Salesforce Authenticator

Fast, Frictionless Two-Factor Authentication

Whitepaper, Spring ’16

CONTENTS

The Importance of Effective Two-Factor Authentication

It is impossible to overemphasize the importance of increasing and preserving trust in our online systems. As threats to enterprise data and business processes continue to evolve, it's imperative that a broad defence-in-depth strategy is deployed, covering both the human and technical components of security. One critical component of this strategy is effective authentication. Authentication helps us keep sensitive information safe from advanced digital threats while still accessible to authentic users.

Unfortunately, as the online security environment has grown more threatening, authentication processes have grown more complex. As authentication processes have gotten stronger, new methods have traditionally introduced more security at the expense of user experience. Whether by modifying a user's behaviors or adding new steps, changes to user experience increase friction and reduce adoption. Until now, the paradigm has been that more security means more friction for users.

Salesforce Authenticator breaks this paradigm, by introducing new authentication capabilities for increasing trust in our systems, with usability designed for broad deployment. With authentication that is fast, user-friendly, and under certain circumstances invisible, enterprises can benefit from the critical protections of two-factor authentication, without the traditional friction and usability challenges.

What Is Two-Factor Authentication?

Two-Factor authentication is some combination of three things:

- Something you know
- Something you have
- Something you are

For example, a common use of two-factor authentication happens during an ATM withdrawal. It requires something you have (your ATM card) and something you know (your PIN). If you (or an imposter) only has one of these two elements, then it's not enough. Biometric authentication, such as Touch ID on the iPhone, has also emerged as a way of adding a crucial 'something you are' component to multifactor authentication.

By requiring more than one factor during an authentication process, we can greatly increase the assurance we have that we are dealing with the right individual. While there is risk that a single factor such as a password may be compromised, requiring a second factor can effectively mitigate this risk.

Note: To be considered two-factor or multifactor authentication, your factors must come from different categories. For example, using a password and answering challenge questions upon login is not technically two-factor authentication, as each factor is 'something you know.'

Why Not a Single-Factor?

At one point, a single-factor of authentication, the password, was considered sufficient for digital safety. As online threats evolved, so did the use of passwords, which in turn began to amplify their limitations.

Increases in password complexity and rotation requirements have made passwords too difficult for users to manage effectively. This has caused users to reuse the same password across many (or all) of their accounts, write down their passwords, and/or share their passwords with other users. In addition, the popularity of passwords has led to hacking techniques such as phishing and brute force attacks that exploit these limitations.

But for all the password-bashing that exists, it is difficult to chide passwords too much when in truth, any single factor -- a phone, a thumbprint, a PIN -- is weak security. Security professionals understand that one factor simply is not strong enough, and that it's critical to deploy multiple factors.

Popular Two-Factor Authentication Approaches and their Limitations

Today, there is a wide variety of options for two-factor authentication. Many enterprises use some method of two-factor authentication to protect their workforce and users from data breaches and identity theft. The advances of two-factor authentication yield security benefits that add value to security professionals and their users, however each has a variety of limitations.

SMS Authentication

Beginning in the early '90s and accelerating in adoption when mobile phones reached ubiquity, SMS-based authentication adds a second factor to the password authentication model. Instead of entering a password alone (something you know), users also receive a one-time passcode on their mobile device (something you have) that they enter into their browser to gain access. While certainly a step up from single-factor authentication, SMS based authentication suffers from the follow limitations.

Insecure channel: SMS is not an inherently secure channel. SMS can be intercepted by bad actors, users' phones can suffer from porting attacks, etc.
Vulnerable: When using SMS delivered one-time passcodes, the user takes the code from the message and places it back into the login interface. This leaves users vulnerable to phishing and other endpoint attacks. In addition, users are increasingly configuring their mobile devices to forward SMS messages to other devices, which increases the risk of interception.
Slow process: Users must wait for a text every time they log in or need a code, which slows the process. In addition, lack of a cell connection may prevent the user from authenticating.

Hard Tokens

Hard token authentication makes use of small hardware devices, as a means of combining digital security with a physical object. While the use of one-time-passwords generated by these devices may easily be adapted to a variety of existing systems, hard tokens suffer from a few key limitations.

Expensive and labor intensive: Hard tokens are costly to enterprises and difficult to provision, manage, and replace.
Cumbersome and inconvenient: The use of one-time codes, and the need to carry an additional device, can be inconvenient for users.
Vulnerable: When using hard tokens, the user takes the code from the token and places it back into the login interface. This leaves users vulnerable to phishing and other endpoint attacks.

Soft Tokens

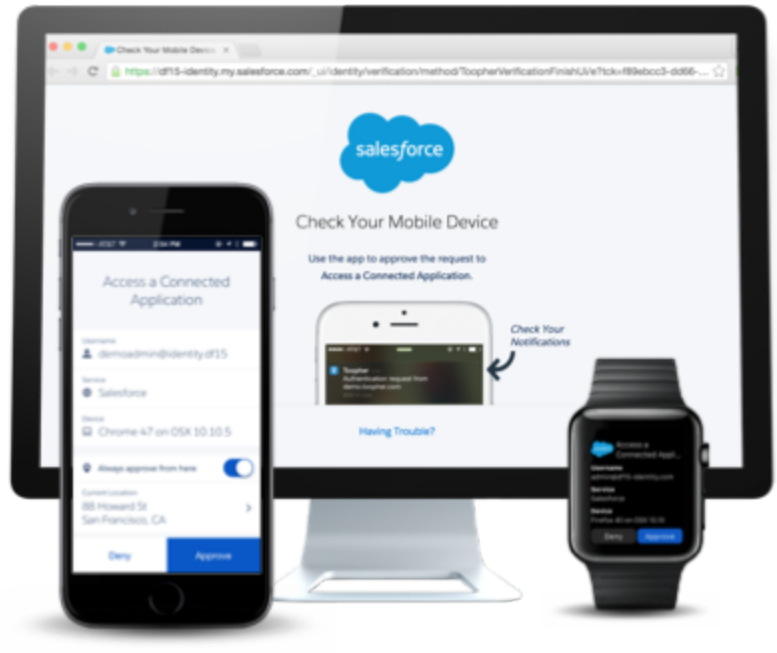
Soft token authentication arose as an alternative to hard token authentication, excising the need for a physical object for the two-factor authentication puzzle. Software tokens leverage existing devices such as the user's phone to generate one-time passwords. While this can significantly reduce the cost to the enterprise, and the annoyance to the user of carrying a second device, soft tokens still have critical gaps.

Cumbersome: The use of one-time codes can be cumbersome for users.
Vulnerable: When using soft tokens, the user takes the code from the token and places it back into the login interface. This leaves users vulnerable to phishing and other endpoint attacks.

While each of these factors adds significant security to the authentication process, clearly there is room for improvement.

Introducing Salesforce Authenticator

Salesforce Authenticator is a smart, simple, two-factor authentication solution that increases the security of your Salesforce deployment, while driving a better user experience for your end users.

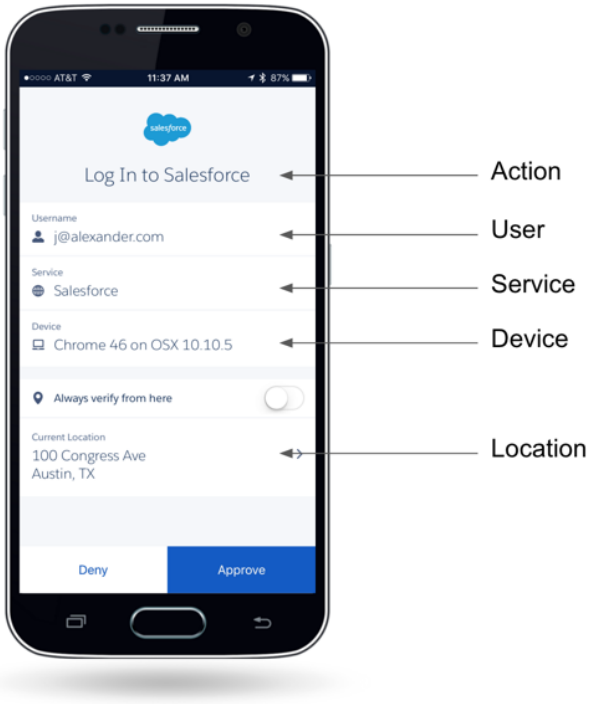


Like soft tokens, Salesforce Authenticator leverages a user's mobile device, making it easy and inexpensive to deploy and manage. However, Salesforce Authenticator advances beyond the limitations of traditional two-factor techniques by using un-phishable, out-of-band approvals. When an action needs to be authenticated, the user is sent a push notification. The user sees contextual information about the authentication and has the option to approve the request or deny it. And since this all happens in a secured, secondary channel, not only is usability improved, but so is the security.

A Superior User Experience for Two-Factor Authentication

When a user authenticates with Salesforce Authenticator, she is treated to an entirely new experience. Instead of the user relying on rote memorization to transcribe a series of numerical values, Salesforce Authenticator provides the user with all of the necessary details to make an informed decision. Specifically, Salesforce Authenticator tells the user:

- What **action** needs to be approved
- What **user** is requesting the action
- From which **service** is the requested action coming
- What **device** the user is using
- From what **location** would the user approve or deny this request



With this information, the user can simply tap the “Approve” or “Deny” button to execute her decision, and thus complete authentication quickly and get on with her intended task.

Contextual Automation and Transaction Security

Not only does Salesforce Authenticator provide even greater usability and trust with out-of-band authentication, but Salesforce Authenticator also delivers two key advancements in the authentication experience:

1. The ability to automate requests based on the mobile device's context, such as its location.
2. The ability to challenge the user for approval during critical transactions.

Users can tell Salesforce Authenticator to approve requests automatically when a user is in a trusted location; for example, if a user normally logs in from her corporate laptop at home, Salesforce Authenticator can learn this user’s trusted location/terminal combination and automatically respond to an authentication request. By using the mobile device’s location awareness, Salesforce Authenticator provides increased trust with far simpler user experience.

Important: Salesforce Authenticator will only automate responses if all of the following criteria are met: It’s the same user, performing the same action, on the same service, from the same device, from a trusted location. If even one of these isn’t true, then the user will be prompted for approval. For example, if an attacker is attempting to log in using stolen credentials, the user would be notified and prompted for approval, even if they were sitting at their desk (in a trusted location), since the attacker is using a different browser.

Automation can also help reduce the risk of teaching the user to reflexively approve actions...a risk known as repetition poisoning. Repetition poisoning is the phenomenon in which a user is prompted too often for a usually-innocuous action, such as a login request several times a day. While the login request appears completely normal, the frequency of notification desensitizes the user to the messaging, likely prompting at best a cursory glance, but more than likely a muscle-memory "approval." Contextual automation provides a less obtrusive user experience as well as a solution for repetition poisoning.

Salesforce Authenticator also enables a deeper level of transaction security. Instead of needing to respond to all authentication requests, users can leave the lower priority, more frequent authentication requests to Salesforce Authenticator's contextual automation. The ability to request approval for critical actions on the platform, such as access to reports, or SSO into critical applications, allows administrators to further protect their data and services. As a result, users are only prompted to respond to new or out of the ordinary authentication requests, such as different actions or perhaps the same action from a new device. With contextual automation, mundane authentication requests melt into the background, ensuring the users' attentions are called into focus when Salesforce Authenticator alerts them.

Using Salesforce Authenticator, you can increase security without sacrificing usability, enabling administrators to deploy critical security protections without strong objection from their users.

Getting Started

Want to learn more? Check out Securing Your Users' Identity on Trailhead!

Secure By Design

Salesforce Authenticator is secure by design, and it leverages a combination of open standards and best practices to deliver simple to use two-factor authentication for your Salesforce deployment.

Salesforce Authenticator is backed by a RESTful API with endpoints hosted at login.salesforce.com. All communications with the server are protected with TLS 1.2 using a 2048-bit key, providing confidentiality, integrity, and authenticity. The application provides additional message-level integrity and authenticity by including an 2048-bit OAuth RSA-SHA1 signature as specified by RFC 5849.

API Interactions

The mobile app engages in a series of interactions with the API. The following describes the exchange of messages.

Initial App Registration

When the app is downloaded and installed on an iOS or Android mobile device, the app generates a device specific 2048-bit RSA key. The app then contacts login.salesforce.com over TLS and registers with the API service. Only the public key is transported and stored on the API service.

Connection

When a user wishes to connect Salesforce Authenticator to their Salesforce account, the app contacts the server to generate a connection request. The client is presented with a unique two-word phrase that the user provides to the Salesforce UI. This phrase is generated from a unique dictionary of single-use bigrams. When the user provides the phrase to Salesforce, the API sends a notification to the device using the Apple or Google push services. Communication with these push systems is protected using TLS. The device is notified to contact the API (or can alternatively discover this through polling for work). The API provides the device with the details of the connection request. The user must approve this connection request, thus preventing unwanted connections. When the user approves, a message is sent to the service confirming the connection request as approved.

Authentication

When Salesforce needs to authenticate a particular action, the API sends a notification to the device using the Apple or Google push services. As with the initial connection, communication with these push systems is protected using TLS. The device is notified to contact the API (or can alternatively discover this through polling for work). The API provides the device with the details of the authentication request. The user must then approve or deny the request. When the user approves or denies, a message is sent to the service with the user's decision.

Location Automation

Salesforce Authenticator relies on the the location services available on the mobile device. Using a combination of GPS, Wi-Fi, and cellular tower triangulation, the mobile platform provides Salesforce Authenticator with the user's latitude/longitude coordinates.

Using this location, when the user approves a request, Salesforce Authenticator is able to identify and assign a “trusted location” to a previously approved action set. (An action set is defined as a unique user, action, service, and device from which the request emanated.) This location has radius of 100M around the trusted location. To minimize the impact to user’s battery, Salesforce Authenticator establishes a geo-fence on the device. The app waits passively for OS notifications that a user is entering or leaving a trusted location, at which point this status is provided to the API via an API update.

Important: In order to maintain the privacy of the user, location is never provided to the server. Only the status of in or out of a trusted location is maintained.

Detailed API Messages: API to Mobile App

Term	Description
pair	A request to connect a specific user to a specific instance of Salesforce Authenticator. Initiated by Salesforce, a notification to the mobile device is sent via push notification service. The app pulls the details of the pairing request from the API service and presents the details to the user for approval.
authenticate	A request to authenticate a specific action. Initiated by Salesforce, a notification to the mobile device is sent via push notification service. The app pulls the details of the authentication request from the API service and presents the details to the user for approval.
nudge	A request to update status of the app and its trusted locations. Initiated by the API via push notification to the mobile device if API has not heard from the app recently.

Detailed API Messages: Mobile App to API

Term	Description
register	A registration request performed when a new instance of Salesforce Authenticator is installed. The request informs the API of the existence of the new app instance and registers its public key.
poll	An app-initiated message that runs periodically whenever the app is open in the foreground. This checks for new work such as pairing requests or authentication requests.
accept/reject pairing	A message that sends the user's response to the API's request to pair.
accept/reject authentication	A message that sends the user's response to the API's request to authenticate an action.
change in trusted location	A message sent whenever the operating system detects a significant location change OR in response to an API request for an updated status.

Frequently Asked Questions

- How are messages and API communications protected?
    - All communications with the server are protected with TLS 1.2 using a 2048-bit key, providing confidentiality, integrity, and authenticity.
    - The application provides additional message-level integrity and authenticity by including an 2048-bit OAuth RSA-SHA1 signature as specified by RFC 5849.
- How are keys generated and protected?
    - 2048-bit RSA keys are generated when the app is first run and are unique to that installation. Only the public key is transported to the server.
    - Private keys are stored using the protected storage mechanisms available through the mobile operating system. This is a recent add to the Android platform and is used where available.
- What happens if a user's credentials are stolen, and the attacker attempts to use them when the user is in a trusted location? Will automation let the attacker in?
    - Absolutely not. In order for automation to occur, the same user must perform the same action on the same service, from the same browser, in a trusted location. In this case, since the attacker does not control the user's browser, when the attacker attempts to use the credentials, the user will receive a notification. The user can then block the request.
- What happens if a user is in a trusted location, but attempts to authenticate from a new browser/app?
    - If the trusted location does not match its previously approved browser/app, the new request from the new browser/app will be treated as an untrusted location. As such, Salesforce Authenticator will prompt the user to approve or deny the request.
- What happens is a user is in a trusted location, but the mobile device cannot identify the location and/or location services are unavailable?
    - In cases where location services are unavailable, Salesforce Authenticator will treat the location as an untrusted location. As such, Salesforce Authenticator will prompt the user to approve or deny the request.
- What happens if a user does not have data connectivity (no Wi-Fi, no data)?
    - In cases where users have no connectivity, users may use the verification code (time-based one-time password) associated with the connection. The TOTP syncs at the time of account connection with Salesforce Authenticator and can be used in situations where no signal is available. This solution also applies to scenarios when users are on a plane without Wi-Fi.