Web-based Attacks on Host-Proof Encrypted Storage

A. Delignat-Lavaud  K. Bhargavan

PROSECCO, INRIA Paris-Rocquencourt

WOOT’12
August 7, 2012
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction

Host-Proof Application Design

Web Patterns

Exploiting the weak points

Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications

Host-Proof Application Design

User

Content Server

Sensitive data

Client Application

Hacker

Sensitive data

Client Application
Host-Proof Application Design

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications

Web-based Attacks on Host-Proof Encrypted Storage
Delignat-Lavaud, Bhargavan

Diagram:
- Content Server
- Client Application
- User
- Hacker
- Sensitive data

Flow:
- User accesses Content Server
- Hacker attempts unauthorized access to Sensitive data
- Encryption and key management to protect data
- Secure application design principles
Host-Proof Application

A foolproof design?
Host-Proof Web Application

Uh-Oh.

Introduction

Host-Proof Application Design

Web Patterns

Exploiting the weak points

Encryption

Authorized release of plaintext

Usual range of web attacks

Key management

Towards secure host-proof web applications
Host-Proof Pattern: Cloud Storage

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications

Host-Proof Pattern: Cloud Storage

- Server
- User
- App Website
- Hacker

Friends?
sharing

authentication
encrypted data
decryption script
decryption

key

Decrypted Data

CSRF
XSS

Hacker
Host-Proof Pattern: Cloud Storage

- **User**
- **Server**
- **App Website**
- **Hacker**

**Connections:**
- From User to App Website: User provides credentials (authentication).
- From App Website to Decrypted Data: Decryption of encrypted data.
- From Decrypted Data to Hacker: Decrypted data may contain malicious scripts.
- From Hacker to User: XSS attack.
- From Hacker to App Website: CSRF attack.

**Key Points:**
- **Encryption**
- **Authorized release of plaintext**
- **Usual range of web attacks**
- **Key management**

**Towards secure host-proof web applications**
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications

Host-Proof Patterns: Password Manager

Server

Extension

User

Host Page

session

URL

password
Host-Proof Patterns: Password Manager

Server

Extension

User

Host Page

session

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Conclusion

Towards secure host-proof web applications

Exploiting the weak points
- Encryption
- Authorized release of plaintext
- Usual range of web attacks
- Key management

Host-Proof Patterns: Password Manager

Server

Extension

User

Host Page

session

URL

password
Host-Proof Patterns: Password Manager

Server

Extension

User

Host Page

session

URL

password

Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications
Host-Proof Pattern: Roaming Password Manager

Introduction

Host-Proof Application Design

Web Patterns

Exploiting the weak points

Encryption

Authorized release of plaintext

Usual range of web attacks

Key management

Towards secure host-proof web applications
Introduction

Host-Proof Application Design

Web Patterns

Exploiting the weak points

Encryption

Authorized release of plaintext

Usual range of web attacks

Key management

Towards secure host-proof web applications
Host-Proof Pattern: Roaming Password Manager

- **Server**
  - Secret

- **Bookmarklet**
  - Session
  - Intention
  - URL
  - Password

- **User**
  - Session
  - Intention

- **App Website**

- **Host Page**

Introduction

Host-Proof Application Design

Web Patterns

Exploiting the weak points
- Encryption
- Authorized release of plaintext
- Usual range of web attacks
- Key management

Towards secure host-proof web applications
Host-Proof Pattern: Roaming Password Manager

Introduction

Host-Proof Application Design

Web Patterns

Exploiting the weak points

Encryption

Authorized release of plaintext

Usual range of web attacks

Key management

Towards secure host-proof web applications
Host-Proof Pattern: Roaming Password Manager

Introduction

Host-Proof Application Design

Web Patterns

Exploiting the weak points

Encryption

Authorized release of plaintext

Usual range of web attacks

Key management

Towards secure host-proof web applications
Host-Proof Pattern: Roaming Password Manager

Server -- secret -- Bookmarklet

-- session --

User

-- intention --

App Website

-- URL --

Host Page

-- password --

[Diagram showing the relationships between the server, bookmarklet, user, app website, and host page.]
Host-Proof Pattern: Roaming Password Manager

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications
Exploiting the weak points

What can go wrong?

- Encryption and ciphertext sharing.
- Authorized release of plaintext.
- Key management.
- Programming errors.
Exploiting the weak points

What can go wrong?

- Encryption and ciphertext sharing.
- Authorized release of plaintext.
- Key management.
- Programming errors.
Exploiting the weak points

What can go wrong?

- Encryption and ciphertext sharing.
- Authorized release of plaintext.
- Key management.
- Programming errors.
Exploiting the weak points

What can go wrong?

- Encryption and ciphertext sharing.
- Authorized release of plaintext.
- Key management.
- Programming errors.
## Strong crypto primitives

<table>
<thead>
<tr>
<th>Name</th>
<th>Key Derivation</th>
<th>Encryption</th>
<th>Integrity</th>
<th>Crypt Meta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wuala</td>
<td>PBKDF2-SHA256</td>
<td>AES, RSA</td>
<td>HMAC</td>
<td>✔</td>
</tr>
<tr>
<td>SpiderOak</td>
<td>PBKDF2-SHA256</td>
<td>AES, RSA</td>
<td>HMAC</td>
<td>✔</td>
</tr>
<tr>
<td>BoxCryptor</td>
<td>PBKDF2</td>
<td>AES</td>
<td>None</td>
<td>✗</td>
</tr>
<tr>
<td>CloudFogger</td>
<td>PBKDF2</td>
<td>AES, RSA</td>
<td>None</td>
<td>✗</td>
</tr>
<tr>
<td>LastPass</td>
<td>PBKDF2-SHA256</td>
<td>AES, RSA</td>
<td>None</td>
<td>✗</td>
</tr>
<tr>
<td>PassPack</td>
<td>SHA256</td>
<td>AES</td>
<td>Records</td>
<td>✔</td>
</tr>
<tr>
<td>RoboForm</td>
<td>PBKDF2</td>
<td>AES, DES</td>
<td>None</td>
<td>✗</td>
</tr>
<tr>
<td>1Password</td>
<td>PBKDF2-SHA1</td>
<td>AES</td>
<td>None</td>
<td>✗</td>
</tr>
<tr>
<td>Clipperz</td>
<td>SHA256</td>
<td>AES</td>
<td>SHA-256</td>
<td>✔</td>
</tr>
</tbody>
</table>
Key derivation

Andrey Belenko and Dmitry Sklyarov
"Secure Password Managers" and "Military-Grade Encryption" on Smartphones: Oh, Really?
Elcomsoft Technical Report
Incorrect use of crypto

Old wisdom from Steven Bellovin
Without integrity protection, encryption is all but useless.
No ciphertext integrity protection

**RoboForm Passcard**

URL3: Encode(URL)  
+PROTECTED-2+  
<ENC_k(username, password)>

**1Password Keychain**

{"uuid":..., "title":..., "location":URL,  
"encrypted":<ENC_k(username, password)>}
No ciphertext integrity protection

Content Server

Friend

google.com
ENC(u, p)

User

Client Application

p

Hacker

Google.com

User

google.com
ENC(u, p)

Content Server

User

google.com

Hacker

p

Content Server

Friend

google.com
ENC(u, p)
No ciphertext integrity protection

Content Server

bad.com
ENC(u, p)

Friend

google.com
ENC(u, p)

User

Client Application

p

Share

Hacker

bad.com

p

Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications
No ciphertext integrity protection
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption

Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications

Classic problem: URL authenticating

- Browser extension-based password managers;
- Match URL with password database in JS.
- Error-prone RegExp matching.

parseUri pattern

/^(?:(?:([^-\/:/?#]+):)?(?:\:\/)\/(?:((?:([^-[@\:]*)\)?(?::([^-[@\:]*)?))?\@)?(\[^:/?#]*)\?((?:[\d*]))\?))?)?((?:[^?#\/:]*)\?([^#]*))?\#(.*))?/"/

Incorrect

http://bad.com/#@accounts.google.com
Authorized release of plaintext

Classic problem: URL authenticating

- Browser extension-based password managers;
- Match URL with password database in JS.
- Error-prone RegExp matching.

```
parseUri pattern

/^(?:([^:/?#]+):)?(?://((?:(([^:@\]*)(?::[^:@\]*))?@)?([^:/?#]*)(?::([^d]*))?))?(((?:[^?#/]*/)*)([^?#]*)?)?(?:([^#]*))?[#:\[\]^\$\&\*^\?\(\)\|\(\)^\(\)\]+)?(?:\?([^#]*))?(?:#(.*))?/"'
```

Incorrect

http://bad.com/#@accounts.google.com
Authorized release of plaintext

Classic problem: URL authenticating

- Browser extension-based password managers;
- Match URL with password database in JS.
- Error-prone RegExp matching.

parseUri pattern

```
/^(<?:(\^[^:]\)?[\^/:\?\#\]*):)?(?:\/:)?(\^[\^/:\?\#\]*)(?:\/(\^[\^/:\?\#\]*))?(?:\?([^#\]*))?(?:#(.*))?$/
```

Incorrect

http://bad.com/#@accounts.google.com
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption

Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications

Classic problem: URL authenticating

- Browser extension-based password managers;
- Match URL with password database in JS.
- Error-prone RegExp matching.

Authorized release of plaintext

Classic problem: URL authenticating

- Browser extension-based password managers;
- Match URL with password database in JS.
- Error-prone RegExp matching.

parseUri pattern

```
/^(?:([^:/?#]+):)?(?://((?:(([^:@\*]+)(?::([^:@\*]+))?)?@)?([^:/?#]*)(?::([^d*]))?)?)?((((?:[^?#\/+]*/)*)([^?#]*))?([^#]*))?([^#]*))/
```

Incorrect

http://bad.com/#@accounts.google.com
 Authorized release of plaintext

Classic problem: URL authenticating

- Browser extension-based password managers;
- Match URL with password database in JS.
- Error-prone RegExp matching.

parseUri pattern

```
/^(?:([^:/?#]+):)?(?://((?:(([^:@\]*)(?::([^:@\]*))?@)?([^:/?#]*)(?::([^d*])*))?)?(/(?:[^?#/\]*))(?:\?([^#]*))?#(.*))?)/
```

Incorrect

http://bad.com/#@accounts.google.com
Phishing attack on 1Password extension

URL parsing code

```javascript
var href = getBrowser().contentWindow.location.href + "/";
var domain = href.replace("/^http[s]*:\/\/(.*)\/.*$"i, "$1");
var middle = domain.replace("/^((www.)*(.*)$/i, "$2");

return middle.substring(0,1).toUpperCase() + middle.substring(1,middle.length);
```

Phishing URL

http://www.google.com:xxx@bad.com
Phishing attack on 1Password extension

URL parsing code

```javascript
var href = getBrowser().contentWindow.location.href + "/";
var domain = href.replace(
    /^http[s]*:\/\/(.*)\/.*$/, "$1");
var middle = domain.replace(
    /(www.)*(.*)/i, "$2");

return middle.substring(0,1).toUpperCase() +
middle.substring(1,middle.length);
```

Phishing URL

http://www.google.com:xxx@bad.com
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications

Gmail
A Google approach to email.

Gmail is built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:

- **Lots of space**
  Over 10276.513376 megabytes (and counting) of free storage.

- **Loss spam**
  Keep unwanted messages out of your inbox.

- **Mobile access**
  Get Gmail on your mobile phone. Learn more
1Password phishing attack

Server → 1Password → User

session

Phishing URL

Google

password

Attacker

1Password phishing attack

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications

1Password phishing attack

Server

1Password

session

User

Phishing URL

Google

password

Attacker

Server

1Password

session

User

Phishing URL

Google

password

Attacker
1Password phishing attack

- Server
- 1Password
- User
- Attacker
- Google

Phishing URL

session

Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications
Usual range of web attacks

Web interfaces

- Hard to maintain client-side decryption due to JavaScript limitations.
- Login form exposed to web attacks.
- Decryption in same scope as GUI and user data.
Usual range of web attacks

Web interfaces

- Hard to maintain client-side decryption due to JavaScript limitations.
- Login form exposed to web attacks.
- Decryption in same scope as GUI and user data.
Usual range of web attacks

Web interfaces

- Hard to maintain client-side decryption due to JavaScript limitations.
- Login form exposed to web attacks.
- Decryption in same scope as GUI and user data.
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications

SpiderOak

User

Server

SpiderOak

Attacker

JSONP query

JSON listing

session

JSON listing
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications
Query

https://spideroak.com/storage/<u32>/?callback=f

Result

f(
    "stats": {
        "firstname": "...",
        "lastname": "...",
        "devices": ...
    },
    "devices": [
        ["pc1", "pc1/"]
    ],
    ["laptop", "laptop/"],...
);
Query

https://spideroak.com/storage/<u32>/shares

Result

{
  "share_rooms" : [ 
    "url" : "/browse/share/<id>/<key>",
    "room_key" : "<key>",
    "room_description" : "",
    "room_name": "<room>"
  ],
  "share_id" : "<id>",
  "share_id_b32" : "<u32>"
}
Key management

A difficult challenge

- All applications implement some form of sharing.
  - Key policy: reencrypt? Key hierarchy? Share keys? Share plaintexts?
  - Design errors virtually impossible to fix.

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications
Key management

A difficult challenge

- All applications implement some form of sharing.
- Key policy: reencrypt? Key hierarchy? Share keys? Share plaintexts?
- Design errors virtually impossible to fix.
Key management

<table>
<thead>
<tr>
<th>A difficult challenge</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ All applications implement some form of sharing.</td>
</tr>
<tr>
<td>▶ Key policy: reencrypt? Key hierarchy? Share keys? Share plaintexts?</td>
</tr>
<tr>
<td>▶ Design errors virtually impossible to fix.</td>
</tr>
</tbody>
</table>
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications

LastPass login bookmarklet

Server

Bookmarklet

User

App Website

Attacker

s

D, Enc_{s,r}(K), r

session

intention

rootkit

K

∥
LastPass login bookmarklet

- **Server**
- **User**
- **App Website**
- **Bookmarklet**
- **Attacker**

Roles and interactions:
- User sends session information to Bookmarklet.
- Bookmarklet decrypts EncS_r(K), r = D, EncS_r(K).
- User approves or disapproves the session.

Key management and encryption:
- K is the key for encryption and decryption.
- EncS_r(K) represents the encrypted data.
- Decryption occurs at the Bookmarklet.

Web-based Attacks on Host-Proof Encrypted Storage

**Introduction**
- Host-Proof Application Design
- Web Patterns

**Exploiting the weak points**
- Encryption
- Authorized release of plaintext
- Usual range of web attacks

**Towards secure host-proof web applications**
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns

Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management

Towards secure host-proof web applications

LastPass login bookmarklet

- Server
- Bookmarklet
- User
- App Website
- Attacker

D, Encs,r(K), r

s

session

intention

K

rootkit

K

24 / 30
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications

LastPass login bookmarklet

Server

bookmarklet

Attacker

App Website

User

K

Encs,r(K), r

D

s

K

session

intention

rootkit

session
Web-based Attacks on Host-Proof Encrypted Storage

Delignat-Lavaud, Bhargavan

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications
Key recovery by rootkiting

function _LP_START() {
    _LP = new _LP_CONTAINER();
    var d = {{encrypted form data}};
    _LP.setVars(d, '<user>',
               '<encrypted_key>', _LASTPASS_RAND, ...);
    _LP.bmMulti(null, null);
}

Ben Adida, Adam Barth and Collin Jackson
Rootkits for JavaScript environments
WOOT’2009
Towards secure host-proof web applications

Is secure crypto possible in browsers?

Matasano plea: “JavaScript Cryptography Considered Harmful”:

- Secure delivery of JavaScript to browsers is a chicken-egg problem.
- Browser Javascript is hostile to cryptography.
- The “view-source” transparency of JavaScript is illusory.
Towards secure host-proof web applications

Is secure crypto possible in browsers?

Matasano plea: “JavaScript Cryptography Considered Harmful”:

- Secure delivery of JavaScript to browsers is a chicken-egg problem.
- Browser Javascript is hostile to cryptography.
- The “view-source” transparency of JavaScript is illusory.
Is secure crypto possible in browsers?

Matasano plea: “JavaScript Cryptography Considered Harmful”:

- Secure delivery of JavaScript to browsers is a chicken-egg problem.
- Browser Javascript is hostile to cryptography.
- The “view-source” transparency of JavaScript is illusory.
Available building blocks

**Server side**
- Origin HTTP header.
- Access-Control-Allow-Origin.
- Content Security Policy.

**Client side**
- Browser extension sandboxing.
- Web Crypto API.
- DOM Storage.
- HTML5 frame sandboxing.
- HTML5 Local Storage.

Introduction
Host-Proof Application Design
Web Patterns
Exploiting the weak points
Encryption
Authorized release of plaintext
Usual range of web attacks
Key management
Towards secure host-proof web applications
Available building blocks

Server side
- Origin HTTP header.
- Access-Control-Allow-Origin.
- Content Security Policy.

Client side
- Browser extension sandboxing.
- Web Crypto API.
- HTML5 frame sandboxing.
- HTML5 local storage.
Available building blocks

**Server side**
- Origin HTTP header.
- Access-Control-Allow-Origin.
- Content Security Policy.

**Client side**
- Browser extension sandboxing.
- Web Crypto API.
- ECMA 6
- HTML5 frame sandboxing.
- HTML5 local storage.
Available building blocks

Server side
- Origin HTTP header.
- Access-Control-Allow-Origin.
- Content Security Policy.

Client side
- Browser extension sandboxing.
  - Web Crypto API.
  - ECMA 6.
  - HTML5 frame sandboxing.
  - HTML5 local storage.
Available building blocks

**Server side**
- Origin HTTP header.
- Access-Control-Allow-Origin.
- Content Security Policy.

**Client side**
- Browser extension sandboxing.
- Web Crypto API.
  - ECMA 6
  - HTML5 frame sandboxing.
  - HTML5 local storage.
Available building blocks

### Server side
- Origin HTTP header.
- Access-Control-Allow-Origin.
- Content Security Policy.

### Client side
- Browser extension sandboxing.
- Web Crypto API.
- ECMA 6
  - HTML5 frame sandboxing.
  - HTML5 local storage.
Available building blocks

<table>
<thead>
<tr>
<th>Server side</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Origin HTTP header.</td>
</tr>
<tr>
<td>▶ Access-Control-Allow-Origin.</td>
</tr>
<tr>
<td>▶ Content Security Policy.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Client side</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Browser extension sandboxing.</td>
</tr>
<tr>
<td>▶ Web Crypto API.</td>
</tr>
<tr>
<td>▶ ECMA 6</td>
</tr>
<tr>
<td>▶ HTML5 frame sandboxing.</td>
</tr>
<tr>
<td>▶ HTML5 local storage.</td>
</tr>
</tbody>
</table>
Available building blocks

**Server side**
- Origin HTTP header.
- Access-Control-Allow-Origin.
- Content Security Policy.

**Client side**
- Browser extension sandboxing.
- Web Crypto API.
- ECMA 6
- HTML5 frame sandboxing.
- HTML5 local storage.
Connecting the dots

New automated security analysis tools

- Formal models of crypto in web apps: WebSpi / ProVerif.
- Formal models of JavaScript security: Alloy model, JS subsets.
- Information flow analyses: Jif (A. Myers), jsflow (D. Hedin, A. Sabelfeld), Zdancewic and Li.

Chetan Bansal, Karthikeyan Bhargavan and Sergio Maffeis

*Discovering Concrete Attacks on Website Authorization by Formal Analysis*

CSF 2012 (to appear)

Akhave, Barth, Lam, Mitchell, Song

*Towards a Formal Foundation of Web Security*

CSF 2010
Thank you.

... as well as ERC and USENIX Sponsors.