ACCOUNT SECURITY BEST PRACTICES GUIDE

Protect your users at sign up, through authentication to recovery

Fraudulent and fake accounts in your application can cost you anywhere from a few hundred dollars to tens of thousands per account. High risk accounts, such as those used for online finances, are frequent targets for takeover; and the financial impact from hacking is often tens of thousands of dollars or more. When application developers then overcompensate by poorly implementing additional security features, it results in users being locked out of accounts and contacting your support team, raising costs and frustrating users.

Every user that signs up for your application needs to be properly verified, adequately authenticated, and be given a safe and user-friendly path to recovery to avoid costly support tickets. Failure to adequately address these account security issues can result in unnecessary loss of revenue for the service providers and personal financial losses for their users.

According to an Identity Fraud Study released by Javelin Strategy & Research, in 2016, account takeovers —where stolen login information is used to access a consumer's accounts— rose 31 percent. Instances where fraudsters opened new accounts in a consumer's name were up 20 percent, compared to 2015. The overall cost to consumers was more than $16 billion. In financial institutions, the loss as a result of a takeover is typically $10,000 to $15,000 per account. Each password reset typically costs support teams $15-$50. If 20% of your users reset their password a month, when you triple your active users, you also triple these support costs.

Expenses related to users grow as your business expands. There are three critical user lifecycle events you should examine to reduce costs, increase security and limit user friction. They are registration, authentication, and recovery, and it's important to address problems sooner rather than later.

What follows are best practices for ensuring the account security of users in your application is well implemented. First, you should be asking yourself three questions:

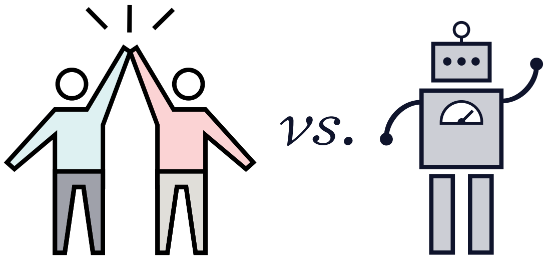
1. **Are our methods to verify new users working?**
2. **Are we doing everything we can to secure our users' information?**
3. **Do we make it easy for our users to recover access to their accounts if they forget or misplace their login credentials?**

Each interaction with your users is an opportunity to provide a positive experience that instills trust. Let's look at the potential costs of failing to address more sophisticated attacks – and how some companies have answered these questions.

Are our methods to verify new users working?

Ensure the account was created by a human, not a bot

The success of any online application is usually measured by user signups and how active those users are, also known as MAU's (monthly active users). Every application owner strives to have as many accounts created as possible, with users interacting with their business as often as possible. This leads to the design of simpler and faster sign up flows; but this presents a problem. How do you know a user registration is legitimate while maintaining an effective signup process?



an email asking you to click a link and confirm your identity before using your newly minted account. This has been the common practice for years. The problem: email addresses are easy to create. Signing up for a Gmail, Yahoo, or Hotmail account is relatively easy. Even easier is registering a domain and creating hundreds of fake email accounts.

Recommended best practice

Twitter and Facebook, two of the platforms with both the largest communities of users and the largest problem with fake accounts, moved away from using just email verification and have started to ask users for their phone number instead. During the account creation phase, a code is sent via SMS to the user's phone number and if they successfully verify the code within the application, the account can be created. You may be wondering, why is a phone number better than an email? Getting an email address can take seconds, getting a phone number is much harder, and it almost always incurs a cost. Even a cheap pay-as-you-go SIM will cost around $10. So, a fraudster trying to spam your application with thousands of fake accounts is going

Phone numbers are much more effective to verify the legitimacy of a user because there is a lot more data behind them. It is possible to look up a phone number and find out if it's tied to a contract or if it's pay-as-you-go, if it's a mobile or landline, or if it's VoIP.

Twitter, Facebook, and YouTube have all struggled with the problem of fake user accounts in their platform. The impact? Reduced trust from users inundated with fake accounts and compounding costs from seeking out and removing them. Worse still are applications where bad actors can make financial gains by signing up as false users, abusing free trials, and repeating such actions over and over again.

Verifying email addresses has been the common practice for years. You've undoubtedly been sent

to find it much harder when it's $10 per attack. Twitter also uses phone verification to track banned users. If someone attempts to sign up with a new account, asking for their phone number makes it easier to associate banned accounts with new sign up attempts.

There is also a lot more data behind a phone number than email and it can be used to determine the legitimacy of the user. For example Yahoo doesn't allow you to use a Voice Over Internet Protocol (VoIP) phone number. VoIP numbers are easier to obtain and often pose a greater risk.

Data such as this can be used to make informed decisions at the time of sign up. For example, is your application hosting very high risk accounts where the loss of data could be in the thousands? Then you would likely only accept sign ups from users with real physical phones.

Two-factor authentication is the quickest way to move away from using passwords alone to protect accounts. Push authentication allows you to present a simple "Accept" or "Deny" option to users to authenticate a transaction or access to an account.

Even as malicious actors develop new methods to abuse your application, this more secure verification method can prove more expensive and time-consuming to beat than email verification alone.

Are we doing everything we can to secure our users' information?

Passwords have been proven to be ineffective

A week doesn't pass by when some company is in the news due to a data breach. The Javelin fraud report also says that so far, in 2017 alone, there have been 1,140 breaches, surpassing the 2016 record of 1,093. These data breaches often expose user account details and in many instances user passwords. So we all ask users to reset passwords, choose stronger ones and not reuse passwords between applications. Worse still, often the answer to fighting poor passwords is asking users to create and remember complex ones, which leads to people constantly forgetting them or having to pay for a password manager. While longer, more complex passwords and password managers do help, we really shouldn't be pushing this problem onto the user. Let's face it: Anyone with more than a few online accounts is tired of dealing with passwords. We need to make the process of authentication better. The easier it is for a user to sign in without issues, the more likely they are to use your service.

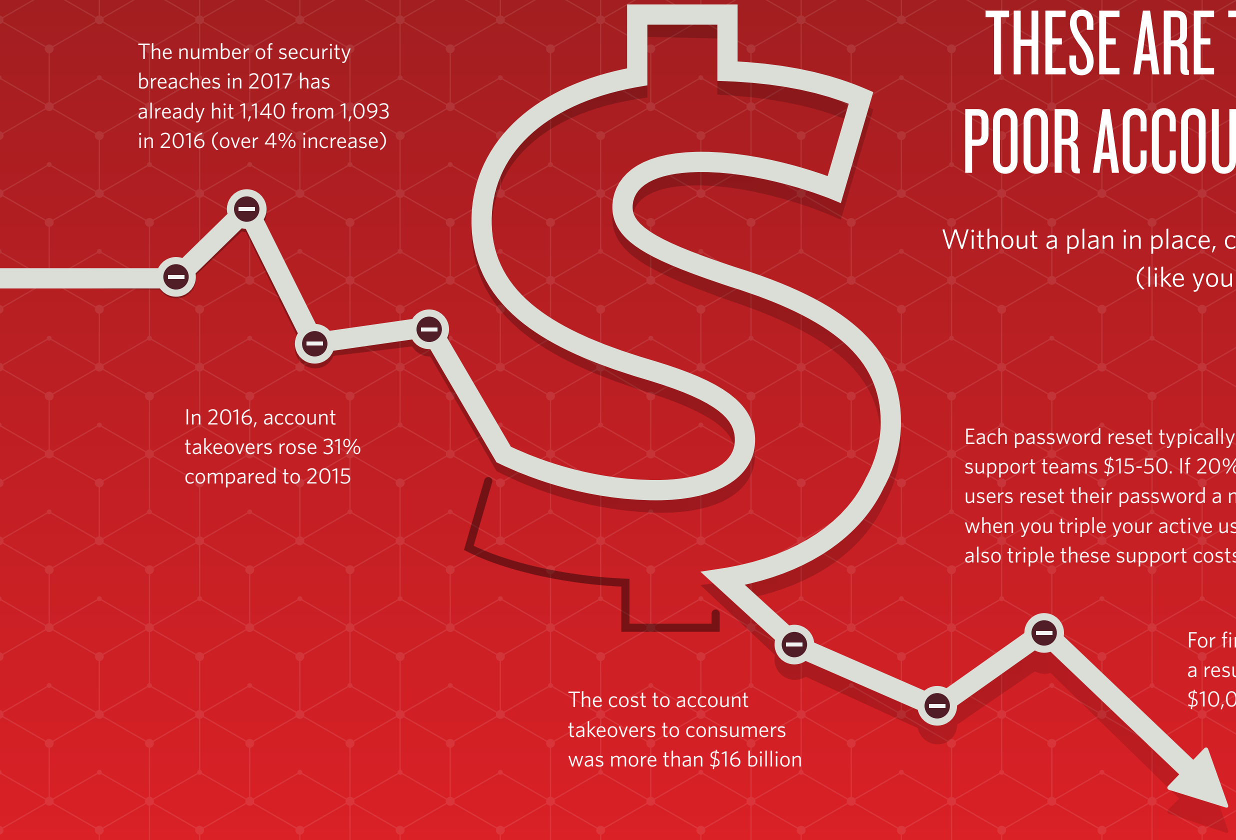
Recommended best practice

Implementing two-factor authentication (2FA) is the quickest way to increase the security of your user accounts and remove the pressure on an ever increasing battle on password complexity. This typically takes the form of a one-time password sent via SMS or voice call at the time of authentication. The user then re-types this code into your application, so even if an attacker has access to a user's password, they also need to have access to their phone, something that's much harder to attain. When a user is offline or roaming, or in a region where SMS and voice calls are not possible, there are mobile and desktop apps which can generate the same code, making sure the solution works in all scenarios.

The most modern form of 2FA is push authentication, where a notification is sent to a user's mobile device or desktop and is presented with a simple "Accept" or "Deny". This removes the hassle of having to read and retype a code, and the notification usually displays information such as the location where the request was originated. This method has gained significant traction in the last few years. Google, Yahoo, Microsoft, Apple, SalesForce, Namecheap, Capital One and Dropbox have all implemented push authentications in their services. Google, a leader in authentication best practices, has recently made push authentication

THESE ARE THE COSTS OF POOR ACCOUNT SECURITY:

Without a plan in place, customers and companies (like yours) are left to foot the bill.



The number of security breaches in 2017 has already hit 1,140 from 1,093 in 2016 (over 4% increase)

In 2016, account takeovers rose 31% compared to 2015

The cost to account takeovers to consumers was more than $16 billion

Each password reset typically costs support teams $15-50. If 20% of your users reset their password a month, when you triple your active users, you also triple these support costs

For financial institutions, the loss as a result of a takeover is typically $10,000 to $15,000 per account

Source: Javelin Strategy & Research 2017 Identity Fraud Study

(which they call Google Prompt) the default 2FA method over SMS, due to increasing concerns over the security of using SMS for 2FA. Google and Yahoo have taken the idea a step further and have replaced the use of a password at login, making the user's phone the primary mechanism for logins.

These tech giants have the resources to build big development teams to implement their own solutions. You can use Google and Yahoo accounts to authenticate your users, however, you will trade-off the ability to brand the login experience and have full visibility and control over the authentication flow and logic. Some would also argue that there are privacy issues with relying on these big companies owning your user's identity. Thankfully, there are API and SDK services on the market that allow you to embed the same 2FA features into your application with ease. And with the added benefit that you still retain ownership of the user's identity.

Account recovery provides yet another opportunity to solidify trust with your users. Since most likely you already have your user's phone number, this should be the fastest way to reestablish access to an account.

Do we make it easy for our users to recover access to their accounts if they forget or misplace their login credentials?

Being locked out of your account is no fun

We've said this already: people forget passwords. And when they make numerous attempts to login, they end up locked out. Truth be told, account access recovery hasn't been an area of focus for developers. Quick and easy signups–check. Better authentication–check. Innovation in recovery methods... not so much. And the costs associated with this problem are typically high. Locked out users are often irate and frustrated, and tired of answering questions to legitimize their account access. Using call centers and email to manage account support is not effective.

As with everything in security, you'll need to strike the right balance of risk versus user experience. Recovering access to your account is often the most critical time to balance these two. You want to ease the pain of already frustrated users, but if you make the process too easy, potential attackers will have easy access to reset passwords and take over accounts.

The act of recovery is really one of authentication, except the user no longer has the necessary login credentials. There are a range of methods by which you can help a user recover access to their account. The most common is via email, which as mentioned before, is easy but often open to abuse. Once an attacker has access to a user's email, it's trivial to then go resetting passwords for all other accounts they own because they can just respond to the password reset email.

Another method is to present the user with a recovery code at the point of account creation, asking them to write it down and save to a safe place. In reality, nobody does this. Here again, users are having to take the burden of effort in securing their own accounts. A similarly frustrating method to verify identity involves security questions. At account creation time, the user provides answers to a range of security questions. They then answer a selection of these when they forget their password. However, the vast amount of personal information that people willingly share via social media, or not (in the case of a data breach), all but renders this approach obsolete.

Recommended best practice

Yet again, phone numbers are a much better way to verify users. The solution can be quite simple and is a very similar process to the one used for verification. Except this time, you don't need to ask for the phone number, because you already have it. What's very important is to not reveal the full phone number during the recovery process. If the recovery attempt is requested illegitimately, you could potentially be giving an attacker the phone number for the victim. A better approach is to ask for the last 4 digits of the phone number or to send an SMS, mask off all but the last 4 digits, so the user can confirm the correct phone number.

To maintain user trust, it's also important to configure the numbers you send the SMS from by using short codes (Twitter has 404-04 and Amazon uses 262-966) instead of a generic long code (e.g. 415 123 4567). A short code tells a user the message is coming from a legitimate service and can further deter attackers from impersonating your business and getting users to send recovery codes to them.

As with everything in security, you'll need to strike the right balance of risk versus user experience.

What to do next? 4 simple steps to improve account security

If you find that you need to improve your account security in any of the areas mentioned above, we have some more advice on possible next steps:

Buy, don't try to build

Today, the technology landscape is full of solutions perfected to tackle a specific problem. **While it's tempting to think you can build in-house all of the parts of account security, don't.** Spend your engineering time and resources on adding value to your application not re-inventing the wheel. Look for a product that helps you easily implement phone verifications, including built-in localization to deliver messages to your customers in the local language. The phone numbers used to send codes are also optimized to ensure high delivery rates, hiding the complexity of regional telecommunications. The price of these products easily outweigh the time and effort your developers would spend to build, maintain, secure, and improve on their own creations.

Work hard to maintain the balance

Make sure that all your efforts to secure your business and your users are balanced against making the use of your application easy and smooth. **It's no good adding the most secure functionality to your application if it's too hard to use.** A bad user experience will surely turn people off and away from your business. Be smart about how you present any security questions or steps, always trying to help the user make the right decision and setting defaults that are secure, but also easy to navigate.

APIs give you the most freedom

The future is cloud, and the cloud is built on APIs. **Implementing an API allows you to embed pre-built solutions directly into the flow of your applications.** You can customize the entire user experience, yet retain the value of buying a solution with all the security and telecommunications logic solved for you. The important things to look for in an API are detailed and up-to-date documentation combined with an active community across many platforms.

Watch this video, then follow our blog

Here at Twilio, we take account security seriously and we know a lot about it. Recently, our head of product for account security, B Byrne, spoke at a conference about the ROI for using account security APIs. If you need help convincing your decision makers to improve account security in your application, watch this video for some good arguments to make. Then, keep an eye on our blog for more articles about account security.