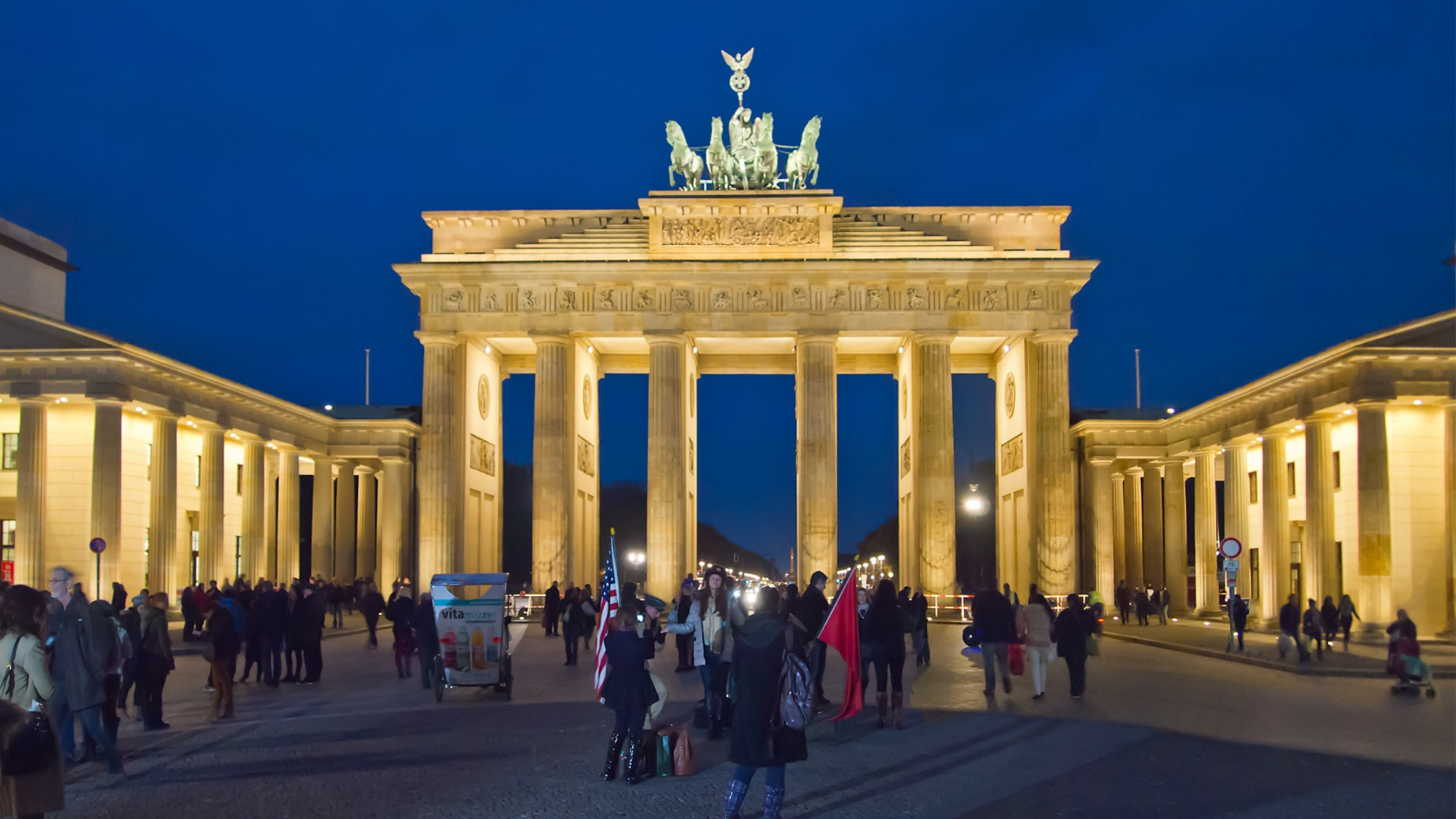
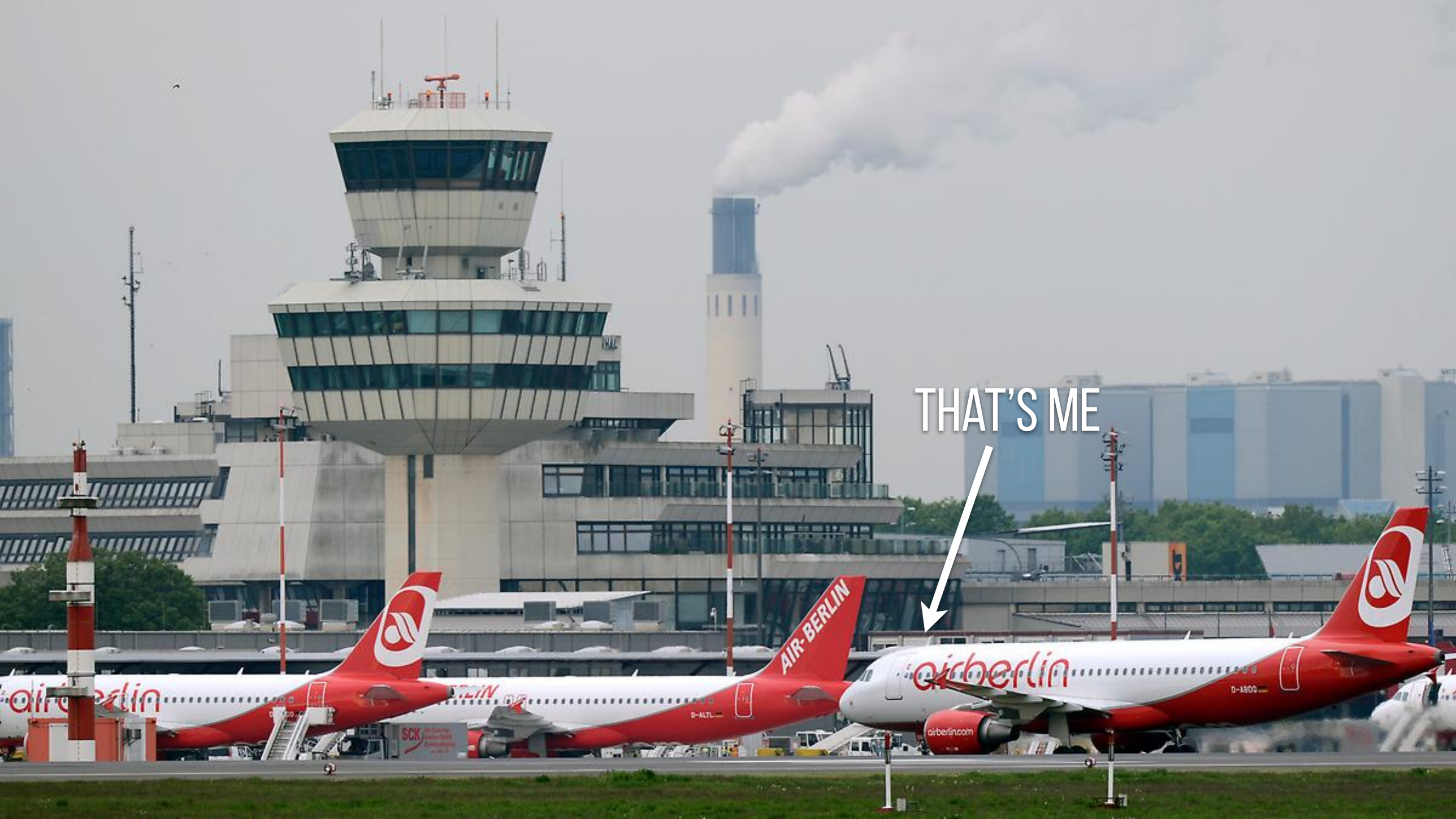
NODE.JS AUTHENTICATION AND DATA SECURITY

Tim Messerschmidt
Head of Developer Relations, International
Braintree


vitamizer


THAT'S ME
AIR-BERLIN
airberlin
airberlin.com

```
threat of replay attacks and more. Express offers an optional
middleware called `csurf`
footnote:[https://github.com/expressjs/csurf] that can be
installed via npm. Effectively the module provides a unique
token that needs to be rendered in forms and will be validated
after form-submission - a mechanism that's similar to what we
have done with providing the `state` parameter when requesting
the Authorization Code in the previous sample.

.Using the csurf to prevent cross-site request forgery
[source,js]
----
var csurf = require('csurf');

var csrfMiddleware = csurf({
  cookie: true
});

app.get('/form', csrfMiddleware, function(req, res) {
  res.render('form', { csrfToken: req.csrfToken() });
});
----

In this example we create a new middleware based on csurf that
stores uses cookies instead of `req.session` in order to store
the CSRF token secret. This middleware is mounted for the route
`/form` and provides the generated token as option to the
`form` template.

[source,jade]
----
extends layout

block content
  h1 CSRF protection using csurf
  form(action="/login" method="POST")
    input(type="text", name="username=", value="Username")
    input(type="password", name="password", value="Password")
    input(type="hidden", name="_csrf", value="#{csrfToken}")
    button(type="submit") Submit
----

By rendering the CSRF token as hidden part of a hypothetical
```


1. Securing the Login with OAuth 2 and OpenID Connect

In this chapter we are going to discuss the concepts behind the two standards *OAuth 2.0* and *OpenID Connect* in order to provide a comprehensive overview of current authentication and authorization standards. In order to do so, the difference between authentication and authorization will be outlined, followed by an explanation of OAuth's evolution throughout the years. Afterwards, we will sketch out a basic implementation of an OAuth 2.0 server and client that leverages OpenID Connect functionality.

1.1. The difference between authentication and authorization

A common issue is seeing both authentication and authorization as one and the same. In fact, they are very different and can be used in different scenarios or combined in order to allow for accessing different kinds of information. In order to understand why multiple standards exist and are being pushed forward at the same time, a basic understanding of the main differences will be provided.

1.1.1. Authentication

Authentication is the process of identifying a user against a service. OpenID used to be the first standard that aimed at providing a decentralized protocol for identifying users across multiple sites. The idea behind this was very simple: avoiding the tedious task of re-entering information over and over. Basically the log-in process is being delegated to another site.

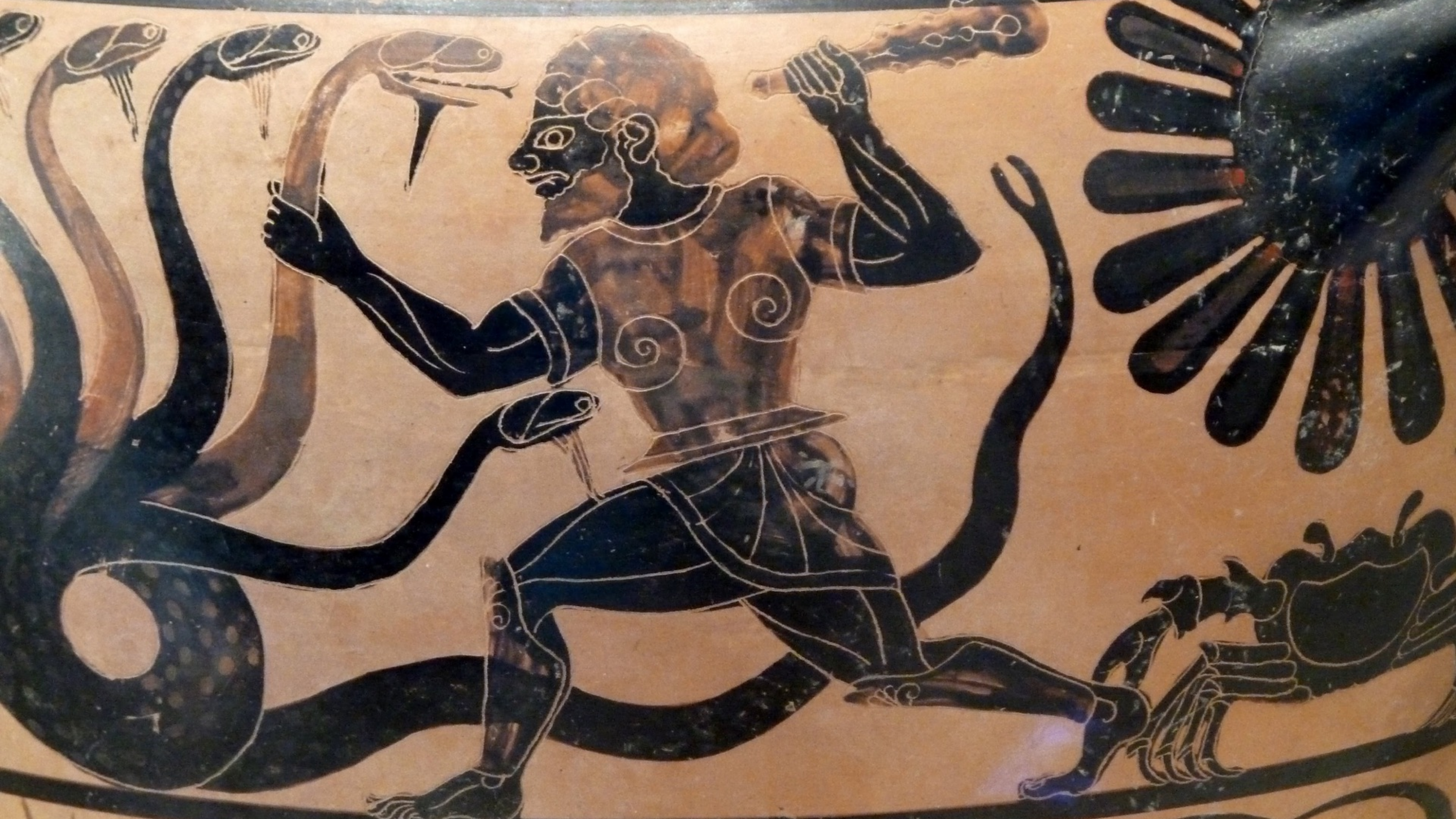
OpenID got introduced in 2005 and saw enormous growth with totaling over 1 billion user accounts in 2009[1]. Recent development showed less demand for OpenID and central identity platforms. Instead hybrid approaches were being introduced that offered both user authentication and authorization at the same time.

+ Braintree

since 2013

CONTENT

1. **Introduction_**
2. Well-known security threats
3. Data Encryption
4. Hardening Express
5. Authentication middleware
6. Great resources



THE HUMAN ELEMENT

TOP 10 PASSWORDS 2014

1. 12345
2. password
3. 12345
4. 12345678
5. qwerty
6. 123456789
7. 1234
8. baseball
9. dragon
10. football

BIT.LY/1XTWYIA

HONORARY MENTION

superman

batman

AUTHENTICATION & AUTHORIZATION

CONTENT

1. Introduction
2. **Well-known security threats_**
3. Data Encryption
4. Hardening Express
5. Authentication middleware
6. Great resources

OWASP TOP 10

BIT.LY/1A3YTVG

1. INJECTION

2. BROKEN AUTHENTICATION

3. CROSS-SITE SCRIPTING

XSS

4. DIRECT OBJECT REFERENCES

5. APPLICATION MISCONFIGURED

6. SENSITIVE DATA EXPOSED

7. ACCESS LEVEL CONTROL

8. CROSS-SITE REQUEST FORGERY

CSRF / XSRF

9. VULNERABLE CODE

10. REDIRECTS / FORWARDS

CONTENT

1. Introduction
2. Well-known security threats
3. **Data Encryption_**
4. Hardening Express
5. Authentication middleware
6. Great resources

HASHING

MD5, SHA-1, SHA-2, SHA-3

Once seen as bulletproof, 11 million+ Ashley Madison passwords already cracked

Programming errors make 15.26 million accounts orders of magnitude faster to crack.

by **Dan Goodin** - Sep 10, 2015 12:00pm CEST

LATEST FROM ARS TECHNICA/UK

Hunted, Channel 4's real-life fugitive thriller, begins tonight at 9pm

Austria plans 10 new spy agencies with vast surveillance powers

LIFE LINE

AMD forms new "more agile" graphics division to recapture GPU market share

Design firm proposes

ishouldnotbedoingthis

ARSTECHNICA.COM/SECURITY/2015/09/ASHLEY-MADISON-PASSWORDS-LIKE-THISISWRONG-TAP-CHEATERS-GUILT-AND-DENIAL

ishouldnotbedoingthis
whyareyoudoingthis

ARSTECHNICA.COM/SECURITY/2015/09/ASHLEY-MADISON-PASSWORDS-LIKE-THISISWRONG-TAP-CHEATERS-GUILT-AND-DENIAL

ishouldnotbedoingthis
whyareyoudoingthis
justtryingthisout

ARSTECHNICA.COM/SECURITY/2015/09/ASHLEY-MADISON-PASSWORDS-LIKE-THISISWRONG-TAP-CHEATERS-GUILT-AND-DENIAL

ishouldnotbedoingthis
whyareyoudoingthis
justtryingthisout
thebestpasswordever

ARSTECHNICA.COM/SECURITY/2015/09/ASHLEY-MADISON-PASSWORDS-LIKE-THISISWRONG-TAP-CHEATERS-GUILT-AND-DENIAL

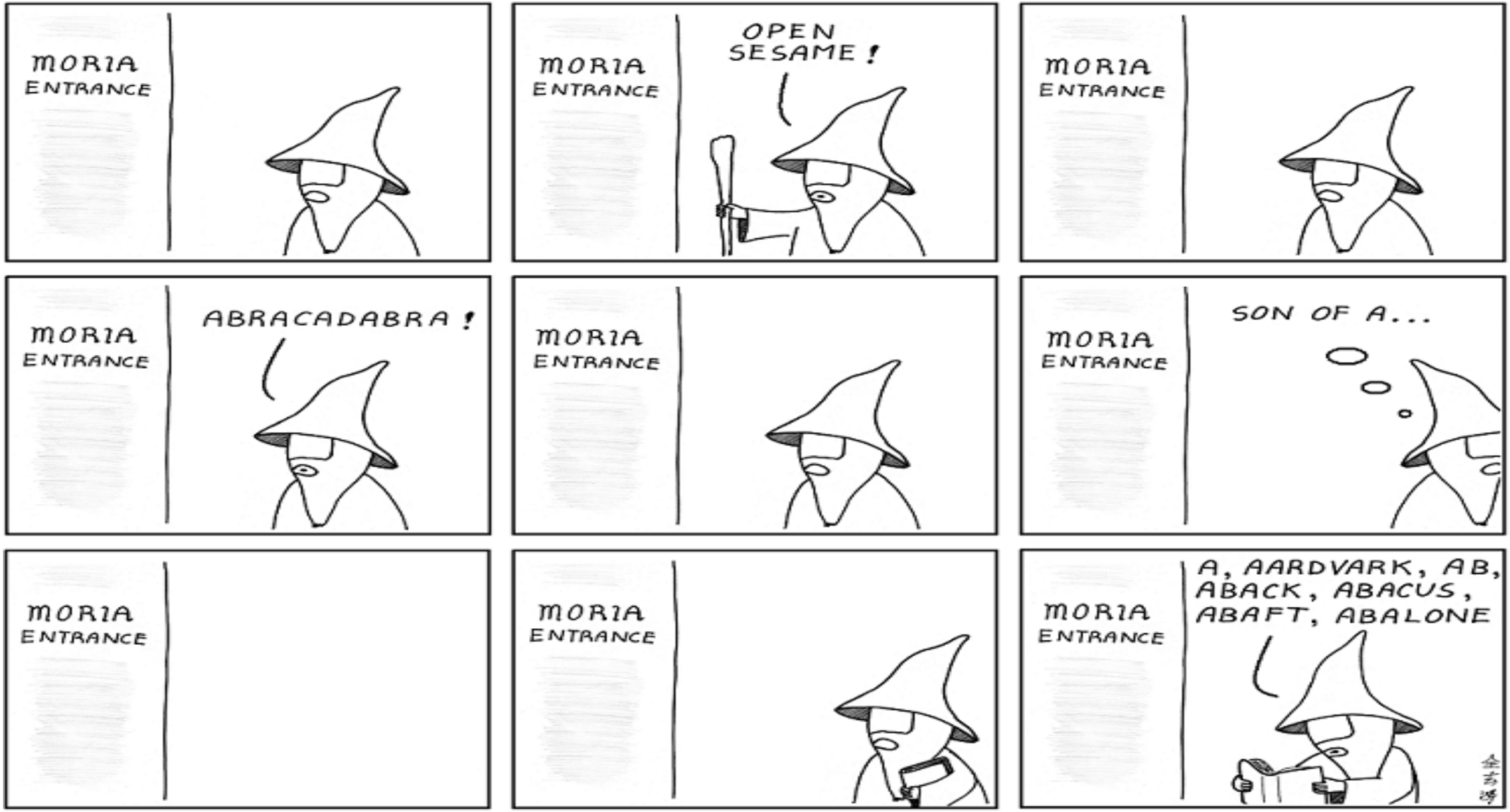
EFFICIENT HASHING

crypt, scrypt, bcrypt, PBKDF2

MD5 VS BCRYPT

10.000 iterations	user	system	total
MD5	0.07	0.0	0.07
bcrypt	22.23	0.08	22.31

GITHUB.COM/CODAHALE/BCRYPT-RUBY



SALTED HASHING

```
algorithm(data + salt) = hash
```

CONTENT

1. Introduction
2. Well-known security threats
3. Data Encryption
4. **Hardening Express_**
5. Authentication middleware
6. Great resources

USE STRICT

REGEX

OWASP.ORG/INDEX.PHP/REGULAR_EXPRESSION_DENIAL_OF_SERVICE_-_REDOS

X-POWERED-BY

NODE-UUID

GITHUB.COM/BROOFA/NODE-UUID

HTTP PARAMETER POLLUTION

```
GET /pay?amount=20&currency=EUR&amount=1
```

```
req.query.amount = ['20', '1'];
```

```
POST amount=20&currency=EUR&amount=1
```

```
req.body.amount = ['20', '1'];
```

BCRYPT

GITHUB.COM/NCBOOOGT/NODE.BCRYPT.JS

A BCRYPT GENERATED HASH

```
$2a$12$YKCxqK/QRgVfIIFeUtcPSOqyVGSorr1pHy5cZKsZuuc2g97bXgotS
```

GENERATING A HASH USING BCRYPT

```
bcrypt.hash('cronut', 12, function(err, hash) {
  // store hash
});

bcrypt.compare('cronut', hash, function(err, res) {
  if (res === true) {
    // password matches
  }
});
```

CSURF

GITHUB.COM/EXPRESSJS/CSURF

USING CSURF AS MIDDLEWARE

```
var csrf = require('csurf');
var csrfProtection = csrf({ cookie: false });

app.get('/form', csrfProtection, function(req, res) {
  res.render('form', { csrfToken: req.csrfToken() });
});

app.post('/login', csrfProtection, function(req, res) {
  // safe to continue
});
```

USING THE TOKEN IN YOUR TEMPLATE

```
extends layout

block content
  h1 CSRF protection using csurf
  form(action="/login" method="POST")
    input(type="text", name="username=", value="Username")
    input(type="password", name="password", value="Password")
    input(type="hidden", name="_csrf", value="#{csrfToken}")
    button(type="submit") Submit
```

HELMET

GITHUB.COM/HELMETJS/HELMET

USING HELMET WITH DEFAULT OPTIONS

```
var helmet = require('helmet');
app.use(helmet.noCache());
app.use(helmet.frameguard());
app.use(helmet.xssFilter());
…

// .. or use the default initialization
app.use(helmet());
```

HELMET FOR KOA

GITHUB.COM/VENABLES/KOA-HELMET

LUSCA

GITHUB.COM/KRAKENJS/LUSCA

APPLYING LUSCA AS MIDDLEWARE

```
var lusca = require('lusca');

app.use(lusca({
    csrf: true,
    csp: { /* ... */},
    xframe: 'SAMEORIGIN',
    p3p: 'ABCDEF',
    xssProtection: true
}));
```

LUSCA FOR KOA

GITHUB.COM/KOAJS/KOA-LUSCA

CONTENT

1. Introduction
2. Well-known security threats
3. Data Encryption
4. Hardening Express
5. **Authentication middleware_**
6. Great resources

TYPES OF EXPRESS MIDDLEWARE

1. Application-level
2. Route-level
3. Error-handling

WRITING CUSTOM MIDDLEWARE

```
var authenticate = function(req, res, next) {
  // check the request and modify response
};

app.get('/form', authenticate, function(req, res) {
  // assume that the user is authenticated
}

// … or use the middleware for certain routes
app.use('/admin', authenticate);
```

PASSPORT

GITHUB.COM/JAREDHANSON/PASSPORT

SETTING UP A PASSPORT STRATEGY

```
passport.use(new LocalStrategy(function(username, password, done) {
  User.findOne({ username: username }, function (err, user) {
    if (err) { return done(err); }
    if (!user) {
      return done(null, false, { message: 'Incorrect username.' });
    }
    if (!user.validPassword(password)) {
      return done(null, false, { message: 'Incorrect password.' });
    }
    return done(null, user);
  });
}));
```

USING PASSPORT STRATEGIES FOR AUTHENTICATION

```
// Simple authentication
app.post('/login', passport.authenticate(‘local'), function(req, res) {
  // req.user contains the authenticated user
  res.redirect('/user/' + req.user.username);
});

// Using redirects
app.post('/login', passport.authenticate('local', {
  successRedirect: ‘/',
  failureRedirect: ‘/login’,
  failureFlash: true
}));
```

NSP

NODESECURITY.IO/TOOLS

```
➜  vzero-node  nsp

Usage: [command] --arg=value --arg2

Help:
  help                          Show help menu
  [cmd] help                    Show command help menu

Options:
  version                       shows the current version of nsp
  shrinkwrap                    alias to audit-shrinkwrap
  audit-shrinkwrap              audits your `npm shrinkwrap` against NSP db
  package                       alias to audit-package
  audit-package                 audits your package.json against NSP db

➜  vzero-node  nsp shrinkwrap
Name       Installed   Patched   Vulnerable Dependency
semver       2.2.1     >=4.3.2   vzero-node > braintree
uglify-js    2.2.5    >= 2.4.24  vzero-node > jade > transformers
➜  vzero-node  nsp package
Name       Installed   Patched   Vulnerable Dependency
semver       2.2.1     >=4.3.2   vzero-node@0.0.0 > braintree@1.29.0 > semver@2.2.1
uglify-js    2.2.5    >= 2.4.24  vzero-node@0.0.0 > jade@1.11.0 > transformers@2.1.0 > uglify-js@2.2.5
➜  vzero-node
```

CONTENT

1. Introduction
2. Well-known security threats
3. Data Encryption
4. Hardening Express
5. Authentication middleware
6. **Great resources_**

PASSWORDLESS AUTH

MEDIUM.COM/@NINJUDD/PASSWORDS-ARE-OBSOLETE-9ED56D483EB

OWASP NODE GOAT

GITHUB.COM/OWASP/NODEGOAT

NODE SECURITY

NODESECURITY.IO/RESOURCES

FAST IDENTITY ONLINE

FIDOALLIANCE.ORG

SECURITY BEYOND CURRENT MECHANISMS

1. Something you have
2. Something you know
3. Something you are

Favor security too much over the experience and you'll make the website a pain to use.

SMASHINGMAGAZINE.COM/2012/10/26/PASSWORD-MASKING-HURT-SIGNUP-FORM

THANK YOU!

@SeraAndroid
tim@getbraintree.com

slideshare.com/paypal
braintreepayments.com/developers