

WEB SECURITY

Defense Against the Dark Arts 101



FUNDAMENTAL CONCERNS

Authentication

Trusting that someone is who they say they are.

Communication

Transferring data through potentially unreliable middlemen.

Authorization

Giving resource access to the right people.

Control

Limiting or understanding the capabilities of agents.

OWASP TOP 10 (2013)

- Injection** Server-side code execution
- Broken Authentication** Allows for impersonation
- XSS** Client-side code execution
- Direct References** Access control can be circumvented
- Security Misconfiguration** Vulnerable default/inherited settings
- Data Exposure** Data is insecurely transmitted, stored, or simply overshared
- Missing Access Control** Users can do things they shouldn't be allowed to do
- XSRF** Abuse the target website's trust in the browser
- Vulnerable Components** Third-party tools are vulnerable
- Unvalidated Redirects** Abusable open-ended forwarding

[source](#)

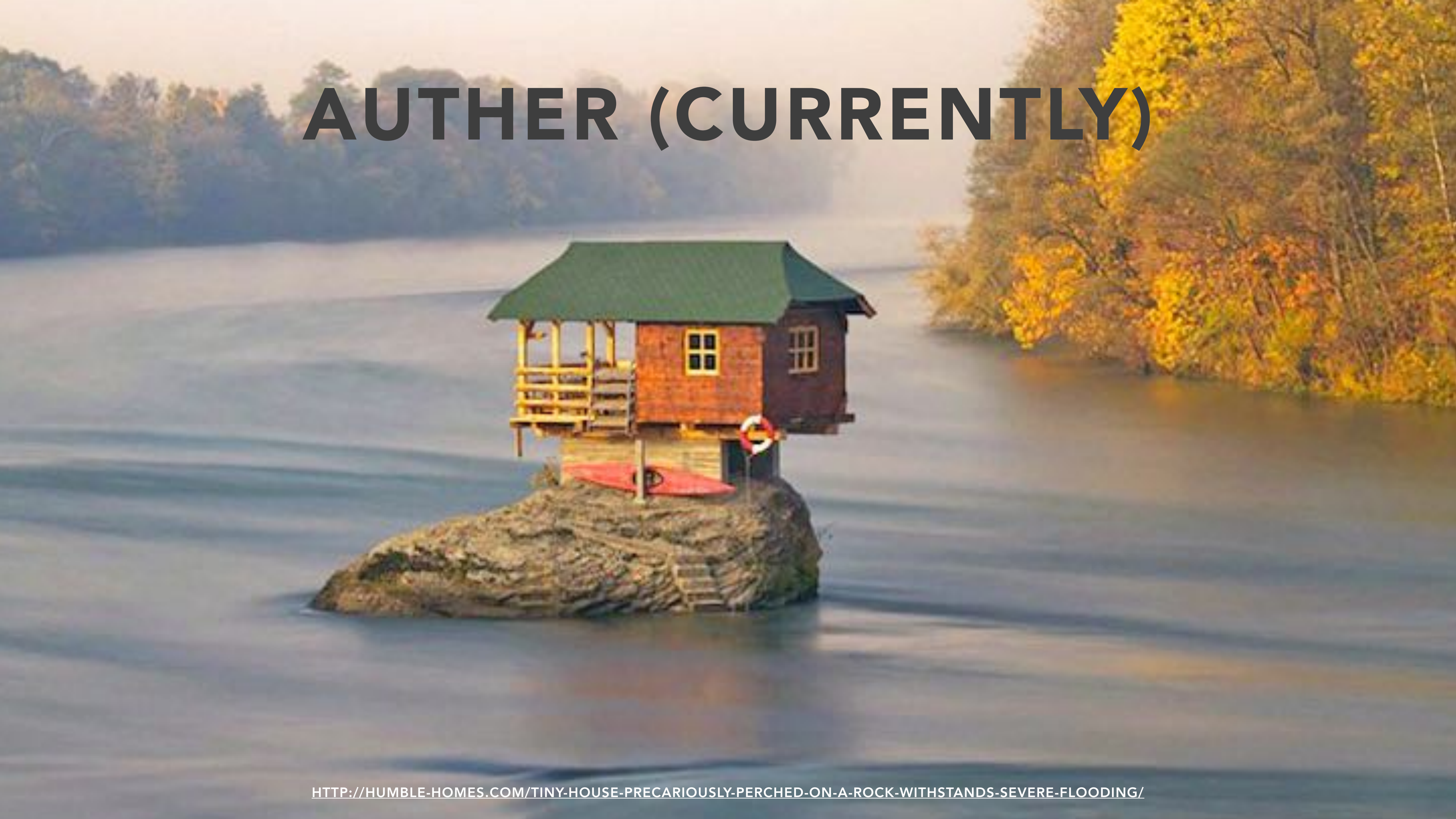
OWASP Top 10 - 2013	➔	OWASP Top 10 - 2017
A1 – Injection	➔	A1:2017-Injection
A2 – Broken Authentication and Session Management	➔	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	➡	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	➡	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	⊗	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	➔	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	⊗	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

AUTHER (CURRENTLY)



AUTHER (CURRENTLY)



[HTTP://HUMBLE-HOMES.COM/TINY-HOUSE-PRECARIOUSLY-PERCHED-ON-A-ROCK-WITHSTANDS-SEVERE-FLOODING/](http://humble-homes.com/tiny-house-precariouly-perched-on-a-rock-withstands-severe-flooding/)



WORKSHOP

- **Game with 5 rounds**
- **Each round**
 - **attack**
 - **attack demo**
 - **defense**
 - **defense review**



WORKSHOP

- **Game with 5 rounds**
- **Each round**
 - **attack**
 - **attack demo**
 - **defense**
 - **defense review**

uncovering secrets
improper access
injection
cross-site scripting
data theft

ROUND 1: UNCOVERING SECRETS (SECURITY MISCONFIGURATION)

40 lines (33 sloc) | 1.13 KB

Raw

Blame

History



```
1  'use strict';
2
3  var router = require('express').Router();
4  var passport = require('passport');
5  var GoogleStrategy = require('passport-google-oauth').OAuth2Strategy;
6
7  var User = require('../api/users/user.model');
8
9  router.get('/', passport.authenticate('google', {
10    scope: 'email'
11  }));
12
13  router.get('/callback', passport.authenticate('google', {
14    successRedirect: '/stories',
15    failureRedirect: '/signup'
16  }));
17
18  passport.use(new GoogleStrategy({
19    clientID: '238524570915-ivf9lnhm9bsfq13cle5ap8s28d4lmhrp.apps.googleusercontent.com',
20    clientSecret: 'GST6VQnVmhx1YIB1vDXXB3PF',
21    callbackURL: 'http://127.0.0.1:8080/auth/google/callback'
22  }, function (token, refreshToken, profile, done) {
23    var info = {
24      name: profile.displayName,
25      // google may not provide an email, if so we'll just fake it
26      email: profile.emails ? profile.emails[0].value : [profile.username, 'fake-author-email.com'].join('@'),
27      photo: profile.photos ? profile.photos[0].value : undefined
28    };
29    User.findOrCreate({
30      where: {googleId: profile.id},
31      defaults: info
32    })
```

40 lines (33 sloc) | 1.13 KB

Raw

Blame

History



```
1  'use strict';
2
3  var router = require('express').Router();
4  var passport = require('passport');
5  var GoogleStrategy = require('passport-google-oauth').OAuth2Strategy;
6
7  var User = require('../api/users/user.model');
8
9  router.get('/', passport.authenticate('google', {
10    scope: 'email'
11  }));
12
13  router.get('/callback', passport.authenticate('google', {
14    successRedirect: '/stories',
15    failureRedirect: '/signup'
16  }));
17
18  passport.use(new GoogleStrategy({
19    clientID: '238524570915-ivf9lnhm9bsfq13cle5ap8s28d4lmhrp.apps.googleusercontent.com',
20    clientSecret: 'GST6VQnVmhx1YIB1vDXXB3PF',
21    callbackURL: 'http://127.0.0.1:8080/auth/google/callback'
22  }, function (token, refreshToken, profile, done) {
23    var info = {
24      name: profile.displayName,
25      // google may not provide an email, if so we'll just fake it
26      email: profile.emails ? profile.emails[0].value : [profile.username, 'fake-author-email.com'].join('@'),
27      photo: profile.photos ? profile.photos[0].value : undefined
28    };
29    User.findOrCreate({
30      where: {googleId: profile.id},
31      defaults: info
32    })
```



40 lines (33 sloc) | 1.13 KB

Raw

Blame

History



```
1 'use strict';
2
3 var router = require('express').Router();
4 var passport = require('passport');
5 var GoogleStrategy = require('passport-google-oauth').OAuth2Strategy;
6
7 var User = require('../api/users/user.model');
8
9 router.get('/', passport.authenticate('google', {
10   scope: 'email'
11 }));
12
13 router.get('/callback', passport.authenticate('google', {
14   successRedirect: '/stories',
15   failureRedirect: '/signup'
16 }));
17
18 passport.use(new GoogleStrategy({
19   clientID: '238524570915-ivf9lnhm9bsfq13cle5ap8s28d4lmhrp.apps.googleusercontent.com',
20   clientSecret: 'GST6VQnVmhx1YIB1vDXXB3PF',
21   callbackURL: 'http://127.0.0.1:8080/auth/google/callback'
22 }, function (token, refreshToken, profile, done) {
23   var info = {
24     name: profile.displayName,
25     // google may not provide an email, if so we'll just fake it
26     email: profile.emails ? profile.emails[0].value : [profile.username, 'fake-author-email.com'].join('@'),
27     photo: profile.photos ? profile.photos[0].value : undefined
28   };
29   User.findOrCreate({
30     where: {googleId: profile.id},
31     defaults: info
32   })
```

lol





UNCOVERING SECRETS

- **Assume attacker has access to codebase**
- **Defense vulnerable if...**
 - **application secrets are easy to discover**
 - **files are improperly shared**
- **Bad stuff: app impersonation, decryption**



UNCOVERING SECRETS

- ◉ Assume attacker has access to codebase
- ◉ Defense vulnerable if...
 - ◉ application secrets are easy to discover
 - ◉ files are improperly shared
- ◉ Bad stuff: app impersonation, decryption

not (yet) about user secrets

ROUND 2: IMPROPER ACCESS (MISSING ACCESS CONTROL)



IMPROPER ACCESS

- **Assume attacker is client**
- **Defense vulnerable if client can act outside of authorization**
- **Bad stuff: depends on the action and resource**



IMPROPER ACCESS

- Assume attacker is client
- Defense vulnerable if client can **act outside of authorization**
- Bad stuff: depends on the action and resource

still not (yet) about user secrets



IMPROPER ACCESS

- **Frontend “access control” is actually just good UX**
- **True access control comes from backend**
- **Considerations (be thorough)...**
 - **requested resource**
 - **requested action**
 - **agent making the request**



```
fetch('/api/users/1', {  
  method: 'DELETE',  
  credentials: 'same-origin'  
});
```



```
fetch('/api/users/1', {  
  method: 'DELETE',  
  credentials: 'same-origin'  
});
```





```
fetch('/api/users/1', {  
  method: 'DELETE',  
  credentials: 'same-origin'  
});
```





BEWARE OVERPROTECTION

```
// "safe server"
var app = require('express')();
app.use(function (req, res) {
  res.status(403);
  res.send('NONE SHALL ENTER');
});
app.get('/', function (req, res) {
  res.sendFile(__dirname + '/index.html');
});
...
```



BEWARE OVERPROTECTION

```
// "safe server"
var app = require('express')();
app.use(function (req, res) {
  res.status(403);
  res.send('NONE SHALL ENTER');
});
app.get('/', function (req, res) {
  res.sendFile(__dirname + '/index.html');
});
...
```

My house is safe from intruders because it does not have a door.



BEWARE OVERPROTECTION

```
// "safe server"
var app = require('express')();
app.use(function (req, res) {
  res.status(403);
  res.send('NONE SHALL ENTER');
});
app.get('/', function (req, res) {
  res.sendFile(__dirname + '/index.html');
});
...
```

My house is safe from intruders because it does not have a door.

ROUND 3: INJECTION



INJECTION

- **Assume attacker is non-admin client**
- **Defense vulnerable if client can execute code on server**
- **Bad stuff: umm, everything?**



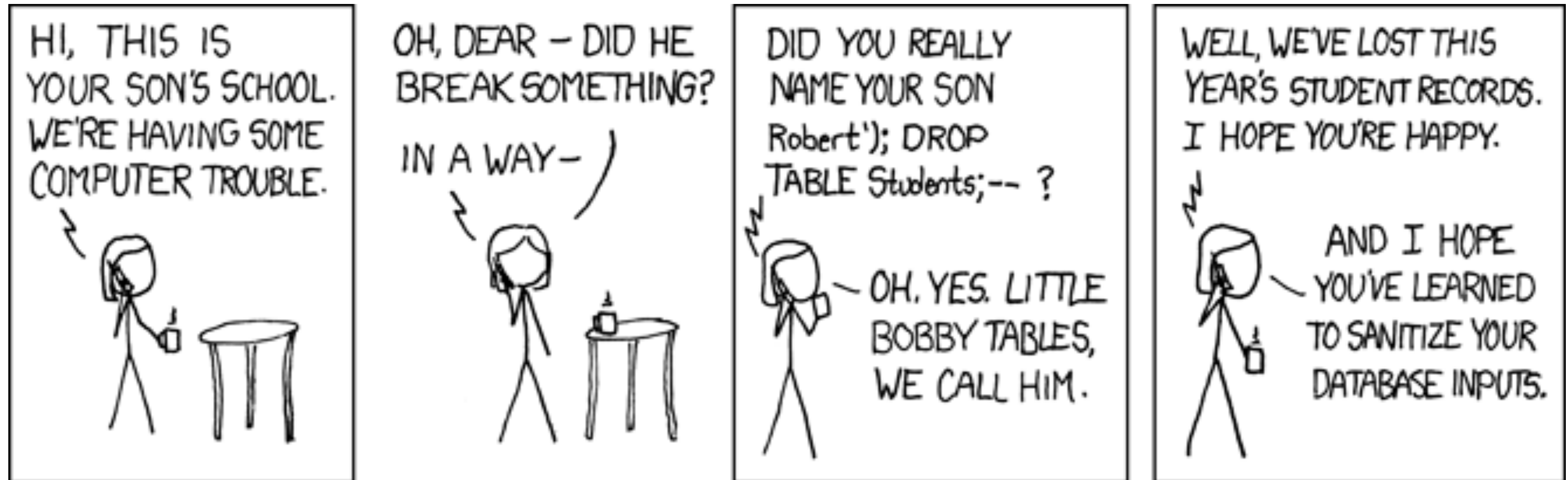
INJECTION

- Assume attacker is non-admin client
- Defense vulnerable if client can execute code on server
- Bad stuff: umm, everything?

not (yet) about executing code on client



INJECTION



[HTTPS://XKCD.COM/327/](https://xkcd.com/327/)



INJECTION *server-side execution of attacker-defined code*



```
...  
// at end of middleware stack  
app.use(function (req, res, next) {  
  // allow client to specify *any* module  
  var theModule = require(req.body.modulePath);  
  // allow client to specify *any* method  
  var method = theModule[req.body.methodName];  
  // allow client to specify *any* arguments  
  var args = req.body.args;  
  // blindly invoke!  
  var result = method.apply(theModule, args);  
  res.json(result);  
});  
...
```





INJECTION *server-side execution of attacker-defined code*



```
...  
// at end of middleware stack  
app.use(function (req, res, next) {  
  // allow client to specify *any* module  
  var theModule = require(req.body.modulePath);  
  // allow client to specify *any* method  
  var method = theModule[req.body.methodName];  
  // allow client to specify *any* arguments  
  var args = req.body.args;  
  // blindly invoke!  
  var result = method.apply(theModule, args);  
  res.json(result);  
});  
...
```

```
POST /whatever HTTP/1.1  
...  
{  
  "modulePath": "../user.model",  
  "method": "destroy",  
  "args": [{}]  
}
```





INJECTION *server-side execution of attacker-defined code*



User.destroy({});

```
...  
// at end of middleware stack  
app.use(function (req, res, next) {  
  // allow client to specify *any* module  
  var theModule = require(req.body.modulePath);  
  // allow client to specify *any* method  
  var method = theModule[req.body.methodName];  
  // allow client to specify *any* arguments  
  var args = req.body.args;  
  // blindly invoke!  
  var result = method.apply(theModule, args);  
  res.json(result);  
});  
...
```

```
POST /whatever HTTP/1.1  
...  
{  
  "modulePath": "../user.model",  
  "method": "destroy",  
  "args": [{}]  
}
```



ROUND 4:

CROSS-SITE SCRIPTING



XSS

- **Assume attacker is non-admin client**
- **Defense vulnerable if client can execute code on *another* client**
- **Bad stuff: yeah pretty much everything**
- **Two flavors: “stored” and “reflected”**



STORED XSS



4 Comments

The Atlantic

1 Login ▾

♥ Recommend

↗ Share

Sort by Oldest ▾



<script>alert('i am an xss attack');</script>

LOG IN WITH



OR SIGN UP WITH DISQUS (?)



STORED XSS



4 Comments

The Atlantic

1 Login ▾

Recommend

Share

Sort by Oldest ▾

submit



`<script>alert('i am an xss attack');</script>`

LOG IN WITH



OR SIGN UP WITH DISQUS (?)

Name



STORED XSS



4 Comments

The Atlantic

1 Login ▾

Recommend

Share

Sort by Oldest ▾

submit



```
<script>alert('xss attack');</script>
```



<script>alert('i am an xss attack');</script>

LOG IN WITH



OR SIGN UP WITH DISQUS ?

Name



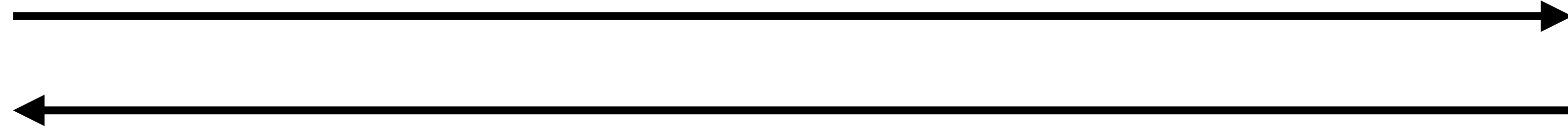
STORED XSS



```
<script>alert('xss attack');</script>
```



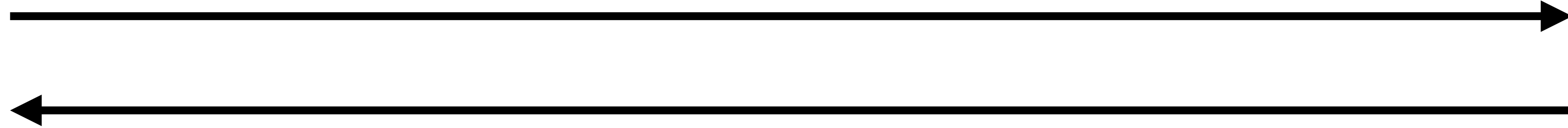
STORED XSS



```
<script>alert('xss attack');</script>
```



STORED XSS



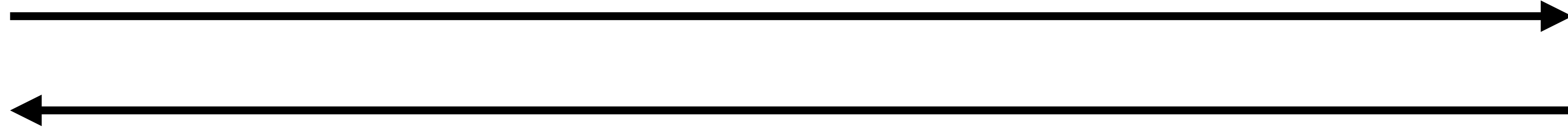
```
<script>alert('xss attack');</script>
```

```
<script>alert('xss attack');</script>
```





STORED XSS



`alert('xss attack');`

`<script>alert('xss attack');</script>`





STORED XSS





REFLECTED XSS

- **Server sends parts of request in response**
- **Attacker forms link with script in it**
- **Victim clicks link**
- **Server responds with script**
- **Script runs on victim's browser**



REFLECTED XSS

[https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_\(OTG-INPVAL-001\)](https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_(OTG-INPVAL-001))



REFLECTED XSS



[https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_\(OTG-INPVAL-001\)](https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_(OTG-INPVAL-001))



REFLECTED XSS

```
http://example.com/index.php?user=<script>window.onload = function() {var  
AllLinks=document.getElementsByTagName("a");  
AllLinks[0].href = "http://badexample.com/malicious.exe"; }</script>
```





REFLECTED XSS

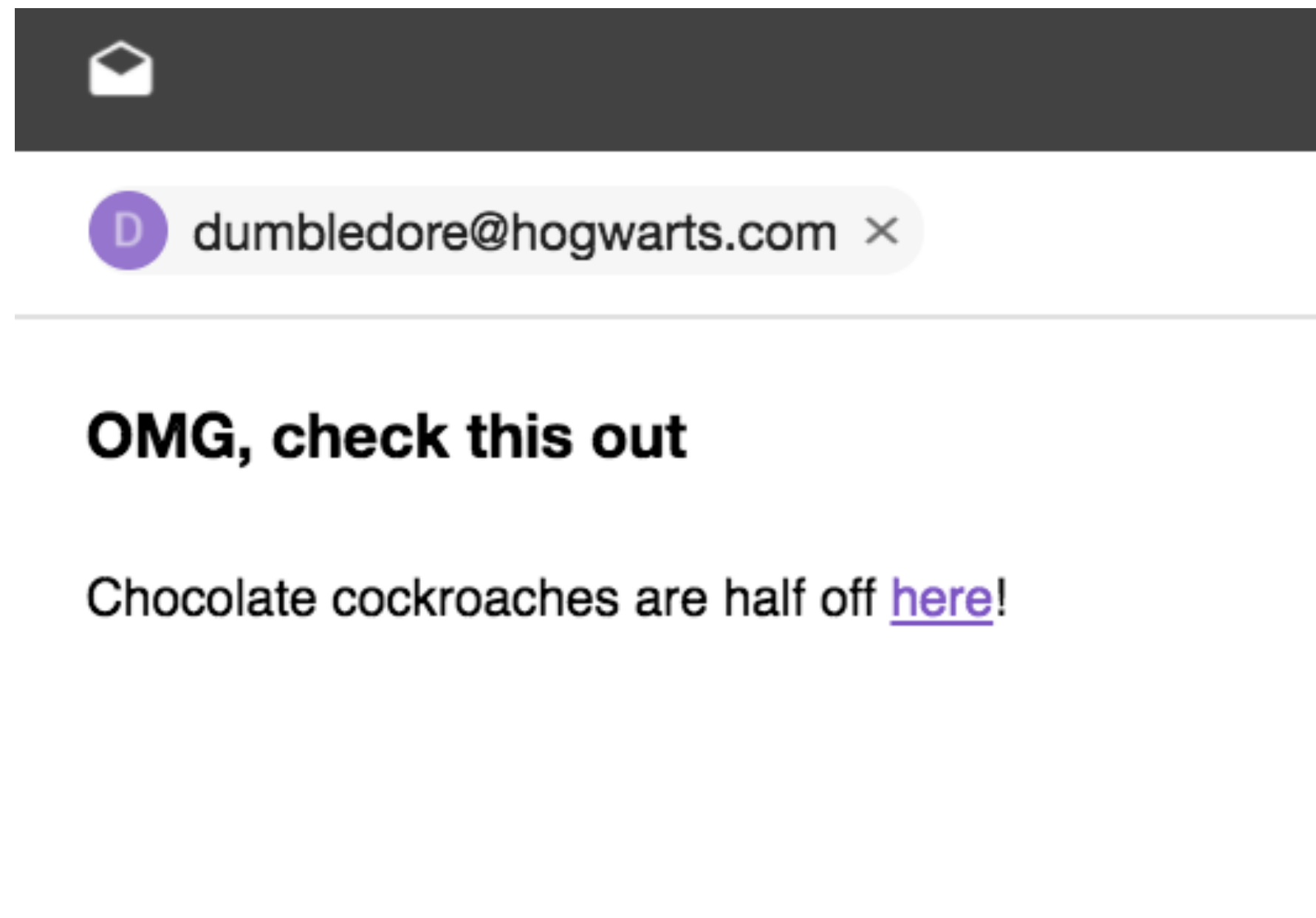


```
http://example.com/index.php?user=<script>window.onload = function() {var  
AllLinks=document.getElementsByTagName( "a" );  
AllLinks[0].href = "http://badexample.com/malicious.exe"; }</script>
```





REFLECTED XSS

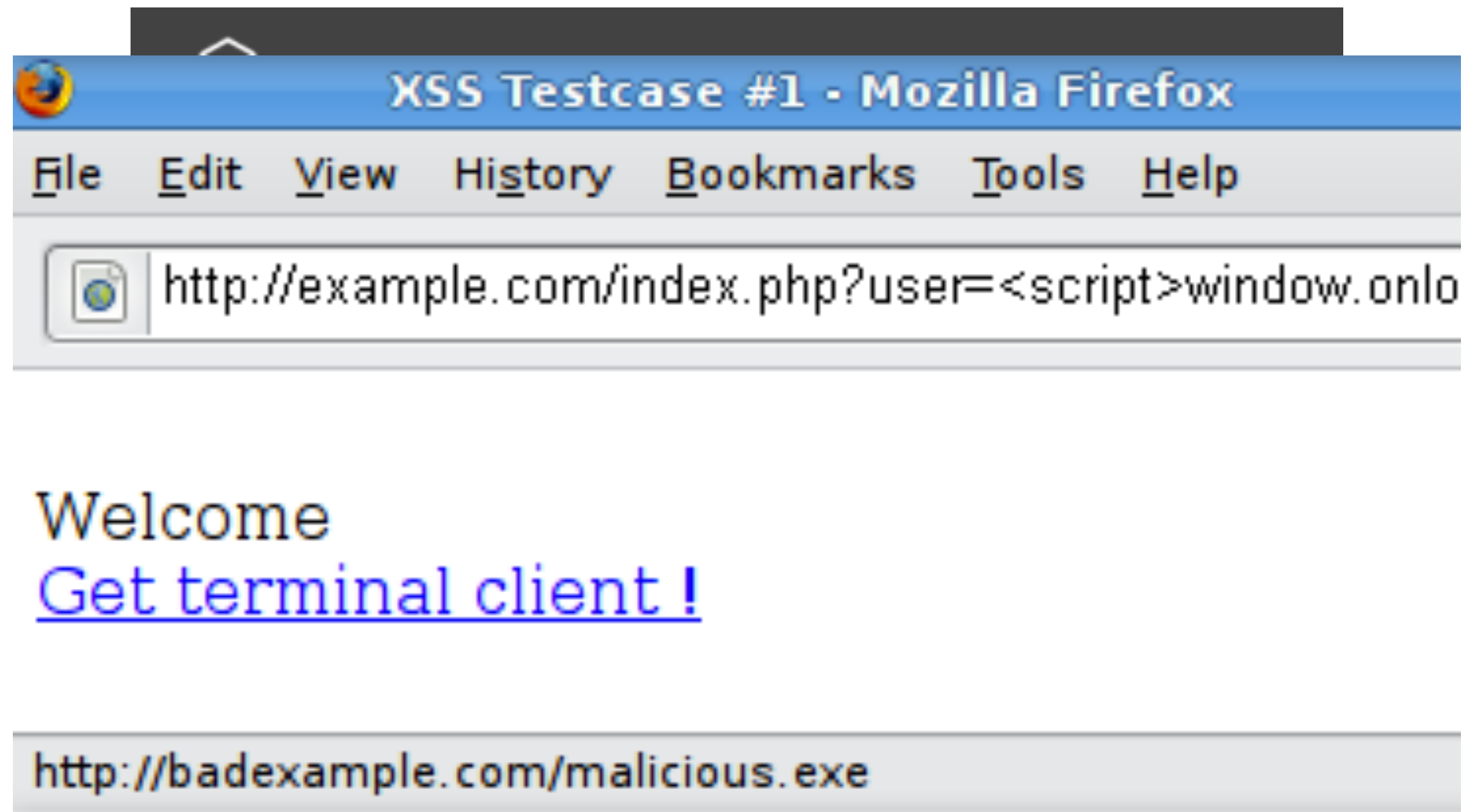


```
http://example.com/index.php?user=<script>window.onload = function() {var  
AllLinks=document.getElementsByTagName("a");  
AllLinks[0].href = "http://badexample.com/malicious.exe"; }</script>
```





REFLECTED XSS

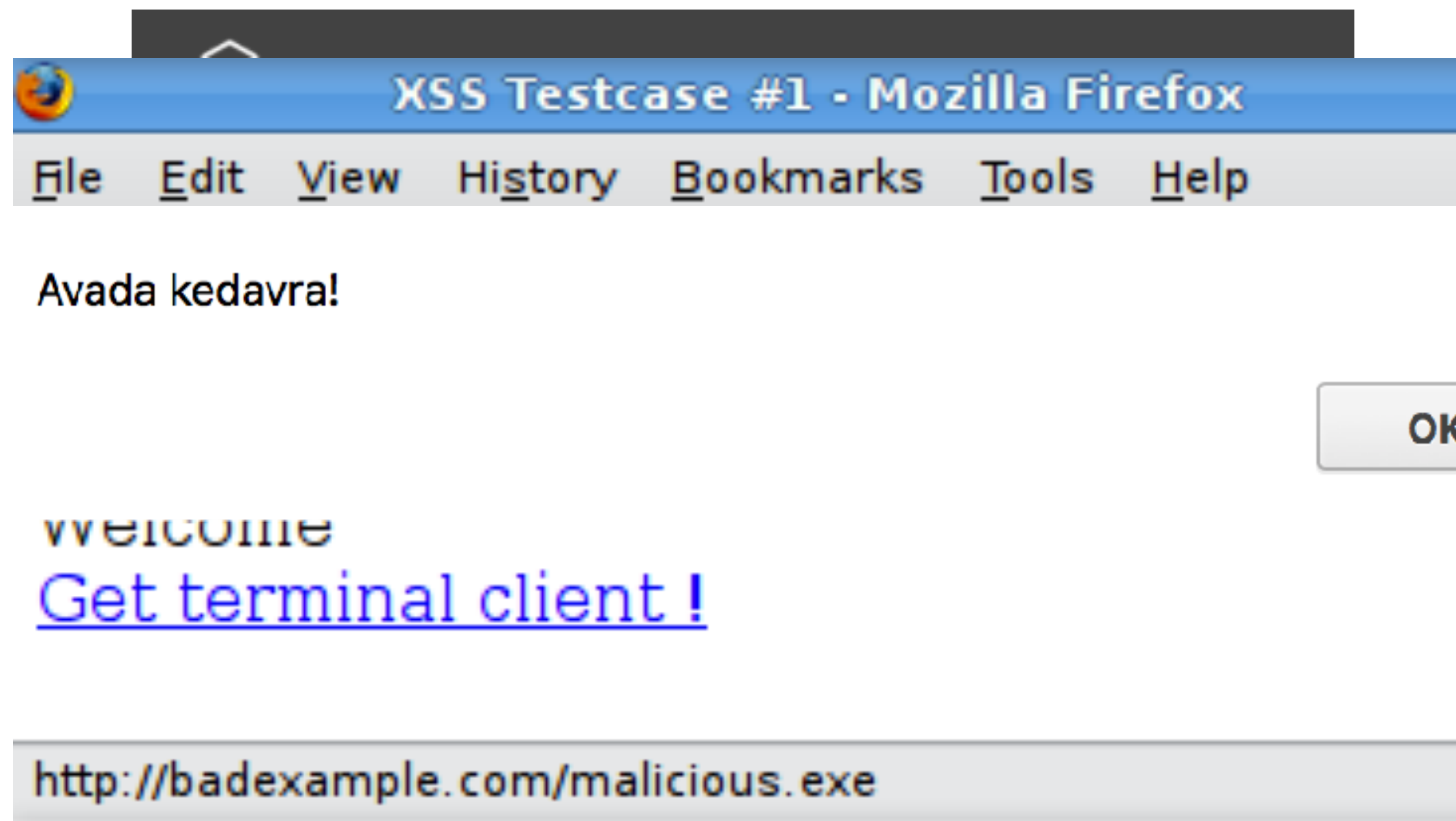


```
http://example.com/index.php?user=<script>>window.onload = function() {var  
AllLinks=document.getElementsByTagName("a");  
AllLinks[0].href = "http://badexample.com/malicious.exe"; }</script>
```





REFLECTED XSS

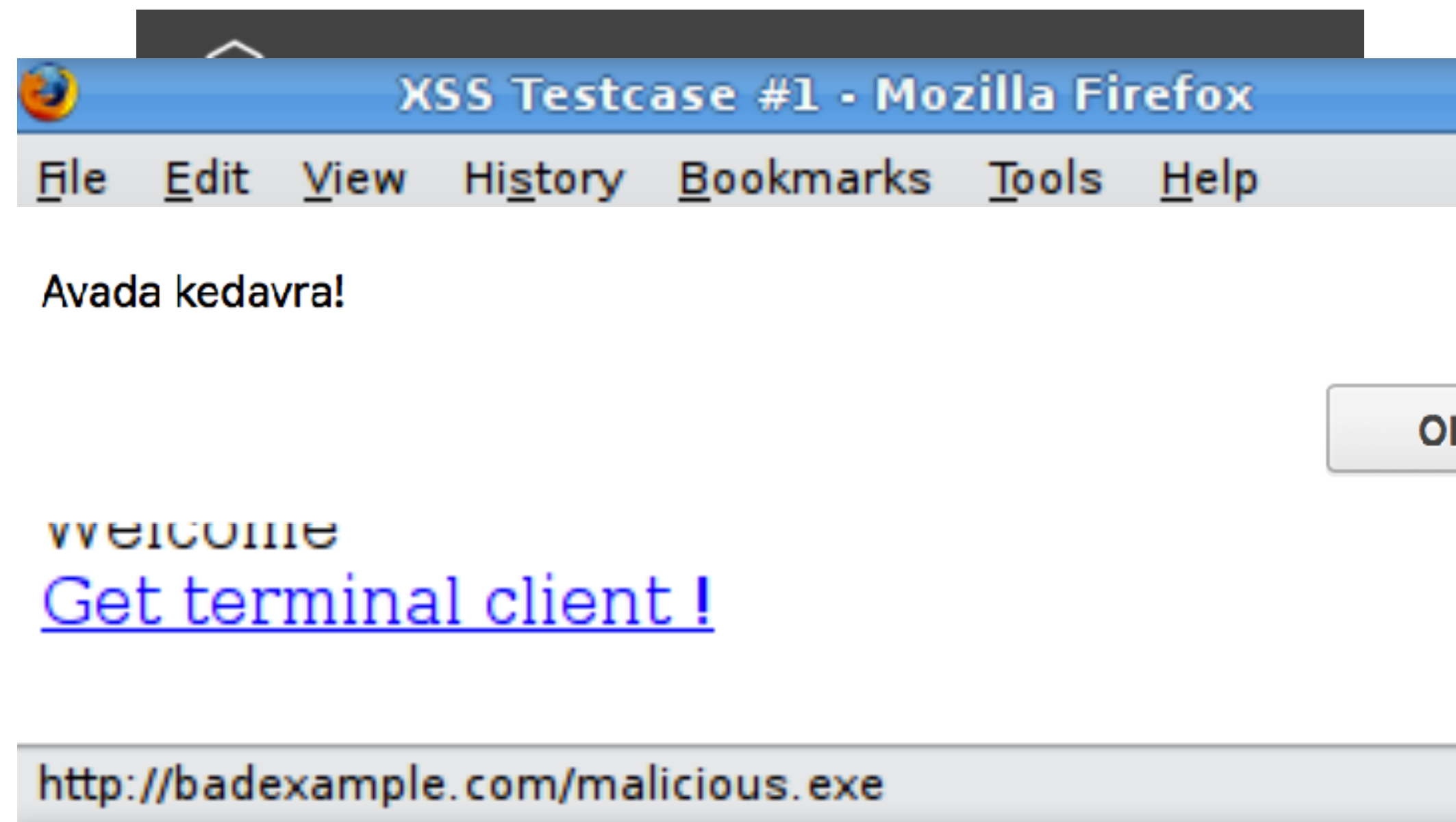


```
http://example.com/index.php?user=<script>window.onload = function() {var  
AllLinks=document.getElementsByTagName("a");  
AllLinks[0].href = "http://badexample.com/malicious.exe"; }</script>
```





REFLECTED XSS



```
http://example.com/index.php?user=<script>window.onload = function() {var  
AllLinks=document.getElementsByTagName("a");  
AllLinks[0].href = "http://badexample.com/malicious.exe"; }</script>
```



ROUND 5: DATA THEFT (BAD AUTH / DATA EXPOSURE)



"IDENTITY THEFT"

- Assume attacker has access to *middlemen* and *database*
- Defense vulnerable if communication or storage exposes passwords
- Bad stuff: user impersonation



PASSWORD SECURITY

THE WORST WAY

create

p@ssword^{HTTP}→ p@ssword → p@ssword

database

p@ssword

verify

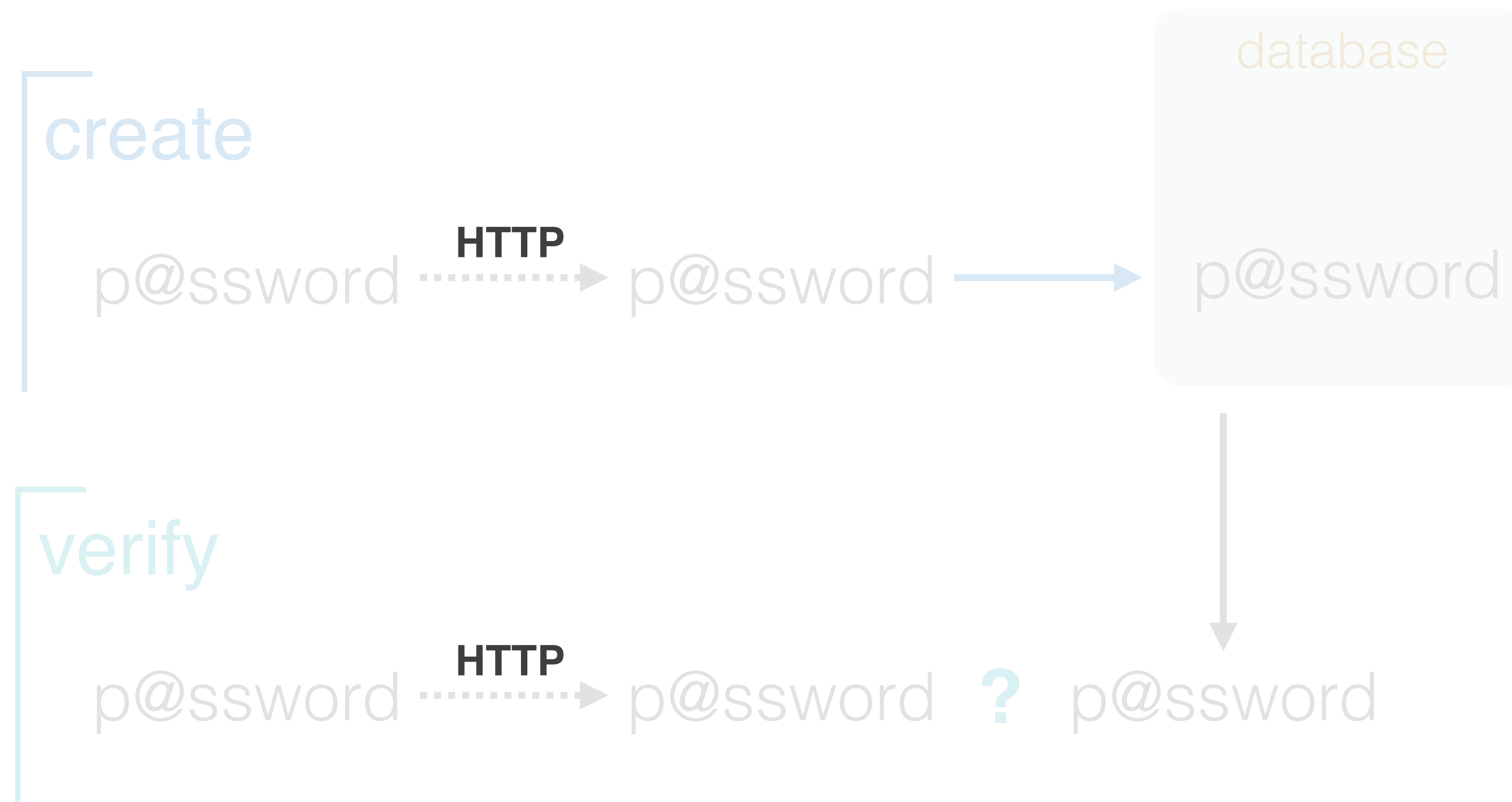
p@ssword^{HTTP}→ p@ssword ? p@ssword





PASSWORD SECURITY

THE WORST WAY

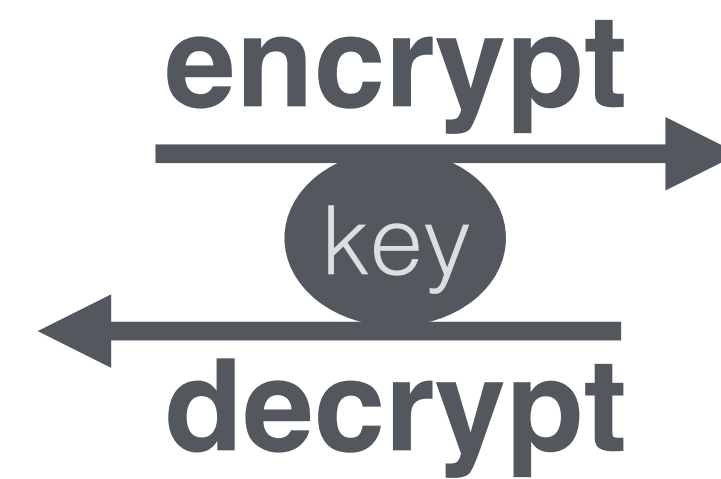




ENCRYPTION

plaintext **scheme** **ciphertext**

word



981kje

HTTP
encryption
+ “authentic” server

HTTPS

HTTP
encryption
+ “authentic” server } SSL

HTTPS

"AUTHENTIC" SERVER

- **Has an SSL certificate**
- **Digitally signed**
- **Forms a web of trust**
- **Self-signed in development**

HTTPS IN PRACTICE

With openssl generate a private key with...

```
$ openssl genrsa -out key.pem 2048
```

Then generate self-signed SSL certificate with...

```
$ openssl req -x509 -new -nodes -key key.pem -days 1024 -out cert.pem
```

Now use node's `https` library with this key and certificate



PASSWORD SECURITY

SECURE COMMUNICATION

create



verify





PASSWORD SECURITY

SECURE-ISH STORAGE

create



verify





PASSWORD SECURITY

SECURE-ISH STORAGE

create

p@ssword^{HTTPS}→ p@ssword →^{encrypt}  database
aosi0310

verify

p@ssword^{HTTPS}→ p@ssword ? p@ssword



decrypt

What do we do with the key??? Uh oh...



HASHING

string

algorithm

hash

word



jads82



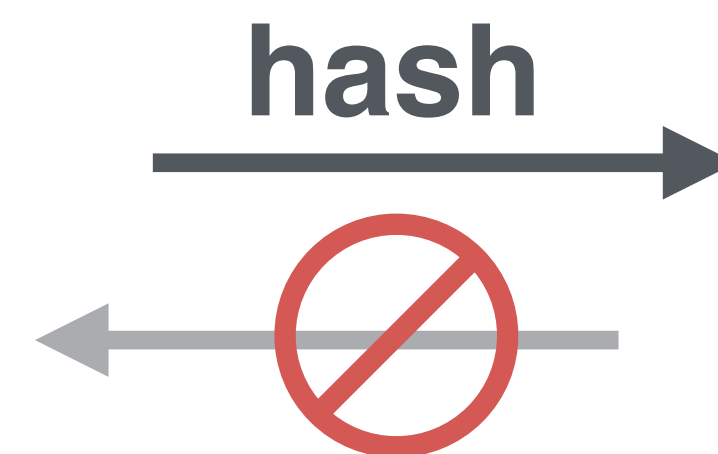
HASHING

string

algorithm

hash

word



jads82



PASSWORD SECURITY

SECURE-ISH STORAGE

create



verify





PASSWORD SECURITY

SECURE STORAGE

create



verify





GOOD HASH KEY FUNCTION

Should be slow

Ultimately, this will hurt would-be attackers more than it will hurt you.

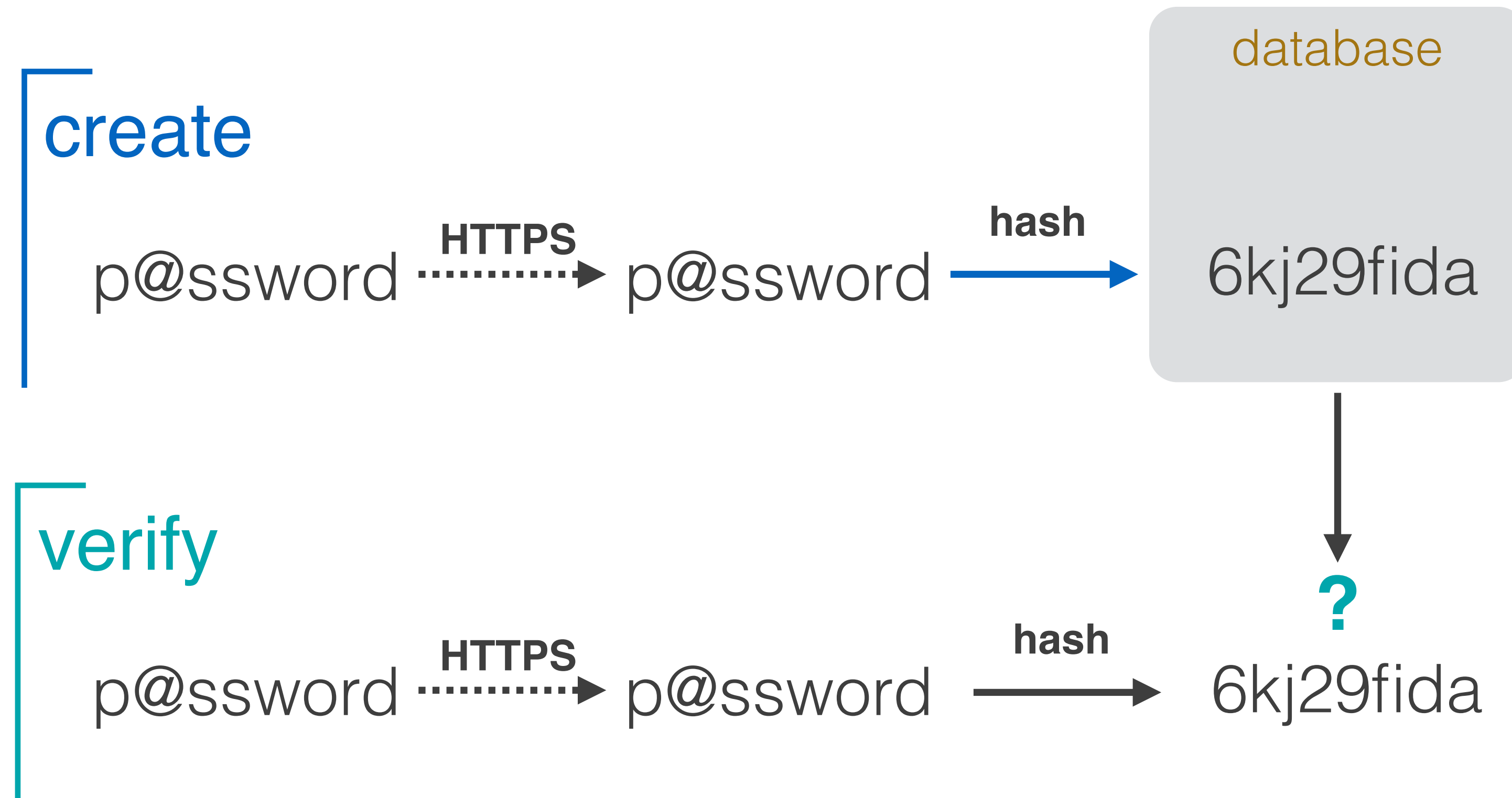
Should have a low collision frequency

Fewest possible duplicates. Or else two inputs could be indistinguishable.



PASSWORD SECURITY

Can we make it even better? **SECURE STORAGE**





SALT

- **Random string, unique to each user**
- **Added before hashing password**
- **Two users with the same password have different hashes**
- **Hashes are not computable ahead-of-time**



PASSWORD SECURITY

SECURE STORAGE

create



verify

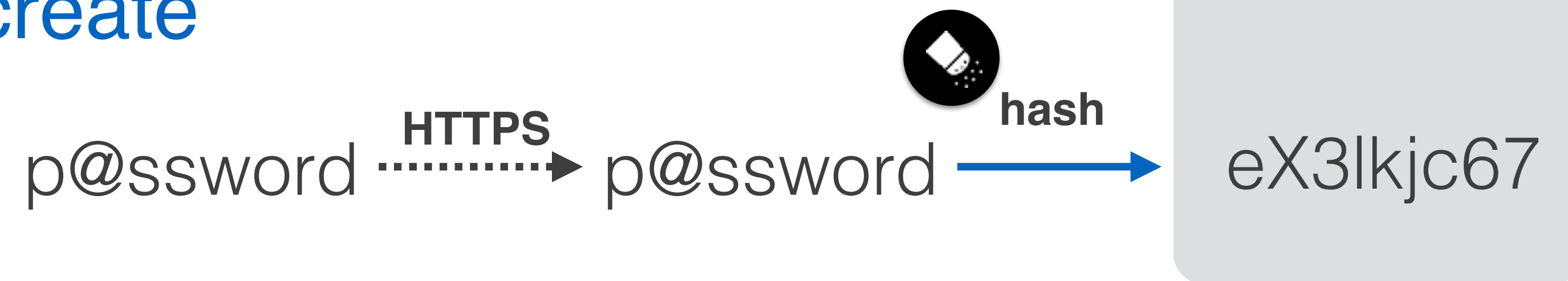




PASSWORD SECURITY

SECURE STORAGE

create



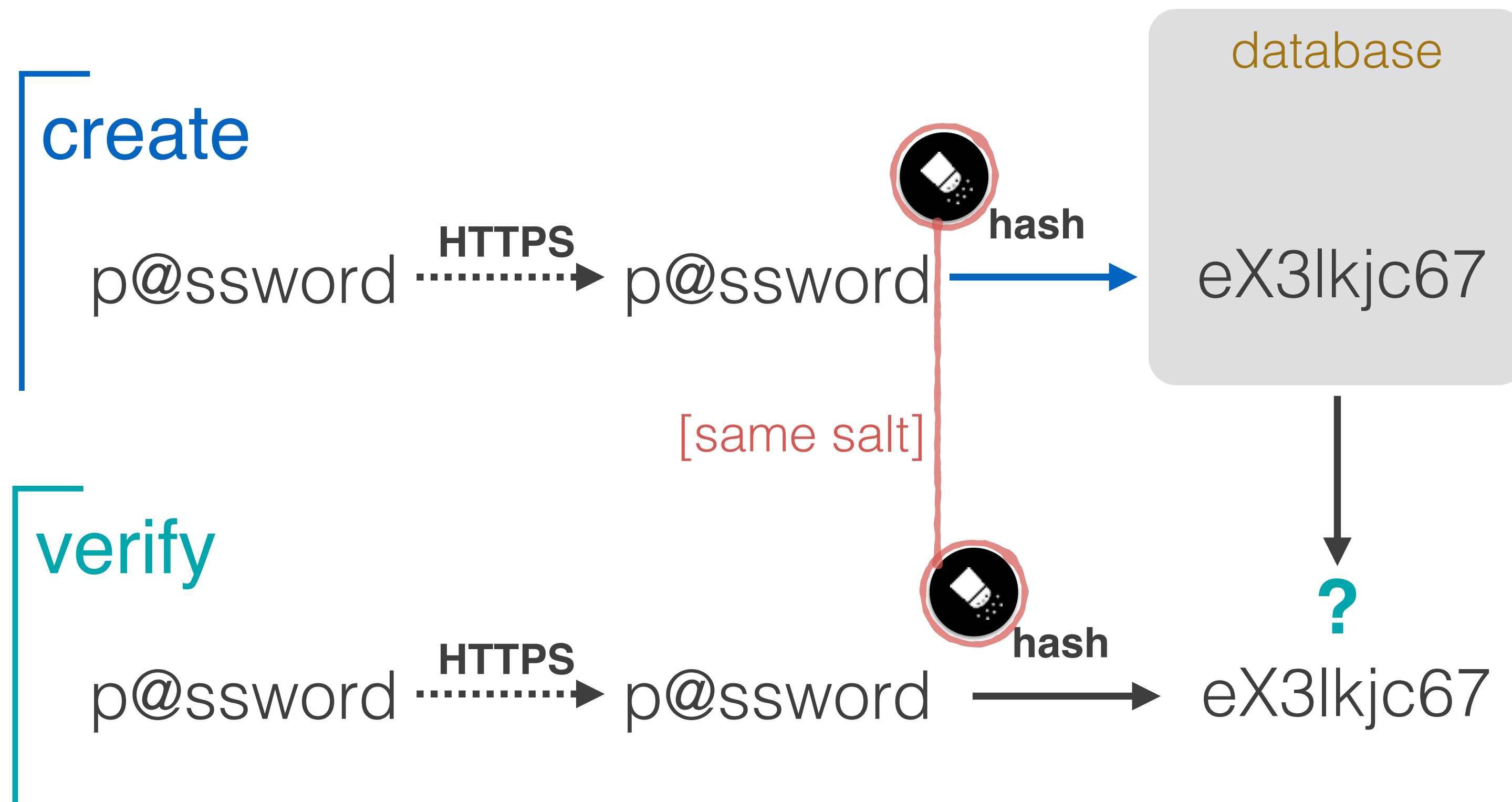
verify





PASSWORD SECURITY

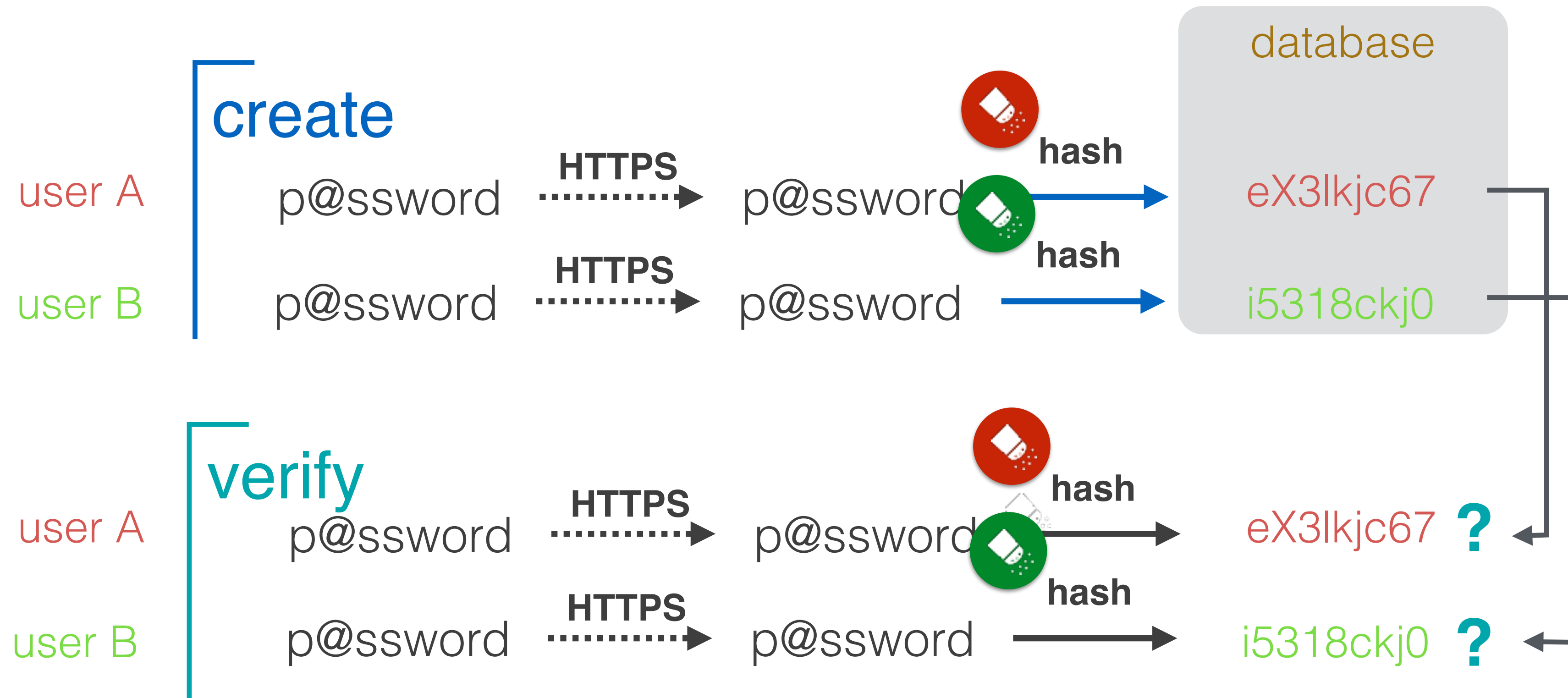
SECURE STORAGE





PASSWORD SECURITY

SECURE STORAGE



HASHING IN PRACTICE

```
// built-in node library
var crypto = require('crypto');
var salt = crypto.randomBytes(16);
var iterations = 1;
var bytes = 64;
var buffer = crypto.pbkdf2Sync('password', salt, iterations, bytes);
var hash = buffer.toString('base64');
// ...
```

```

User.generateSalt = function () {
  return crypto.randomBytes(16).toString('base64')
}

User.encryptPassword = function (plainText, salt) {
  return crypto
    .createHash('RSA-SHA256')
    .update(plainText)
    .update(salt)
    .digest('hex')
}

const setSaltAndPassword = user => {
  if (user.changed('password')) {
    user.salt = User.generateSalt()
    user.password = User.encryptPassword(user.password, user.salt)
  }
}

User.beforeCreate(setSaltAndPassword)
User.beforeUpdate(setSaltAndPassword)

```




OTHER CONSIDERATIONS

- **Broken authentication flow**
- **Oversharing data**



BROKEN AUTH

- **Assume attacker is guest client**
- **Defense vulnerable if signup or login are improper**
- **Bad stuff: user impersonation, admin impersonation**



OVERSHARING

- **Assume attacker is non-admin client**
- **Defense vulnerable if attacker can see sensitive info of another**
- **Bad stuff: failed user privacy, NSA loves you**







```
curl http://hogwarts.com/api/teachers
```





```
curl http://hogwarts.com/api/teachers
```

```
[  
  {  
    name: 'Dumbledore',  
    position: 'headmaster',  
    weaknesses: ['candy']  
  },  
  {  
    name: 'Snape',  
    position: 'Potions Master',  
    weaknesses: ['unrequited love', 'conditioner']  
  }  
]
```





```
curl http://hogwarts.com/api/teachers
```

```
[  
  {  
    name: 'Dumbledore',  
    position: 'headmaster',  
    weaknesses: ['candy']  
  },  
  {  
    name: 'Snape',  
    position: 'Potions Master',  
    weaknesses: ['unrequited love', 'conditioner']  
  }  
]
```

pwned





GOOD AUTH

- **Communication is secure**
- **Storage is secure**
- **Cannot set privileges via signup**
- **Logging in *requires* username and password**
- **Data is not inadvertently shared**

RECAP



FUNDAMENTAL CONCERNS

Authentication

Trusting that someone is who they say they are.

Communication

Transferring data through potentially unreliable middlemen.

Authorization

Giving resource access to the right people.

Control

Limiting or understanding the capabilities of agents.

OWASP TOP 10 (2013)

Server-side code execution

Allows for impersonation

Client-side code execution

Access control can be circumvented

Vulnerable default/inherited settings

Data is insecurely transmitted, stored, or simply overshared

Users can do things they shouldn't be allowed to do

Abuse the target website's trust in the browser

Third-party tools are vulnerable

Abusable open-ended forwarding

[source](#)

OWASP TOP 10 (2013)

Injection Server-side code execution

Allows for impersonation

Client-side code execution

Access control can be circumvented

Vulnerable default/inherited settings

Data is insecurely transmitted, stored, or simply overshared

Users can do things they shouldn't be allowed to do

Abuse the target website's trust in the browser

Third-party tools are vulnerable

Abusable open-ended forwarding

[source](#)

OWASP TOP 10 (2013)

Injection Server-side code execution

Broken Authentication Allows for impersonation

Client-side code execution

Access control can be circumvented

Vulnerable default/inherited settings

Data is insecurely transmitted, stored, or simply overshared

Users can do things they shouldn't be allowed to do

Abuse the target website's trust in the browser

Third-party tools are vulnerable

Abusable open-ended forwarding

[source](#)

OWASP TOP 10 (2013)

Injection Server-side code execution

Broken Authentication Allows for impersonation

XSS Client-side code execution

Access control can be circumvented

Vulnerable default/inherited settings

Data is insecurely transmitted, stored, or simply overshared

Users can do things they shouldn't be allowed to do

Abuse the target website's trust in the browser

Third-party tools are vulnerable

Abusable open-ended forwarding

[source](#)

OWASP TOP 10 (2013)

Injection Server-side code execution

Broken Authentication Allows for impersonation

XSS Client-side code execution

Direct References Access control can be circumvented

Vulnerable default/inherited settings

Data is insecurely transmitted, stored, or simply overshared

Users can do things they shouldn't be allowed to do

Abuse the target website's trust in the browser

Third-party tools are vulnerable

Abusable open-ended forwarding

[source](#)

OWASP TOP 10 (2013)

Injection Server-side code execution

Broken Authentication Allows for impersonation

XSS Client-side code execution

Direct References Access control can be circumvented

Security Misconfiguration Vulnerable default/inherited settings

Data is insecurely transmitted, stored, or simply overshared

Users can do things they shouldn't be allowed to do

Abuse the target website's trust in the browser

Third-party tools are vulnerable

Abusable open-ended forwarding

[source](#)

OWASP TOP 10 (2013)

Injection Server-side code execution

Broken Authentication Allows for impersonation

XSS Client-side code execution

Direct References Access control can be circumvented

Security Misconfiguration Vulnerable default/inherited settings

Data Exposure Data is insecurely transmitted, stored, or simply overshared

Users can do things they shouldn't be allowed to do

Abuse the target website's trust in the browser

Third-party tools are vulnerable

Abusable open-ended forwarding

[source](#)

OWASP TOP 10 (2013)

- Injection** Server-side code execution
- Broken Authentication** Allows for impersonation
- XSS** Client-side code execution
- Direct References** Access control can be circumvented
- Security Misconfiguration** Vulnerable default/inherited settings
- Data Exposure** Data is insecurely transmitted, stored, or simply overshared
- Missing Access Control** Users can do things they shouldn't be allowed to do
 - Abuse the target website's trust in the browser
 - Third-party tools are vulnerable
 - Abusable open-ended forwarding

[source](#)

OWASP TOP 10 (2013)

- Injection** Server-side code execution
- Broken Authentication** Allows for impersonation
- XSS** Client-side code execution
- Direct References** Access control can be circumvented
- Security Misconfiguration** Vulnerable default/inherited settings
- Data Exposure** Data is insecurely transmitted, stored, or simply overshared
- Missing Access Control** Users can do things they shouldn't be allowed to do
 - XSRF** Abuse the target website's trust in the browser
 - Third-party tools are vulnerable
 - Abusable open-ended forwarding

[source](#)

OWASP TOP 10 (2013)

- Injection** Server-side code execution
- Broken Authentication** Allows for impersonation
- XSS** Client-side code execution
- Direct References** Access control can be circumvented
- Security Misconfiguration** Vulnerable default/inherited settings
- Data Exposure** Data is insecurely transmitted, stored, or simply overshared
- Missing Access Control** Users can do things they shouldn't be allowed to do
- XSRF** Abuse the target website's trust in the browser
- Vulnerable Components** Third-party tools are vulnerable
 - Abusable open-ended forwarding

[source](#)

OWASP TOP 10 (2013)

- Injection** Server-side code execution
- Broken Authentication** Allows for impersonation
- XSS** Client-side code execution
- Direct References** Access control can be circumvented
- Security Misconfiguration** Vulnerable default/inherited settings
- Data Exposure** Data is insecurely transmitted, stored, or simply overshared
- Missing Access Control** Users can do things they shouldn't be allowed to do
- XSRF** Abuse the target website's trust in the browser
- Vulnerable Components** Third-party tools are vulnerable
- Unvalidated Redirects** Abusable open-ended forwarding

[source](#)

weak **authentication**

poor **authorization**

insecure **communication**

breakdown of **control**

Injection Server-side code execution

Broken Authentication Allows for impersonation

XSS Client-side code execution

Direct References Access control can be circumvented

Security Misconfiguration Vulnerable default/inherited settings

Data Exposure Data is insecurely transmitted, stored, or simply overshared

Missing Access Control Users can do things they shouldn't be allowed to do

XSRF Abuse the target website's trust in the browser

Vulnerable Components Third-party tools are vulnerable

Unvalidated Redirects Abusable open-ended forwarding

[source](#)

weak **authentication**

poor **authorization**

insecure **communication**

breakdown of **control**

Injection Server-side code execution

Broken Authentication Allows for impersonation

XSS Client-side code execution

Direct References Access control can be circumvented

Security Misconfiguration Vulnerable default/inherited settings

Data Exposure Data is insecurely transmitted, stored, or simply overshared

Missing Access Control Users can do things they shouldn't be allowed to do

XSRF Abuse the target website's trust in the browser

Vulnerable Components Third-party tools are vulnerable

Unvalidated Redirects Abusable open-ended forwarding

[source](#)

weak **authentication**

poor **authorization**

insecure **communication**

breakdown of **control**

Injection Server-side code execution

Broken Authentication Allows for impersonation

XSS Client-side code execution

Direct References Access control can be circumvented

Security Misconfiguration Vulnerable default/inherited settings

Data Exposure Data is insecurely transmitted, stored, or simply overshared

Missing Access Control Users can do things they shouldn't be allowed to do

XSRF Abuse the target website's trust in the browser

Vulnerable Components Third-party tools are vulnerable

Unvalidated Redirects Abusable open-ended forwarding

[source](#)

weak **authentication**

poor **authorization**

insecure **communication**

breakdown of **control**

Injection Server-side code execution

Broken Authentication Allows for impersonation

XSS Client-side code execution

Direct References Access control can be circumvented

Security Misconfiguration Vulnerable default/inherited settings

Data Exposure Data is insecurely transmitted, stored, or simply overshared

Missing Access Control Users can do things they shouldn't be allowed to do

XSRF Abuse the target website's trust in the browser

Vulnerable Components Third-party tools are vulnerable

Unvalidated Redirects Abusable open-ended forwarding

[source](#)

weak **authentication**

poor **authorization**

insecure **communication**

breakdown of **control**

Injection Server-side code execution

Broken Authentication Allows for impersonation

XSS Client-side code execution

Direct References Access control can be circumvented

Security Misconfiguration Vulnerable default/inherited settings

Data Exposure Data is insecurely transmitted, stored, or simply overshared

Missing Access Control Users can do things they shouldn't be allowed to do

XSRF Abuse the target website's trust in the browser

Vulnerable Components Third-party tools are vulnerable

Unvalidated Redirects Abusable open-ended forwarding

[source](#)

Injection	Server-side code execution
Broken Authentication	Allows for impersonation
XSS	Client-side code execution
Direct References	Access control can be circumvented
Security Misconfiguration	Vulnerable default/inherited settings
Data Exposure	Data is insecurely transmitted, stored, or simply overshared
Missing Access Control	Users can do things they shouldn't be allowed to do
XSRF	Abuse the target website's trust in the browser
Vulnerable Components	Third-party tools are vulnerable
Unvalidated Redirects	Abusable open-ended forwarding

[source](#)

	Exploitability	Prevalence	Detectability	Impact
Injection	easy	common	average	severe
Broken Authentication	average	widespread	average	severe
XSS	average	very widespread	easy	moderate
Direct References	easy	common	easy	moderate
Security Misconfiguration	easy	common	easy	moderate
Data Exposure	difficult	uncommon	average	severe
Missing Access Control	easy	common	average	moderate
XSRF	average	common	easy	moderate
Vulnerable Components	average	widespread	difficult	moderate
Unvalidated Redirects	average	uncommon	easy	moderate

[source](#)

	Exploitability	Prevalence	Detectability	Impact
Injection	easy	common	average	severe
Broken Authentication	average	widespread	average	severe
XSS	average	very widespread	easy	moderate
Direct References	easy	common	easy	moderate
Security Misconfiguration	easy	common	easy	moderate
Data Exposure	difficult	uncommon	average	severe
Missing Access Control	easy	common	average	moderate
XSRF	average	common	easy	moderate
Vulnerable Components	average	widespread	difficult	moderate
Unvalidated Redirects	average	uncommon	easy	moderate

[source](#)

Injection	Server-side code execution	example
Broken Authentication	Allows for impersonation	example
XSS	Client-side code execution	example
Direct References	Access control can be circumvented	example
Security Misconfiguration	Vulnerable default/inherited settings	example
Data Exposure	Data is insecurely transmitted, stored, or simply overshared	example
Missing Access Control	Users can do things they shouldn't be allowed to do	
XSRF	Abuse the target website's trust in the browser	example
Vulnerable Components	Third-party tools are vulnerable	example
Unvalidated Redirects	Abusable open-ended forwarding	example

Lesson: do not execute arbitrary code on the server.

<u>Injection</u>	Server-side code execution	example
Broken Authentication	Allows for impersonation	example
XSS	Client-side code execution	example
Direct References	Access control can be circumvented	example
Security Misconfiguration	Vulnerable default/inherited settings	example
Data Exposure	Data is insecurely transmitted, stored, or simply overshared	example
Missing Access Control	Users can do things they shouldn't be allowed to do	
XSRF	Abuse the target website's trust in the browser	example
Vulnerable Components	Third-party tools are vulnerable	example
Unvalidated Redirects	Abusable open-ended forwarding	example

Lesson: verify credentials, use HTTPS, hash and salt passwords.

Injection	Server-side code execution	example
<u>Broken Authentication</u>	Allows for impersonation	example
XSS	Client-side code execution	example
Direct References	Access control can be circumvented	example
Security Misconfiguration	Vulnerable default/inherited settings	example
Data Exposure	Data is insecurely transmitted, stored, or simply overshared	example
Missing Access Control	Users can do things they shouldn't be allowed to do	
XSRF	Abuse the target website's trust in the browser	example
Vulnerable Components	Third-party tools are vulnerable	example
Unvalidated Redirects	Abusable open-ended forwarding	example

Lesson: do not execute arbitrary code on the client.

Injection	Server-side code execution	example
Broken Authentication	Allows for impersonation	example
<u>XSS</u>	Client-side code execution	example
Direct References	Access control can be circumvented	example
Security Misconfiguration	Vulnerable default/inherited settings	example
Data Exposure	Data is insecurely transmitted, stored, or simply overshared	example
Missing Access Control	Users can do things they shouldn't be allowed to do	
XSRF	Abuse the target website's trust in the browser	example
Vulnerable Components	Third-party tools are vulnerable	example
Unvalidated Redirects	Abusable open-ended forwarding	example

Lesson: clients should only be able manipulate resources indirectly, through the server.

Injection	Server-side code execution	example
Broken Authentication	Allows for impersonation	example
XSS	Client-side code execution	example
<u>Direct References</u>	Access control can be circumvented	example
Security Misconfiguration	Vulnerable default/inherited settings	example
Data Exposure	Data is insecurely transmitted, stored, or simply overshared	example
Missing Access Control	Users can do things they shouldn't be allowed to do	
XSRF	Abuse the target website's trust in the browser	example
Vulnerable Components	Third-party tools are vulnerable	example
Unvalidated Redirects	Abusable open-ended forwarding	example

Lesson: be wary of default security configurations.

Injection	Server-side code execution	example
Broken Authentication	Allows for impersonation	example
XSS	Client-side code execution	example
Direct References	Access control can be circumvented	example
<u>Security Misconfiguration</u>	Vulnerable default/inherited settings	example
Data Exposure	Data is insecurely transmitted, stored, or simply overshared	example
Missing Access Control	Users can do things they shouldn't be allowed to do	
XSRF	Abuse the target website's trust in the browser	example
Vulnerable Components	Third-party tools are vulnerable	example
Unvalidated Redirects	Abusable open-ended forwarding	example

Lesson: encrypt data in transit and storage; share only proper fields.

Injection	Server-side code execution	example
Broken Authentication	Allows for impersonation	example
XSS	Client-side code execution	example
Direct References	Access control can be circumvented	example
Security Misconfiguration	Vulnerable default/inherited settings	example
<u>Data Exposure</u>	Data is insecurely transmitted, stored, or simply overshared	example
Missing Access Control	Users can do things they shouldn't be allowed to do	
XSRF	Abuse the target website's trust in the browser	example
Vulnerable Components	Third-party tools are vulnerable	example
Unvalidated Redirects	Abusable open-ended forwarding	example

Lesson: appropriately restrict certain users to certain actions on certain resources.

Injection	Server-side code execution	example
Broken Authentication	Allows for impersonation	example
XSS	Client-side code execution	example
Direct References	Access control can be circumvented	example
Security Misconfiguration	Vulnerable default/inherited settings	example
Data Exposure	Data is insecurely transmitted, stored, or simply overshared	example
<u>Missing Access Control</u>	Users can do things they shouldn't be allowed to do	
XSRF	Abuse the target website's trust in the browser	example
Vulnerable Components	Third-party tools are vulnerable	example
Unvalidated Redirects	Abusable open-ended forwarding	example

Lesson: do not only use cookies for session authentication.

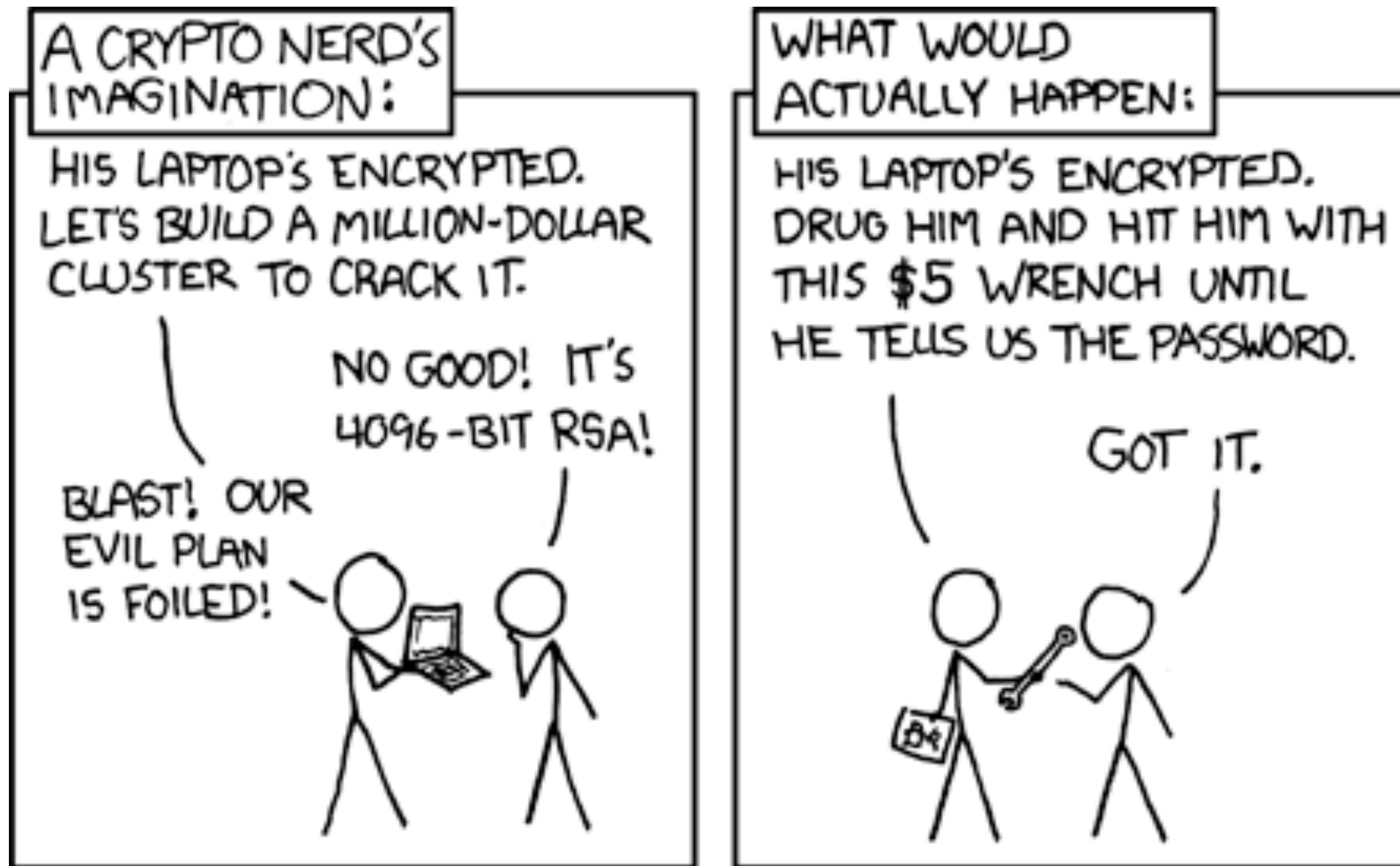
Injection	Server-side code execution	example
Broken Authentication	Allows for impersonation	example
XSS	Client-side code execution	example
Direct References	Access control can be circumvented	example
Security Misconfiguration	Vulnerable default/inherited settings	example
Data Exposure	Data is insecurely transmitted, stored, or simply overshared	example
Missing Access Control	Users can do things they shouldn't be allowed to do	
<u>XSRF</u>	<u>Abuse the target website's trust in the browser</u>	example
Vulnerable Components	Third-party tools are vulnerable	example
Unvalidated Redirects	Abusable open-ended forwarding	example

Lesson: carefully consider the libraries you use.

Injection	Server-side code execution	example
Broken Authentication	Allows for impersonation	example
XSS	Client-side code execution	example
Direct References	Access control can be circumvented	example
Security Misconfiguration	Vulnerable default/inherited settings	example
Data Exposure	Data is insecurely transmitted, stored, or simply overshared	example
Missing Access Control	Users can do things they shouldn't be allowed to do	
XSRF	Abuse the target website's trust in the browser	example
<u>Vulnerable Components</u>	Third-party tools are vulnerable	example
Unvalidated Redirects	Abusable open-ended forwarding	example

Lesson: forwards should be defined by the developers, not the users.

Injection	Server-side code execution	example
Broken Authentication	Allows for impersonation	example
XSS	Client-side code execution	example
Direct References	Access control can be circumvented	example
Security Misconfiguration	Vulnerable default/inherited settings	example
Data Exposure	Data is insecurely transmitted, stored, or simply overshared	example
Missing Access Control	Users can do things they shouldn't be allowed to do	
XSRF	Abuse the target website's trust in the browser	example
Vulnerable Components	Third-party tools are vulnerable	example
<u>Unvalidated Redirects</u>	Abusable open-ended forwarding	example



[HTTPS://XKCD.COM/538/](https://xkcd.com/538/)