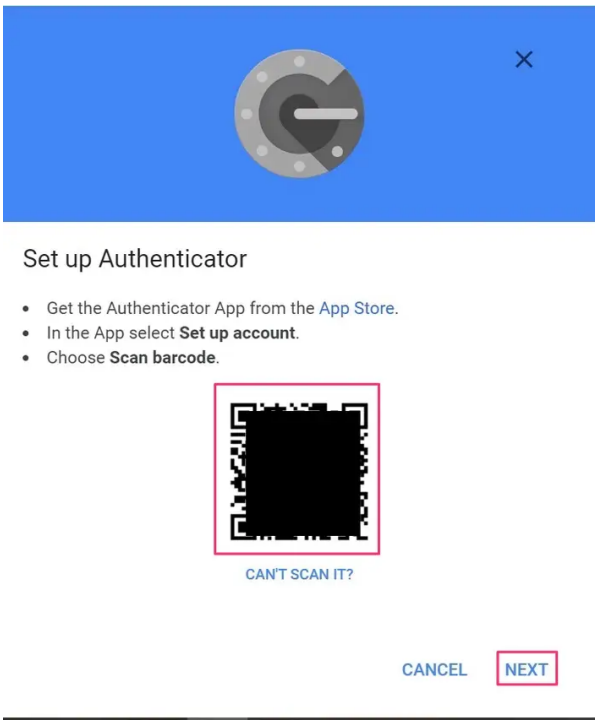
۵. ذیل بخش Set up alternative second step، دکمه‌ی Set Up را در گزینه‌ی Authenticator app انتخاب کنید.



۶. نرم‌افزار Google Authenticator را بر روی گوشی همراه خود باز کنید و اگر تا به حال از آن استفاده نکرده‌اید، دکمه‌ی Begin setup را بزنید. در غیر این‌صورت فقط یک برنامه‌ی جدید به برنامه‌های موجود اضافه کنید.



۷. گزینه‌ی Scan barcode را روی گوشی خود بزنید و سپس QR Codeای که روی صفحه‌ی کامپیوتر شخصی‌تان نمایش داده‌شده‌است را اسکن کنید.



۸. اکنون اپلیکیشن احراز هویت در گوشی شما به‌طور خودکار کدهای موقتی ۶ رقمی تولید می‌کند. بعد از این‌که دکمه‌ی Next را در کامپیوترتان زدید، کد نشان‌داده‌شده در گوشی همراهتان را وارد کنید. بعد از تایید، احراز هویت انجام می‌شود. هر بار که بخواهید احراز هویت مجدد انجام دهید، کافیست کد نمایش داده‌شده برای برنامه‌ی دلخواهتان را وارد کنید.

۷ نتیجه‌گیری

پیاده‌سازی و استفاده از Google Authenticator نشان‌دهنده‌ی پیشرفت‌های قابل توجه در امنیت وب از طریق احراز هویت دو عاملی است. با به‌کارگیری لایه‌های متعدد احراز هویت، این سیستم، امنیت حساب‌های کاربری را افزایش می‌دهد و دسترسی غیرمجاز را به‌طور قابل توجهی دشوارتر می‌سازد. این تحقیق اثربخشی احراز هویت دو عاملی را در کاهش تهدیدات امنیتی رایج مانند حملات جستجوی فراگیر، فیشینگ و حملات مرد میانی برجسته می‌کند و یک دفاع قوی‌تر در برابر نفوذهای احتمالی فراهم می‌کند.

علاوه بر این، سهولت استفاده و قابلیت‌های آفلاین Google Authenticator، بدون کاهش امنیت، راحتی بیش‌تری را فراهم می‌کند. کاربران می‌توانند رمزهای یکبار مصرف مبتنی بر زمان (TOTP) را حتی بدون اتصال به اینترنت تولید کنند و حفاظت مداوم را تضمین کنند. این ویژگی به‌ویژه در سناریوهایی که دسترسی به اینترنت غیرقابل اعتماد یا غیرقابل دسترس است، مفید است. به‌طور کلی، این تحقیق نقش حیاتی احراز هویت دو عاملی و ابزارهایی مانند Google Authenticator را در حفاظت از هویت‌های دیجیتال و بهبود امنیت کلی برجسته می‌کند.

۸ مراجع

- RFC 4949
- RFC 6238
- RFC 4226
- Github - Google Authenticator
- اسلایدهای درس امنیت داده و شبکه - دانشگاه صنعتی شریف - دکتر خرازی، دکتر امینی
- TechTarget - Google Authenticator
- Wikipedia - Google Authenticator
- Eye on Tech Youtube Channel - What is Two-Factor Authentication (2FA)?
- Medium - How does Google Authenticator work?
- Wikipedia - Unix time
- Business Insider - What is Google Authenticator?