inkl. CD!

Ausgabe 3.11 April | Mai www.phpmagazin.de S&S

Deutschland 9,80 € Österreich 10,80 € | Schweiz 19,20 sFr Niederlande 11,25 € | Luxemburg 11,25 €

PHPmagazin

Frontend-Architektur
Architekturmuster für Graphical User Interfaces

YUI Library
Webanwendungen mit DataGrids modernisieren

Urheberrecht
Alles, was Sie darüber wissen müssen

Sicherheit
Die Sache mit den Headern

Ausgabefilter im praktischen Einsatz

Video der WebTech IPC 2010: Wie das Pflegen von Webseiten mit Filtern erleichtert werden kann.

YUI 2
Die Yahoo! User Interface Library mit mehr als 30 Komponenten inklusive High-Level UI Widgets.

Joomla! 1.6
Die neue Version des beliebten und einfachen Content-Management-Systems.

SELFPHP
Geballtes PHP-Wissen einschließlich einer Befehlsreferenz, Tutorial, Code-Snippets und einem PHP-Kochbuch.

Arbeit und Ressourcen sparen mit RedSpark und FLOW3

The Power of Frameworks

Joomla! Template-Entwicklung mit Version 1.6

Yubikey One-Time-Passwörter gegen Keylogger und Mithörer

Datenträger enthält Info- und Lehrprogramme gemäß §14 JuSchG

Außerdem mit auf der CD

RedSpark 1.2 , Zend Framework 1.11.3, FLOW3 1.0.0-alpha14 und alle Beispielprogramme zu den Artikeln.

4 195649 609809 03

Yubikey: Mit One-Time-Passwörtern gegen Keylogger und Mithörer

Mithören ist zwecklos!

Normale Passwörter sind out, One-Time-Passwörter sind in! Coole kleine Hardwaregeräte, die Log-ins unknackbar machen, will jeder haben. Banken verteilen sie fürs Online-Banking, warum nicht auch diese Sicherheit auf der eigenen Webseite nutzen? Hier zeige ich, wie das geht.

von Michael Kliewe

Kaum jemand überblickt heutzutage noch seine Passwörter. Sie liegen auf diversen Festplatten, auf USB-Sticks und Notebooks, Backup-Festplatten und CDs, iPhones, iPads, auf Servern im Internet und auf Zetteln. Sie werden durch verschlüsselte und unverschlüsselte WLANs geschickt, über oftmals nicht gesicherte HTTP-, FTP-, SMTP- und IMAP-Verbindungen übertragen, und niemand kann wirklich sicher sein, dass er der einzige Besitzer seiner (statischen) Passworte ist. In Zeiten von millionenfach installierten Bots, Keyloggern und gut gemachten Phishing-E-Mails und -Webseiten muss man davon ausgehen, dass die Logins abgehört und an interessierte Dritte weitergegeben werden, gerade wenn es um Onlinebanking, wichtige Administrationsportale oder E-Mail-Postfächer geht. Selbst wenn die Daten nicht abgehört werden, können sie häufig durch Brute-Force-Attacken angegriffen werden, da sie entweder zu einfach oder zu kurz gewählt werden.

Banken und Firmen

Eine interessante Lösung dazu hat uns das Online-Banking vorgemacht: TAN und iTAN [1]. Das sind Codes auf einem Zettel Papier, die man zusätzlich bei wichtigen Transaktionen eingeben muss. Jede TAN ist nur einmal nutzbar, es nützt Angreifern also nichts, sie abzuhören bzw. mitzuloggen. Der Nachfolger ist der TAN-Generator, ein kleines Stück Hardware mit einem kleinen Display, auf dem nach einem Knopfdruck eine TAN generiert wird, alle 60 Sekunden eine neue. Diese ist dann für einige Minuten gültig.

Große Firmen nutzen immer häufiger ähnliche Hardware-Token, beispielsweise den RSA SecurID Token [2]. Diese werden dann zur Einwahl in das VPN oder zur Freischaltung von Firewallports genutzt. Doch der Preis ist für Privatanwender zu hoch, häufig liegen die Kosten für Lizenzen und Hardware (Token + Appliance) im hohen dreistelligen Bereich je User pro Jahr.

Ein weiteres schönes Beispiel ist der World of Warcraft Authenticator. Dies ist ebenfalls ein One-Time-Password-Generator, der für wenige Euro bezogen werden kann und dann den WoW-Account schützt, da bei jedem Login (sowohl Webseite als auch Spiel) ein 6-stelliger Code eingegeben werden muss. Damit sind Account-Hacks quasi ausgeschlossen. Doch der Token funktioniert nur für Spiele von Blizzard.

Es gibt aber eine günstige und vielfältig einsetzbare Alternative: den *Yubikey* von Yubico [3]. Das ist ein kleiner universeller USB-Token, der One-Time-Passworte (OTP) generiert und nicht auf einen Einsatzzweck beschränkt ist, sondern in mehreren Applikationen eingesetzt werden kann, solange sie es unterstützen. Das Besondere am Yubikey ist, dass alles Open Source ist, wodurch es eine Menge Clients und auch Serverimplementierungen gibt, im Gegensatz zu den Alternativen, bei denen teure Applicances und Serverinfrastruktur aufgebaut werden müssen. Der Yubikey kann sowohl für eine One-Factor- als auch für eine Two-Factor-Authentifizierung genutzt werden. One-Factor meint dabei, dass allein der Code des Tokens für den Login reicht. Es ist also kein separater Username oder Passwort nötig, denn der Username steckt bereits im Code. Two-Factor ist die sichere Variante, weil dabei sowohl ein Passwort („etwas, das man weiß") als auch ein Token („etwas, das man hat") für die Authentifizierung nötig sind. Wenn also der Token auf der Straße gestohlen wird, nützt das dem Dieb nichts, denn er kennt nicht das zusätzlich benötigte Passwort.

Vorgehensweise

Eine weitere Besonderheit ist die Art und Weise, wie das Passwort in das Passwortfeld gelangt. Normalerweise muss man einen 4- oder 6-stelligen Code abtippen. Der Yubikey emuliert jedoch eine USB-Tastatur und „schreibt" das Passwort so in das Passwortfeld. Dadurch ist es möglich, 44 Zeichen zu verwenden, was ziemlich sicher ist, und man muss sie nicht abtippen, sondern kann einfach den Token in einen USB-Slot stecken und auf den

Abb. 1: Der Yubikey ist klein und praktisch

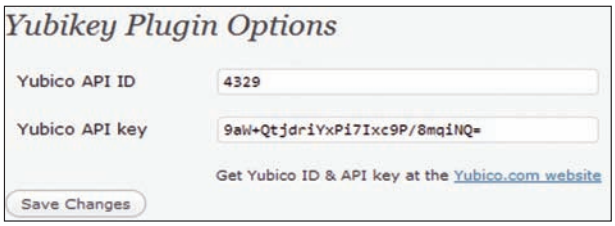

Abb. 2: WordPress-Plug-in „yubikey-plugin“: Optionen

Knopf drücken. Es ist keine zusätzliche Software nötig, der Yubikey funktioniert unter Windows, Mac und Linux gleichermaßen. Die einzige Situation, in der diese Vorgehensweise nachteilig ist, sind Internetcafes oder Kiosksysteme, an denen man keinen Zugriff auf den USB-Port hat.

Abb. 3: Yubikey-Aktivierung eines WordPress-Benutzer-Accounts

Listing 1

```
<?php
$login = false;
if (dbAuthenticate($_POST['username'], $_POST['password'])) {
  $login = true;
}

function dbAuthenticate($username, $password) {
  // Datenbankabfrage

  return true;
}
```

Konfigurationen

Der Yubikey kann umprogrammiert werden und zwei unabhängige Konfigurationen verwenden. Folgende drei Alternativen stehen zur Verfügung:

1. Standard YubiKey: 44-Zeichen-One-Time-Passwort
2. OATH (RFC4226): 6- oder 8-Zeichen-One-Time-Passwort
3. Statisches Passwort mit 1-44 Zeichen

OTP

Ein *One-Time-Passwort* (deutsch: Einmalpasswort) ist einmalig und für eine begrenzte Zeit gültig. Dazu wird es bei Bedarf beim Benutzer generiert (unter Zuhilfenahme eines Verschlüsselungsalgorithmus) und kann dann beim Dienstanbieter verifiziert werden. Da es nur einmalig gültig ist, sind so genannte Replay-Attacken nicht möglich, auch einfaches Mithören ist nutzlos, da es kein zweites Mal benutzt werden kann. Ein Angreifer müsste schon die Verbindung abfangen *(Man-in-the-Middle)*, was jedoch viel schwerer ist als nur mitzuhören. Weitere Informationen zu OTP bei Wikipedia [4].

Alternativ zum immer wechselnden Passwort beherrscht der Token auch eine zweite Methode: das statische Passwort. Dieses kann zusätzlich zur OTP-Fähigkeit konfiguriert werden, sodass man beide Möglichkeiten in einem Token hat. Für Programme und Webseiten, die nur normale Passworte unterstützen, verwendet man ein langes statisches Passwort, für Yubikey-fähige Authentifizierungsmöglichkeiten dann die One-Time-Passworte. Um beispielsweise meine TrueCrypt-Bootpartition zu entschlüsseln, muss ich ein langes Passwort wählen und es jedes Mal beim Hochfahren eingeben. Anstatt mir das lange Passwort zu merken und einzutippen, kann ich es auch im Yubikey speichern und automatisch eingeben lassen. Ein so langes Passwort kann sich niemand merken und auch ein Abgucken beim Eintippen ist nicht möglich.

Auf Benutzerseite werden also Yubikey Token benötigt, die je nach Menge zwischen 15 $ und 25 $ kosten. Damit kann der Benutzer den Login bei allen Diensten, die den Yubikey unterstützen, um diesen zusätzlichen Faktor erweitern und die Authentifizierung sicherer machen. Bekannte Beispiele sind Google Apps, Microsofts LiveID, TrueCrypt, Roundcobe Webmailer, Drupal, Joomla, WordPress, KeePass, Password Safe, osCommerce, MediaWiki, Radius, Windows Log-in, Mac OS X Log-in und viele mehr [5],[6],[7]. Aber auch eigene Applikationen können sicherer gemacht werden, mehr dazu weiter unten.

Um meinen Blog, der auf WordPress basiert, abzusichern, habe ich das WordPress-Plug-in *yubikey-plugin* installiert und konfiguriert, und schon nach fünf Minuten hatte ich meinen Account zusätzlich abgesichert. Hierbei ist es sogar möglich, dieses Plug-in userspezifisch zu aktivieren, sodass Benutzer-Accounts ohne konfigurierten Yubikey sich weiterhin nur mit ihrem Passwort einloggen können.

Um nun auch die Log-in-Formulare auf den eigenen Webseiten abzusichern, muss man etwas tiefer in die Trickkiste greifen und mithilfe von PHP die Authentifizierung erweitern. Nehmen wir folgendes einfache Log-in-Formular:

```
<form action="authenticate.php" method="POST">
  <input type="text" name="username"/>
  <input type="password" name="password"/>
  <input type="submit" value="Login"/>
</form>
```

Das PHP-Skript auf dem Server (*authenticate.php*) sieht sehr vereinfacht so aus wie in Listing 1 (das ist keinesfalls sicher und sollte so nicht verwendet werden).

Um diesen Log-in-Prozess nun um den Yubikey zu erweitern, sind nur wenige Schritte nötig:

1. Dem Benutzer wird ein weiteres Eingabefeld für das One-Time-Passwort präsentiert, falls der Useraccount dafür konfiguriert wurde. Das kann entweder

ein weiteres Formular auf der nächsten Seite sein oder ein zusätzliches Textfeld, das per JavaScript eingefügt wird. Falls der Yubikey Pflicht für alle Benutzer ist, kann es natürlich permanent eingeblendet werden.

2. Der PHP-Code auf dem Server muss um die zusätzliche Prüfung erweitert werden. Dazu nutzen wir die bereits exisierende PHP-Klasse *yubikey-php-webservice-class* von Tom Corwine [8] und den kostenlosen Yubico Authentication Web Service. Wir könnten auch einen eigenen lokalen Authentifizierungsserver installieren, sodass die OTP nicht durch das Internet übertragen werden kann und man nicht abhängig ist von einem externen Web Service.

Vor allem der zweite Punkt ist interessant, er soll hier auch näher erklärt werden. Das PHP-Skript erhält also ein 44-stelliges Passwort und muss es nun validieren. Dazu gehört nicht nur die Prüfung, ob es eine gültige Zeichenfolge ist, sondern auch, ob es schon einmal benutzt wurde. Das alles macht der Authentifizierungsserver für uns, wir müssen die Zeichenfolge nur sicher dorthin übertragen und das Ergebnis auslesen. Damit der Service genutzt werden kann, muss ein API-Key generiert werden. Das kann kostenlos auf der entsprechenden Webseite bei Yubico erledigt werden: https://upgrade.yubico.com/getapikey/

Falls beim Zugriff auf den Authentifizierungsserver kein HTTPS zum Einsatz kommt, kann man mithilfe dieses API-Keys die Antwort verifizieren, denn der Server signiert die Antwort optional. Mit dieser ClientID und dem API-Key können wir nun den Web Service nutzen. In der einfachsten Form ist es nichts anderes als ein spezieller GET-Request an die Server von Yubico:

```
http://api.yubico.com/wsapi/2.0/verify?id=1851&otp=ccccccccbchdifctrndncc
                                hkftchjlnbhvhtugdljibej&nonce=IrgendeinZufallstext
```

Die Antwort sieht dann so aus:

```
h=vjhFxZrNHB5CjI6vhuSeF2n46a8=
t=2010-09-23T20:34:51Z0678
otp=ccccccccbchdifctrndncchkftchjlnbhvhtugdljibej
nonce=IrgendeinZufallstext
sl=75
status=OK
```

Detaillierte Informationen zum API befinden sich im Kasten „Yubico Web API“. Dieses Verfahren kann man auch auf dieser Demotestseite ausprobieren und die Rückgaben überprüfen: http://yubico.com/demo/php-yubico.php

Da der Authentifizierungsservice von elementarer Bedeutung ist, sollte er natürlich permanent verfügbar sein. Bei Benutzung des Yubico-Dienstes stehen gleich fünf Server zur Verfügung, sodass bei einem Ausfall noch Alternativen zur Verfügung stehen.

Diese Kommunikation mit dem Server wickelt die bereits angesprochene PHP-Klasse *yubikey-php-webservice-class* ab. Sie generiert eine Signatur, sendet den Request mithilfe von *curl* an das Yubico-API, überprüft die Signatur des Ergebnisses, den Zeitstempel und den

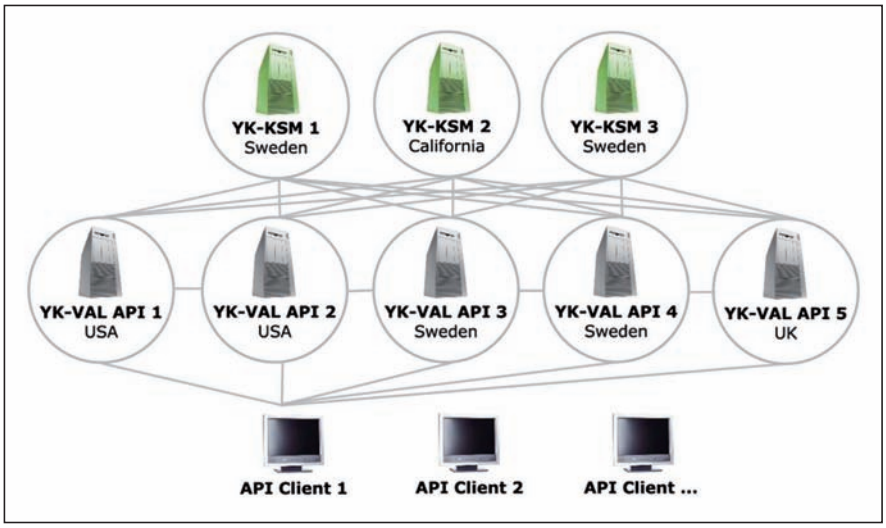

Abb. 4: Mehrere API-Server stehen zur Verfügung

Status. Das erweiterte Login-Skript sieht dann aus wie in Listing 2.

Mithilfe der Yubikey-Klasse ist die Prüfung also schnell integriert. Wenn man eine One-Factor-Authentifizierung implementieren möchte, kann man das bereits vorhandene Passwortfeld nutzen. Falls das Passwort 44 Zeichen lang ist, handelt es sich um ein Yubikey-OTP und man muss die entsprechende Yubikey-Authentifizierung durchführen.

Fazit

Der Yubikey ist ein wirklich tolles OTP-Token, das kein lästiges Abtippen verlangt, ohne Batterien auskommt und darüber hinaus sehr klein und sehr vielfältig einsetzbar ist. Der Sicherheitsexperte Steve Gibson hat bereits im September 2008 in seinem Podcast „Security Now“ Episode 143 [10] vom Yubikey berichtet, und er war sehr begeistert. Auf den Webseiten des Herstellers finden sich auch etliche Paper und Untersuchungen zur Sicherheit und Anwendbarkeit. Und ja, ich bin ein großer Fan dieses Produkts.

Yubico Web API

Anfrage: http://api.yubico.com/wsapi/2.0/verify

Parameter	Zweck
Id	ClientID, wird benötigt, um die Antwort zu signieren
otp	Das 44-stellige One-Time-Passwort
nonce	16-40 Zufallszeichen

Antwort:

Parameter	Zweck
h	Signatur
t	Zeitstempel
otp	Das 44-stellige One-Time-Passwort
nonce	Die Zufallszeichen
sl	Wie viele externe Validierungsserver haben erfolgreich authentifiziert
status	Statuscode: OK, BAD_OTP, REPLAYED_OTP, BAD_SIGNATURE, REPLAYED_REQUEST und weitere

Eine ausführliche Protokollbeschreibung befindet sich unter [9].

Michael Kliewe hat Informatik an der Universität Paderborn studiert und arbeitet nun als Programmierer bei mail.de in Gütersloh. In seiner Freizeit betreibt er einen der größten deutschen PHP-Blogs unter http://www.phpgangsta.de.

Links & Literatur

[1] http://de.wikipedia.org/wiki/Transaktionsnummer
[2] http://de.wikipedia.org/wiki/SecurID
[3] http://www.yubico.com/products/yubikey/
[4] http://de.wikipedia.org/wiki/Einmalpasswort
[5] http://wiki.yubico.com
[6] http://vimeo.com/yubikey/videos/sort:date
[7] http://www.youtube.com/results?search_query=yubikey
[8] http://code.google.com/p/yubikey-php-webservice-class/
[9] http://code.google.com/p/yubikey-val-server-php/wiki/ValidationProtocolV20

Listing 2

```
<?php
$login = false;

if (dbAuthenticate($_POST['username'], $_POST['password']) &&
    yubikeyAuthenticate($_POST['username'], $_POST['otp'])) {
    $login = true;
}

function dbAuthenticate($username, $password) {
    // Datenbankabfrage zur Prüfung von Username und Passwort

    return true;
}

function yubikeyAuthenticate($username, $otp) {
    // Datenbankabfrage ob die ersten 12 Zeichen von $otp zum Usernamen gehören

    // OTP überprüfen
    $token = new Yubikey($apiID, $signatureKey);
    $token->setCurlTimeout(20);
    $token->setTimestampTolerance(500);
    if ($token->verify($otp)) {
        return true;
    }
    return false;
}
```