Doppelt hält besser

Multi-Faktor-Authentifizierung ist kein Hexenwerk

Noch immer ist häufig das Passwort das alleinige Mittel der Wahl zum Schutz digitaler Identitäten. Unser Autor betont, dass heute eine Multi-Faktor-Authentifizierung eigentlich als Stand der Technik gelten muss und auch längst keinen unüberwindlichen Aufwand oder Flexibilitätsverlust mehr bedeutet – und illustriert das am Beispiel der Einbindung einer Open-Source-Lösung.

Von Amir Alsbih, Weiterstadt

Zwei- (2FA) oder Multi-Faktor-Authentifizierung (MFA) trägt dazu bei, Risiken für digitale Identitäten zu minimieren, sie wirksam vor Verlusten zu schützen und gesetzliche Anforderungen zu erfüllen (Stichwort „Stand der Technik"). Dennoch sind MFA-Systeme in den meisten deutschen Unternehmen noch nicht angekommen. Diese Zögerlichkeit liegt unter anderem darin begründet, dass viele Verantwortliche die Multi-Faktor-Authentifizierung noch als benutzerunfreundliche und schwer zu integrierende Systeme in Erinnerung haben – obwohl diese Zeiten mittlerweile vorbei sind. Moderne MFA-Systeme bieten heute ein breites Spektrum von Einsatzmöglichkeiten.

Unterstützung unterschiedlicher Token

MFA-Lösungen lassen sich heute flexibel mit verschiedenen Token-Typen einsetzen (z. B. Software-, Hardware-, SMS-, Voice- oder mOTP-Token), die je nach Einsatzzweck und Risikostufe des Anwenders für unterschiedliche Zielgruppen konfiguriert werden können und verschiedene Vorteile je nach Nutzeranforderung bieten (vgl. Abb. 1). So ist es möglich, für hochsichere, seltene Authentifizierungen ein Hardware-Token zu verwenden. Müssen viele Benutzer – ohne den Einsatz von Apps oder Smartphones – authentifiziert werden, kommen SMS-Token zum Einsatz. In Massenszenarien mit vielen Authentifizierungen können Benutzer per Push-Token einfach auf dem Handy einen Button anklicken und sich damit einloggen, ohne den Token-Wert selbst eingeben zu müssen.

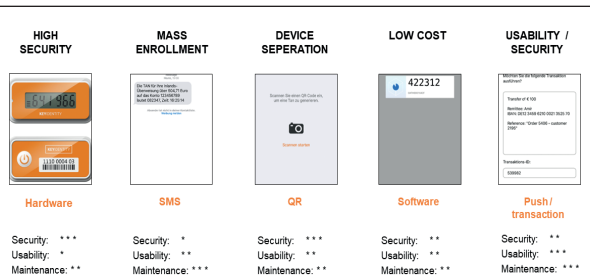

Abbildung 1: Überblick über fünf zentrale Token-Typen und ihre Eigenschaften

Selbst Szenarien mit Offline-Authentifizierung lassen sich mittlerweile per QR-Token in Kombination mit Public-Key-Mechanismen darstellen. Wichtig ist, dass dabei kein Geheimnis auf dem Rechner liegt. So funktionierten frühere Offline-Authentifizierungsverfahren noch nach dem Prinzip, dass die nächsten zu verwendenden Token-Werte wie „ABCDE", „23456" oder „A2b3c4" auf dem Rechner des autorisierten Benutzers gespeichert werden – bei Bedarf vergleicht der Computer dann den Wert, den der Anwender eingibt, mit diesem hinterlegten Wert. Wird der Rechner kompromittiert, kann allerdings auch ein Angreifer in den Besitz dieser Daten gelangen – und erbeutete Token für die Authentifizierung an anderen Systemen nutzen.

Unterstützung von Third-Party-Szenarien

Viele Unternehmen arbeiten eng mit externen Partnern oder Lieferanten zusammen, die in Zeiten der Digitalisierung uneingeschränkten Zugriff auf die für sie notwendigen Daten haben müssen. Dies betrifft oft auch den Zugang zu sensiblen Kunden- und Mitarbeiterdaten, der strengen Datenschutzbestimmungen unterliegt. Im Rahmen einer Multi-Faktor-Authentifizierung lässt sich die Gültigkeit eines Tokens im Hinblick auf die Anzahl der Verwendungen oder die Zeitdauer für die Nutzung sehr detailliert definieren. Damit kann man unter anderem wirksam verhindern, dass beispielsweise ein externer Partner nach dem Auslaufen der Zusammenarbeit weiterhin Zugriff auf die Daten eines Unternehmens hat.

Unterstützung von Token mit Transaktionsbindung

Moderne Token für die 2FA/MFA ermöglichen die Absicherung von Prozessen durch die Berechnung eines One-Time-Passwords (OTP) auf Basis von Transaktionsdetails – zum Beispiel Konto- und Bankdaten. Dies fordert unter anderem die EU-Zahlungsdienste-Richtlinie PSD2. So kann ein geeignetes System neben Vertraulichkeit und Integrität auch Nichtabstreitbarkeit für Transaktionen gewährleisten.

Anbindung über Standardschnittstellen

Moderne MFA- oder 2FA-Systeme lassen sich vergleichsweise einfach in bestehende IT-Umgebungen von Unternehmen und Behörden integrieren. Durch Standardanbindungen an ein Active Directory, LDAP-Netzwerkprotokolle, SQL-Datenbanken oder JSON-Datenformate können Anwendungen schnell um eine entsprechende Komponente erweitert und die darin verwalteten Benutzer abgesichert werden.

Zudem lassen sich über bereitgestellte Schnittstellen (APIs) auch Eigenentwicklungen, Custom-Software oder Portale zügig und mit wenigen Zeilen Code absichern. Die Integrationszeiten und der damit verbundene Aufwand für die IT-Abteilung sinken dadurch noch einmal zusätzlich. So ist es mittlerweile möglich, den Großteil von Applikationen innerhalb eines Tages um einen zweiten Authentifizierungsfaktor zu ergänzen, um den Schutz digitaler Identitäten deutlich zu verbessern.

Integrations-Beispiele

Das in Abbildung 2 dargestellte Beispiel zur Integration der Open-Source-basierten Multi-Faktor-Authentifizierungslösung LinOTP (www.linotp.org) verdeutlicht, dass der Aufwand für die Anbindung via LDAP oder RADIUS im Idealfall nur wenige Minuten beträgt – die Konfiguration zur Anbindung an ein Active Directory (AD) erfolgt in einfachen Schritten über den LDAP-Resolver.

MFA-Integration in Web-Applikationen

Gerade Web-Applikationen benötigen einen guten Schutz für digitale Identitäten – dem Open Web Application Security Project (OWASP) zufolge liegt hier die Authentifizierung auf Platz 2 der Sicherheitslücken. Im Falle der bereits als Beispiel dienenden Open-Source-Lösung LinOTP genügt es, eine Applikation um wenige Codezeilen zu ergänzen (s. a. www.linotp.org/doc/latest/part-module-dev/authentication/validate.html):

```
auth = requests.post(‚https://LINOTP/validate/check’,
    data={‚user’: ‚MyUserName’, ‚pass’:
    ‚MyOneTimePasswordValue’})

auth_response = json.loads(auth.text)
if (auth_response[‚result’][‚status’] == True and
    auth_response[‚result’][‚value’] == True):
    print(‚Token is valid’)
else:
    print(‚Token is invalid’)
```

Dabei werden der Benutzername sowie der aktuelle Token-Wert, den der Benutzer in ein Formularfeld eingegeben hat, an die MFA-Instanz übermittelt. Diese vergleicht den Token-Wert mit dem nächsten erwarteten Token-Wert und sendet als Antwort ein „True“ oder „False“ zurück. Damit die Applikation unterstützt wird, genügt nach diesem Prinzip eine einzige http-Abfrage und das Auswerten der Antwort.

Integration mittels Push-Token

Auch Push-Token, bei denen der Benutzer zur Bestätigung eines Logins oder einer Transaktion mittels Smartphone nur auf „ja“ oder „nein“ klicken muss, lassen sich über wenige einfache Schritte integrieren. Die Push-Nachricht wird entsprechend der LinOTP-Konfiguration über den Nutzernamen und die Token-PIN ausgelöst. Als Token-PIN kann auch das Passwort des Benutzers verwen-

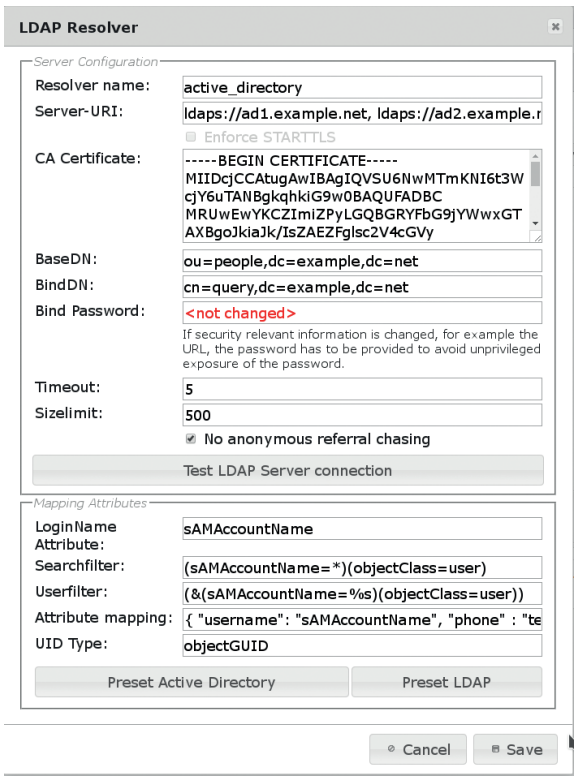

Abbildung 2: Anbindung der Open-Source-Lösung LinOTP über den LDAP-Resolver

det werden, wenn die Regel „otppin=1“ definiert wurde (vgl. www.linotp.org/doc/latest/part-management/policy/authentication.html#otp-pin-variants). Die generische API-Abfrage für den Vorgang lautet:

```
https://LINOTP/validate/check?user=USERNAME[@REALM]
    &pass=TOKENPIN
    &data=TRANSACTIONDATA
    &content_type=0
```

Der LinOTP-Server antwortet bei einer erfolgreichen Challenge mit den folgenden Transaktionsdetails inklusive der Seriennummer des Tokens und der Transaktions-ID, beispielsweise:

```
{
  "detail": {
    "linotp_tokenserial": "KIPT00031EDC",
    "transactionid": "353846450659",
    "message": "",
    "linotp_tokentype": "push"
  },
  "version": "LinOTP 2.9.1.3",
  "jsonrpc": "2.0802",
  "result": {
    "status": true,
    "value": false
  },
  "id": 0
}
```

Die Transaktion sollte nun in der Authenticator-App der empfangenden Plattform angezeigt und vom Benutzer freigegeben werden. Das Smartphone kommuniziert das Ergebnis der Nutzerentscheidung zurück an den LinOTP-Server. Die generische API-Abfrage für die Überprüfung der Ergebnisse lautet:

```
https://LINOTP/validate/check_status?user=USER
        &pass=OTPPIN&transactionid=TRANSACTIONID
```

Im Gegenzug wird der Status der Transaktion empfangen – das Ergebnis „valid_tan“: true zeigt dabei an, dass die Transaktion vom Benutzer freigegeben wurde.:

```
{   "detail": {
    "transactions"353846450659": {
      "": {
        "status": "open",
        "token": {
          "serial": "KIPT00031EDC",
          "type": "push"
        },
        "received_tan": true,
        "message": "",
        "received_count": 1,
        "valid_tan": true
      }
    }
  },
  "version": "LinOTP 2.9.1.3",
  "jsonrpc": "2.0802",
  "result": {
    "status": true,
    "value": true
  },
  "id": 0
}
```

Fazit

Die Beispiele zeigen, dass die Einbindung einer Multi-Faktor-Authentifizierung in bestehende Standard-Web-Applikationen oder IT-Umgebungen heute vergleichsweise schnell und einfach umzusetzen sein kann. Nach ähnlichen Prinzipien lassen sich unter anderem Anwendungen ergänzen, die keinem Standard für die Authentifizierung folgen – zum Beispiel selbst entwickelte Web-Anwendungen, die das XML-Framework SAML zum Austausch von Authentifizierungs- und Autorisierungsinformationen nicht unterstützen. Darüber hinaus lassen sich Workflows automatisieren, sodass etwa Token oder Benutzer automatisch ausgerollt werden können.

Funktionen einer MFA-Plattform lassen sich zudem in zentrale proprietäre Anwendungen einbinden – beispielsweise Self-Services für Benutzer in ein Unternehmens-Portal. Für die Integration in Standardprodukte wie VPNs oder Serversysteme kommen unterstützte Standards wie LDAP, RADIUS oder SAML zum Tragen – dadurch genügt es, einen Server beidseitig einzutragen, um das System „MFA-ready“ zu machen.

Um all dies zu gewährleisten, ist eine modulare Architektur notwendig, die verschiedene Authentifizierungsprotokolle (z. B. für Microsoft Windows, macOS, Linux), Token-Typen (z. B. Push-, QR- und Hardware-Token) und Benutzerverzeichnisse (z. B. Microsoft AD, SQL-Datenbanken oder OpenLDAP) unterstützt. Eine Open-Source-Konzeption macht dabei alle APIs und den Quellcode vollständig einsehbar und ermöglicht es der gesamten Community, den Code entsprechend ihrer jeweiligen Anforderungen anzupassen – und im Extremfall sogar Sicherheits-Patches selbst zu entwickeln und einzupflegen, sollte dies einmal notwendig sein. ■

Dr. Amir Alsbih ist CEO von KeyIdentity.