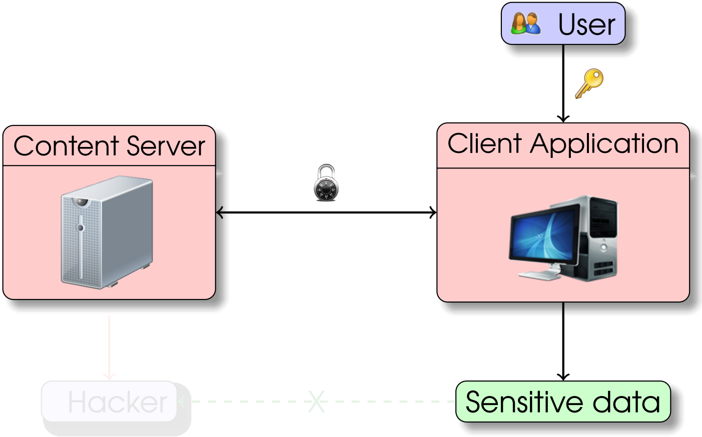
Web-based Attacks on Host-Proof Encrypted Storage

A. Delignat-Lavaud   K. Bhargavan

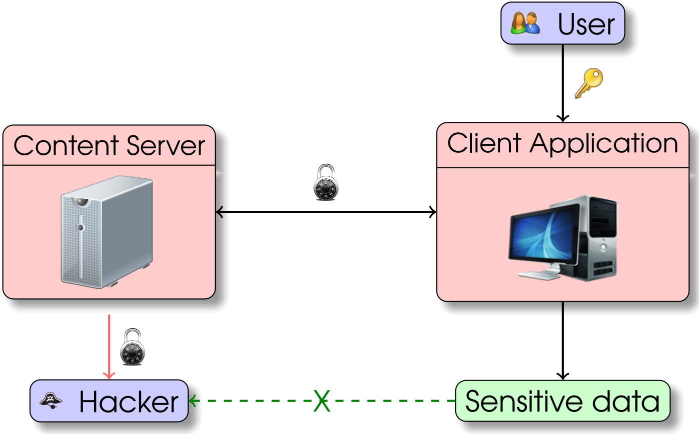
PROSECCO, INRIA Paris-Rocquencourt

WOOT'12
August 7, 2012

Host-Proof Application Design



Host-Proof Application Design



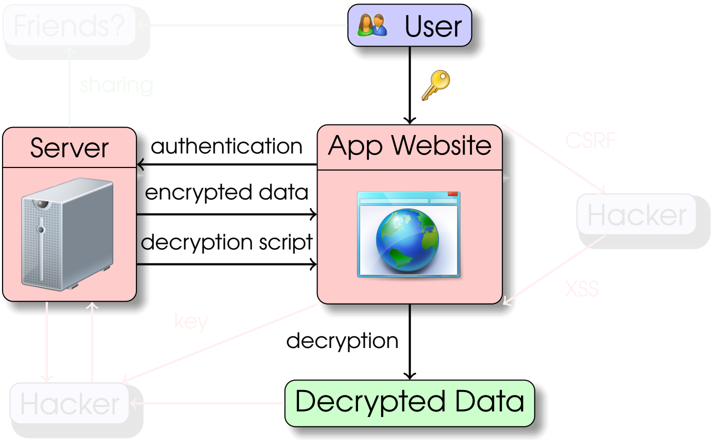
Host-Proof Application

A foolproof design?

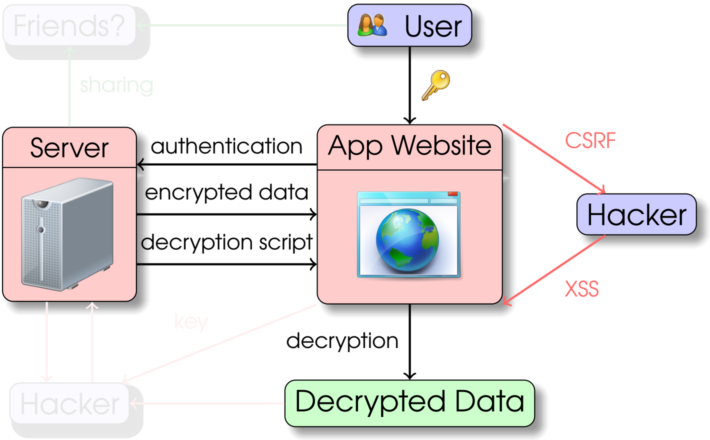
Host-Proof Web Application

Uh-Oh.

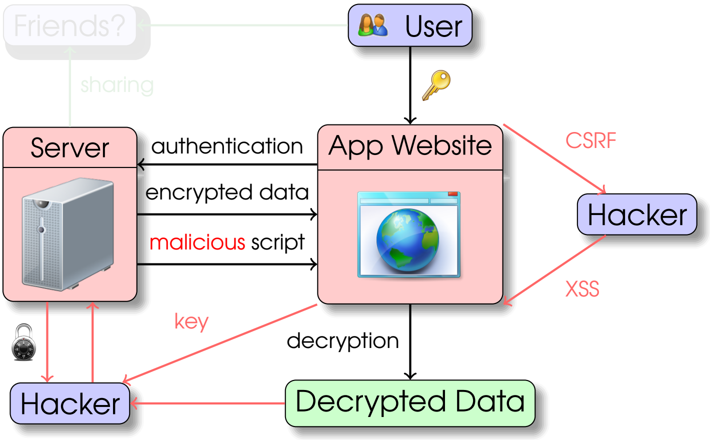
Host-Proof Pattern: Cloud Storage



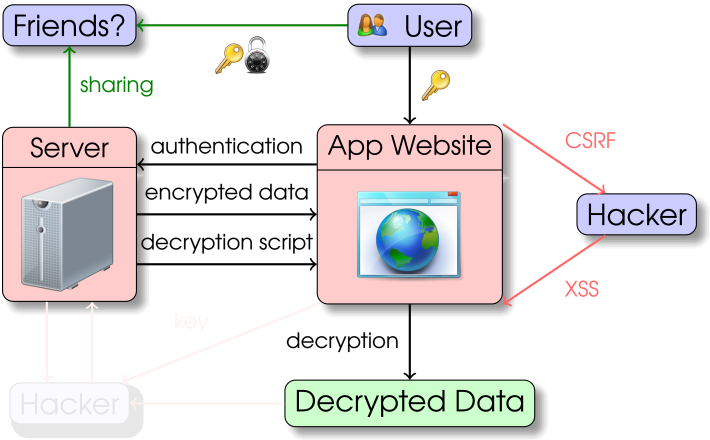
Host-Proof Pattern: Cloud Storage



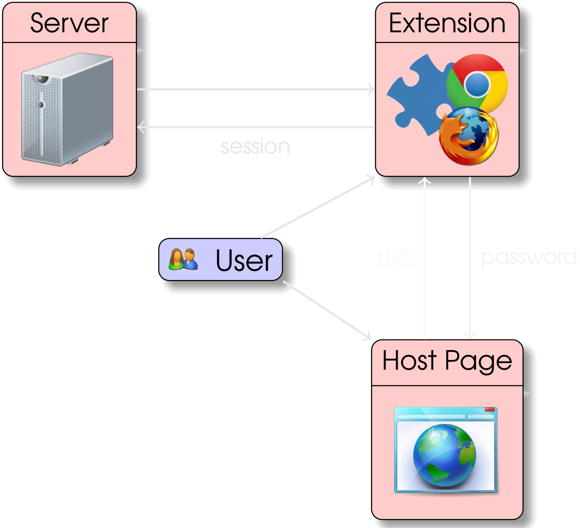
Host-Proof Pattern: Cloud Storage



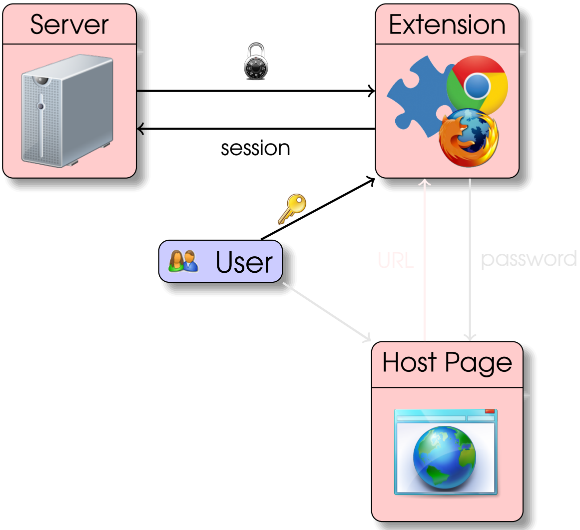
Host-Proof Pattern: Cloud Storage



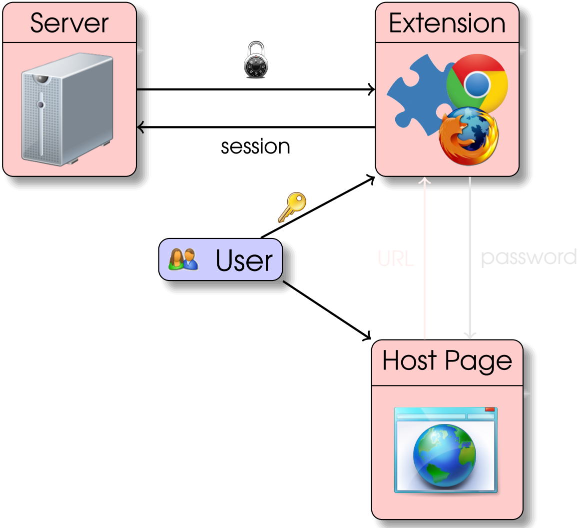
Host-Proof Patterns: Password Manager



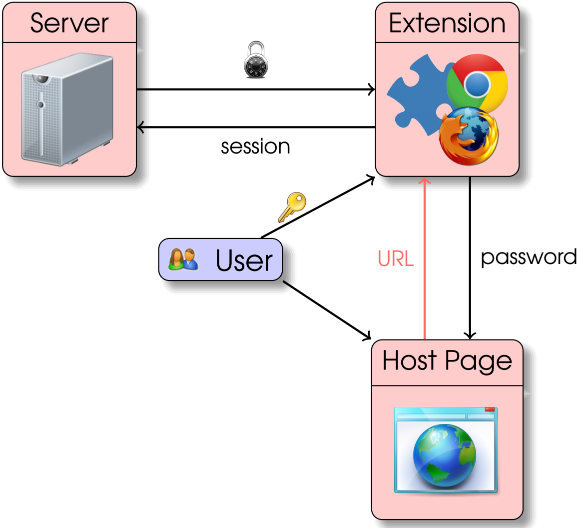
Host-Proof Patterns: Password Manager



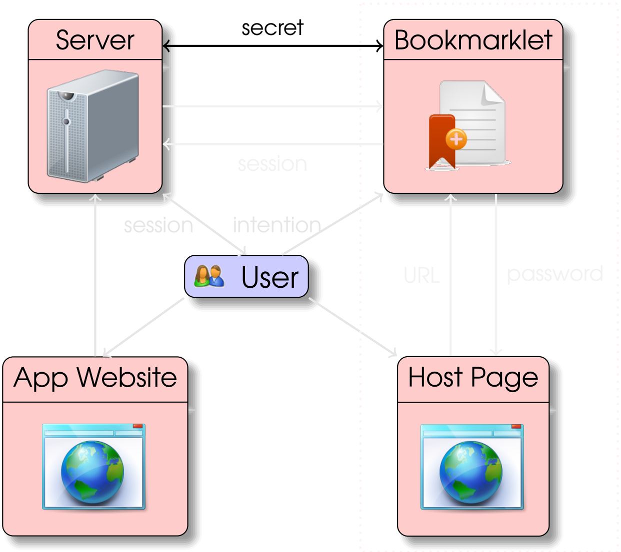
Host-Proof Patterns: Password Manager



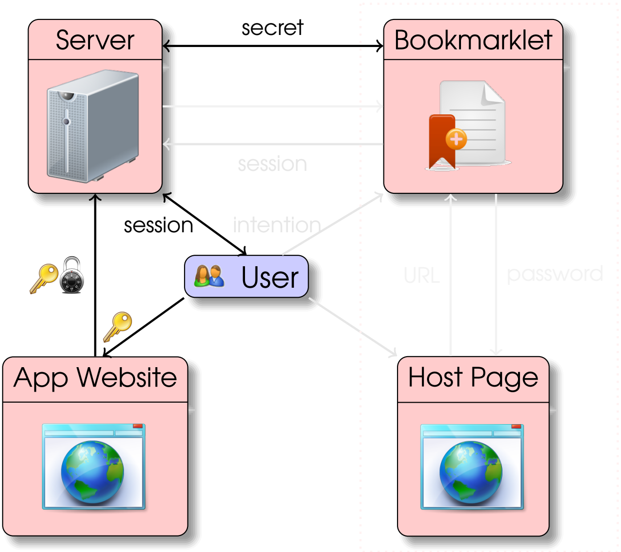
Host-Proof Patterns: Password Manager



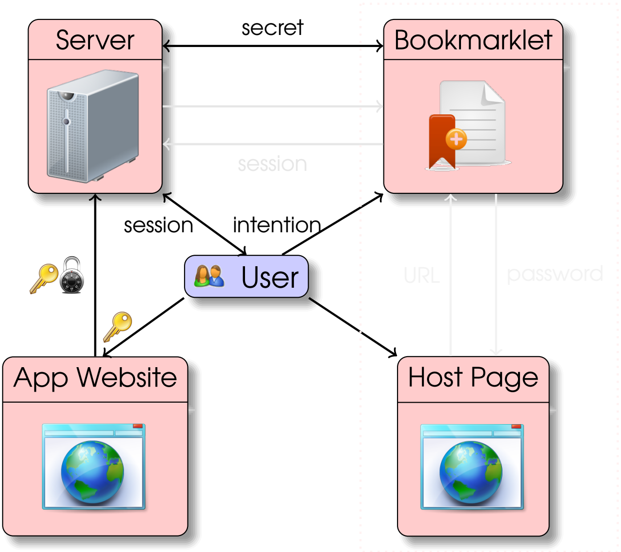
Host-Proof Pattern: Roaming Password Manager



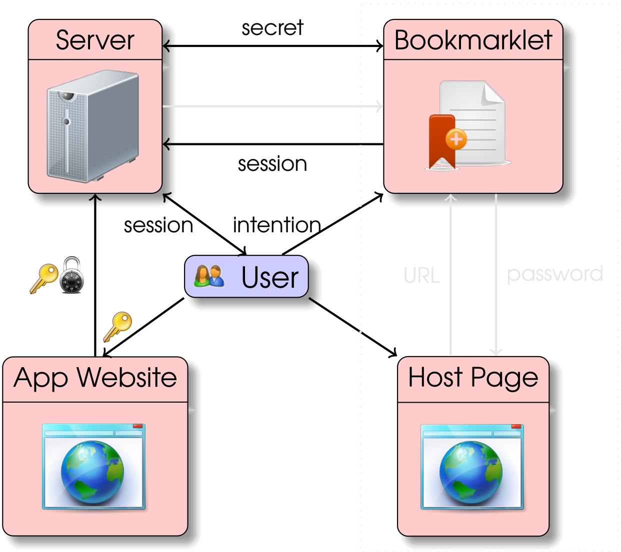
Host-Proof Pattern: Roaming Password Manager



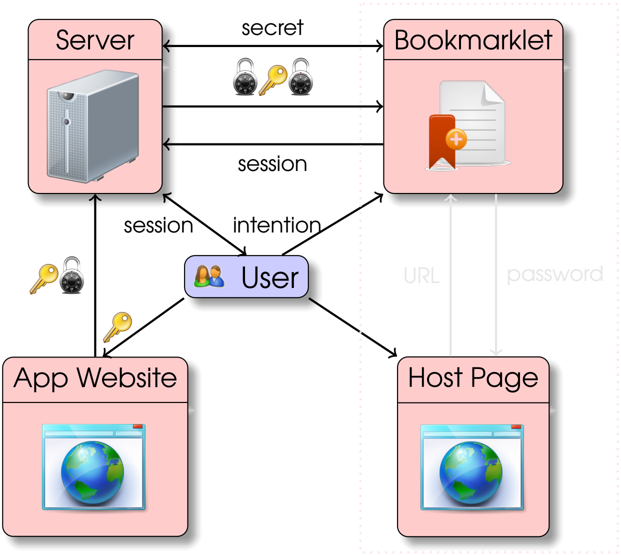
Host-Proof Pattern: Roaming Password Manager



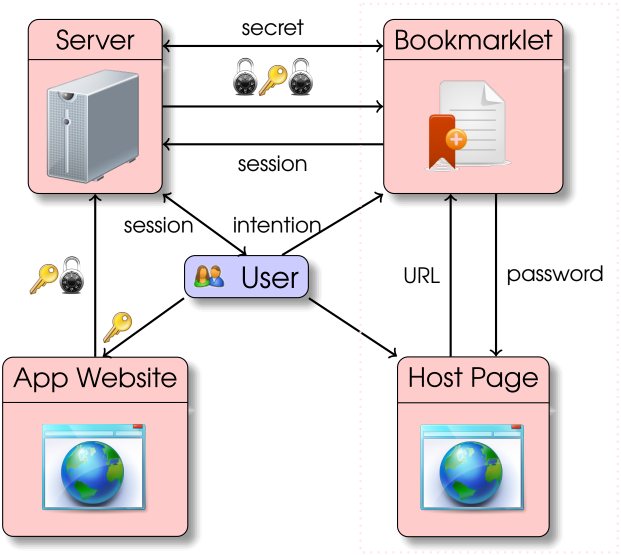
Host-Proof Pattern: Roaming Password Manager



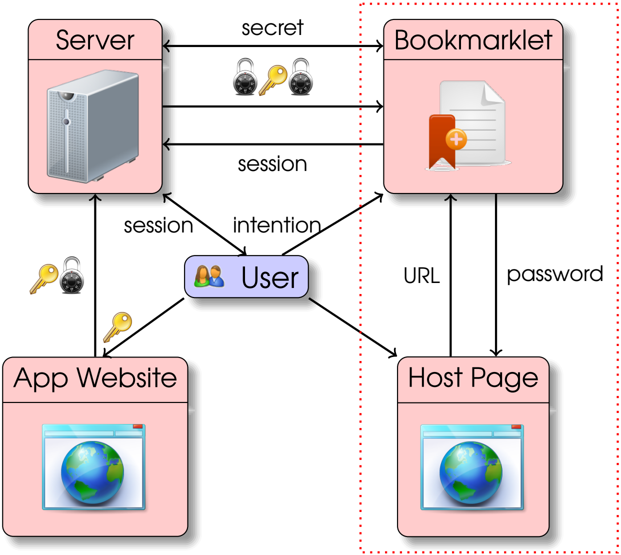
Host-Proof Pattern: Roaming Password Manager



Host-Proof Pattern: Roaming Password Manager



Host-Proof Pattern: Roaming Password Manager



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

Name	Key Derivation	Encryption	Integrity	Crypt Meta
Wuala	PBKDF2-SHA256	AES, RSA	HMAC	✓
SpiderOak	PBKDF2-SHA256	AES, RSA	HMAC	✓
BoxCryptor	PBKDF2	AES	None	✗
CloudFogger	PBKDF2	AES, RSA	None	✗
LastPass	PBKDF2-SHA256	AES, RSA	None	✗
PassPack	SHA256	AES	Records	✓
RoboForm	PBKDF2	AES, DES	None	✗
1Password	PBKDF2-SHA1	AES	None	✗
Clipperz	SHA256	AES	SHA-256	✓

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

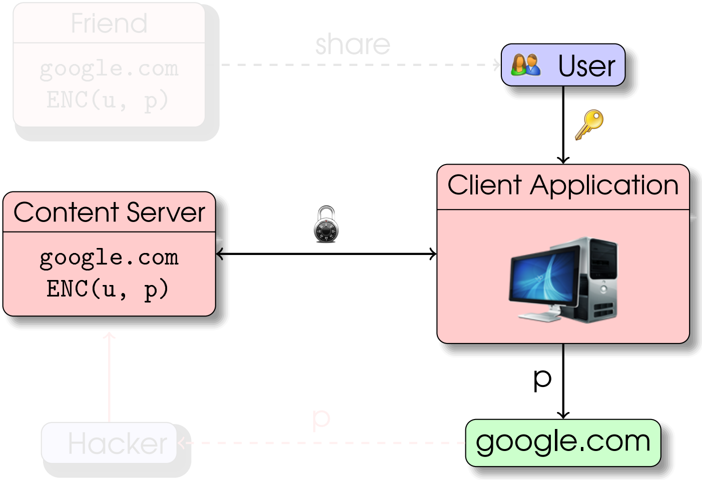
RoboForm Passcard

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

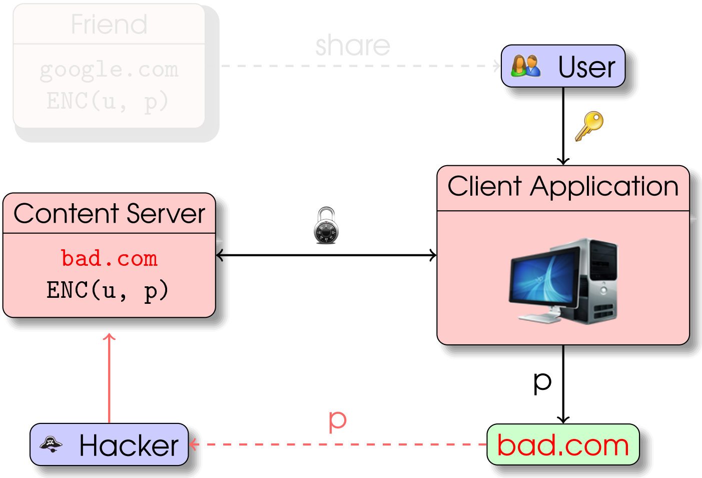
1Password Keychain

`{"uuid":...,"title":..., "location":URL,`
`"encrypted":<`ENC_k`(username,password)>}`

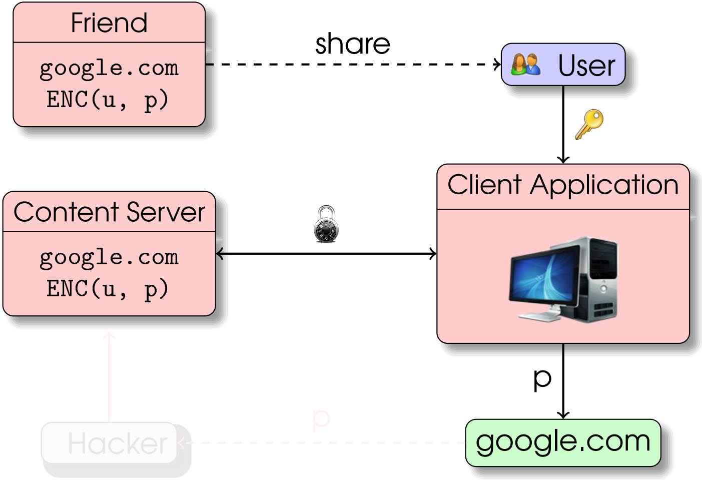
No ciphertext integrity protection



No ciphertext integrity protection



No ciphertext integrity protection



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
```

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`

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

```
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

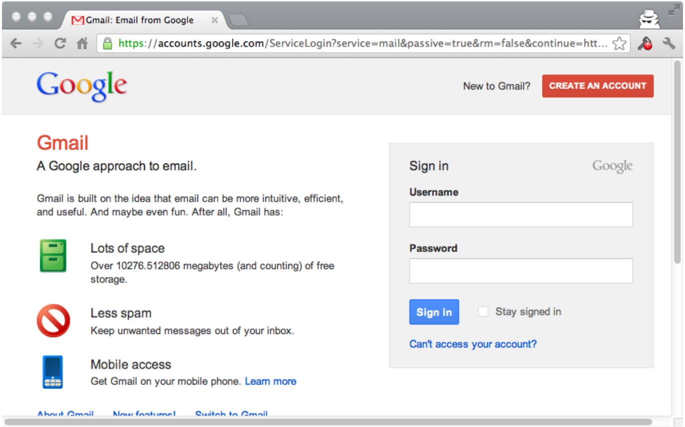
URL parsing code

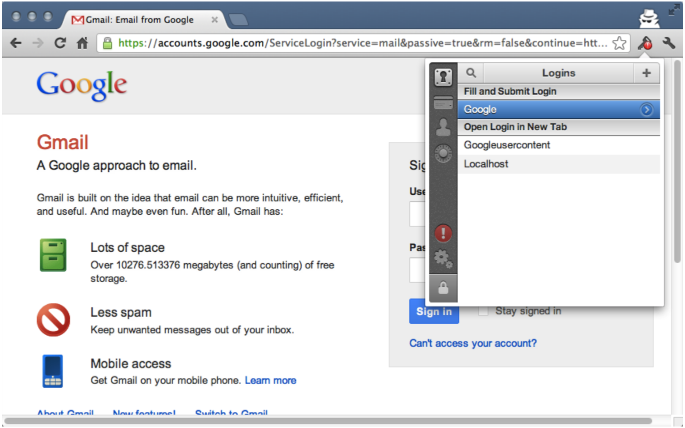
```
var href = getBrowser().contentWindow
                 .location.href + "/";
var domain = href.replace(
         /^http[s]*:\/\/(.*?)\/.*$/i, "$1");
var middle = domain.replace(
                    /^(www.)*(.*)/i, "$2");

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

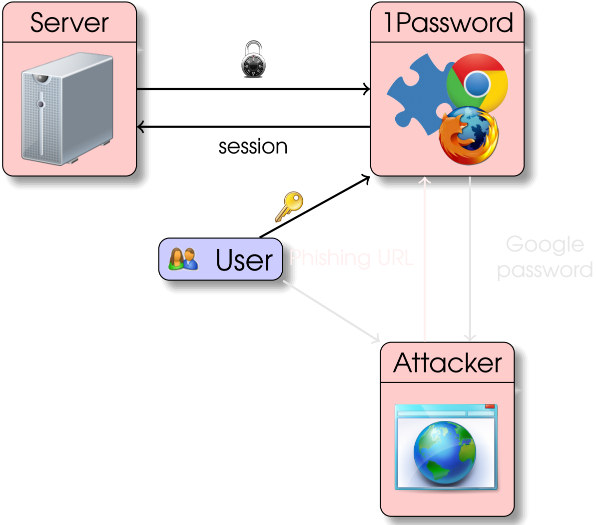
Phishing URL

```
http://www.google.com:xxx@bad.com
```

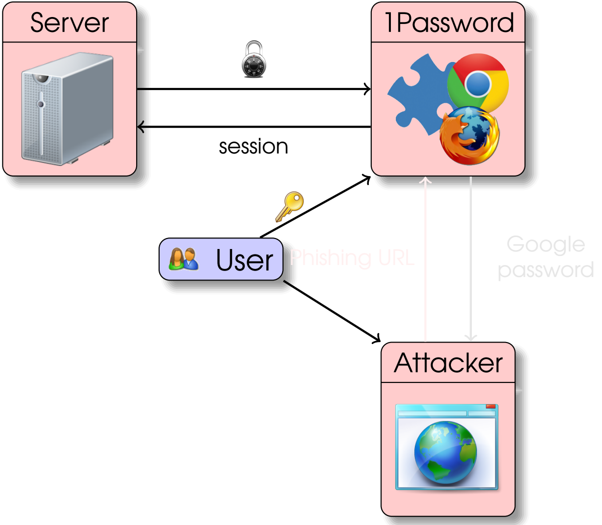

Gmail: Email from Google
https://accounts.google.com/ServiceLogin?service=mail&passive=true&rm=false&continue=htt...
Google
New to Gmail?
CREATE AN ACCOUNT
Gmail
A Google approach to email.
Gmail is built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:
Lots of space
Over 10276.512806 megabytes (and counting) of free storage.
Less spam
Keep unwanted messages out of your inbox.
Mobile access
Get Gmail on your mobile phone. Learn more
Sign in
Google
Username
Password
Sign in
Stay signed in
Can't access your account?


Gmail: Email from Google
https://accounts.google.com/ServiceLogin?service=mail&passive=true&rm=false&continue=htt...
Google
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.
Less spam
Keep unwanted messages out of your inbox.
Mobile access
Get Gmail on your mobile phone. Learn more
Logins
Fill and Submit Login
Google
Open Login in New Tab
Googleusercontent
Localhost
Sign in
Stay signed in
Can't access your account?

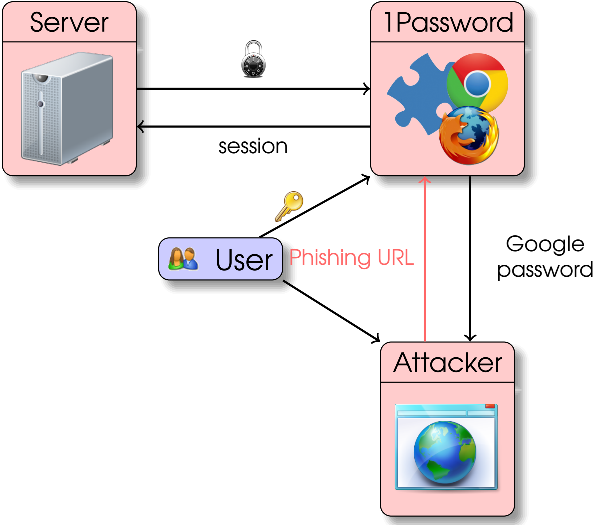
1Password phishing attack



1Password phishing attack



1Password phishing attack



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

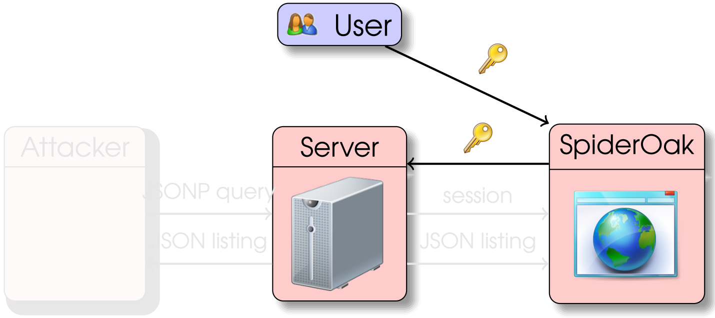
- Hard to maintain client-side decryption due to JavaScript limitations.
- Login form exposed to web attacks.
- Decryption in same scope as GUI and user data.

Usual range of web attacks

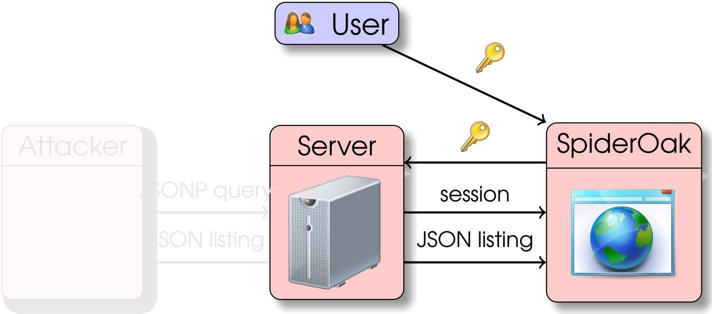
Web interfaces

- 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.

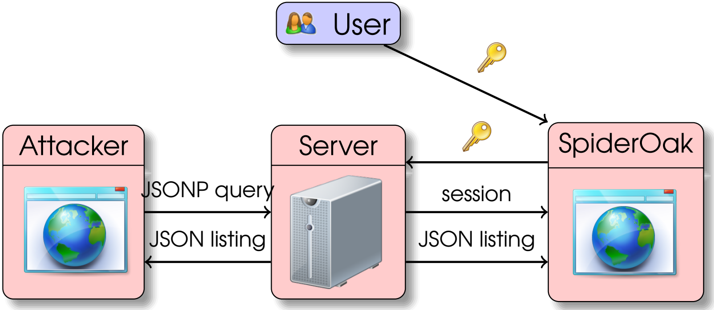
SpiderOak


User
Attacker
Server
SpiderOak
JSONP query
session
JSON listing
JSON listing

SpiderOak


User
Attacker
JSONP query
JSON listing
Server
session
JSON listing
SpiderOak

SpiderOak


User
Attacker
Server
SpiderOak
JSONP query
JSON listing
session
JSON listing

SpiderOak

Query

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

Result

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

SpiderOak

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.

Key management

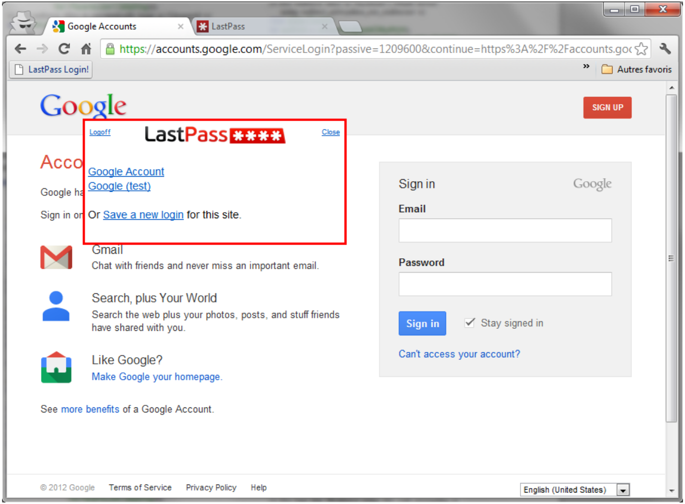
A difficult challenge

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

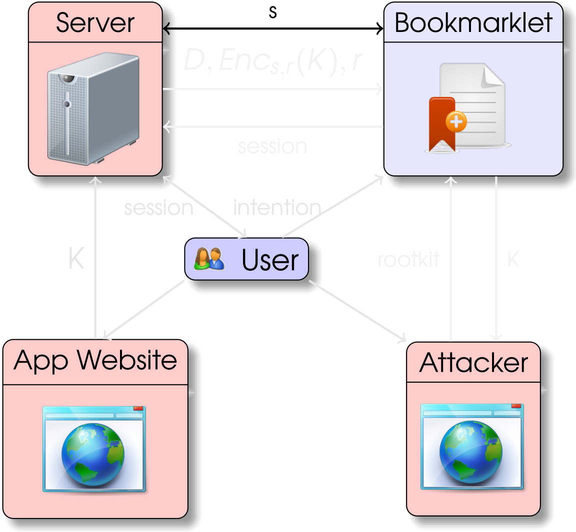
Key management

A difficult challenge

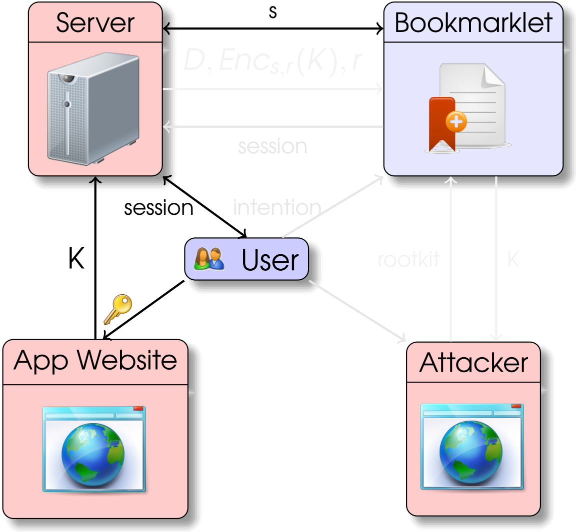
- All applications implement some form of sharing.
- Key policy: reencrypt? Key hierarchy? Share keys? Share plaintexts?
- Design errors virtually impossible to fix.


Google Accounts
LastPass
https://accounts.google.com/ServiceLogin?passive=1209600&continue=https%3A%2F%2Faccounts.goo
LastPass Login!
Autres favoris
Google
SIGN UP
Logoff
LastPass★★★★
Close
Google Account
Google (test)
Or Save a new login for this site.
Gmail
Chat with friends and never miss an important email.
Search, plus Your World
Search the web plus your photos, posts, and stuff friends have shared with you.
Like Google?
Make Google your homepage.
See more benefits of a Google Account.
Sign in
Email
Password
Sign in
Stay signed in
Can't access your account?
© 2012 Google
Terms of Service
Privacy Policy
Help
English (United States)

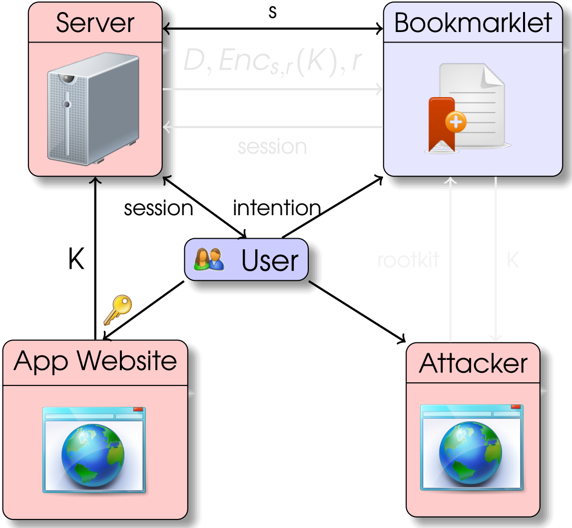
LastPass login bookmarklet



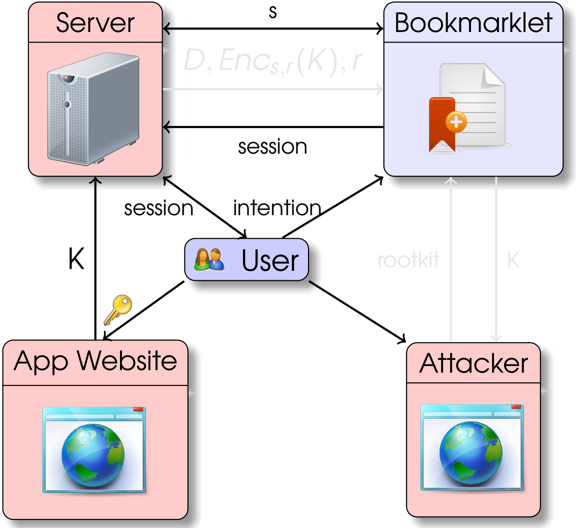
LastPass login bookmarklet



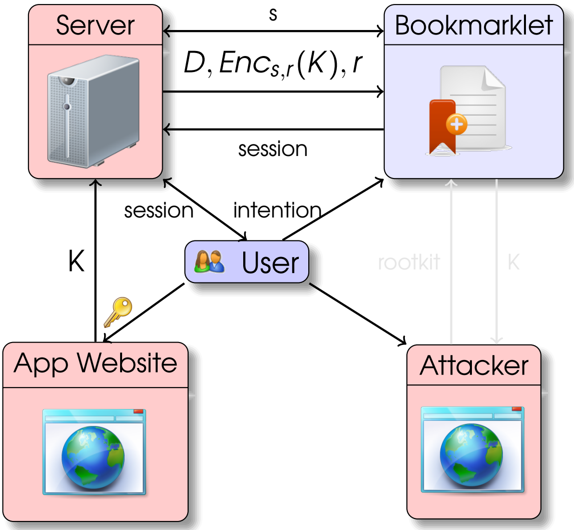
LastPass login bookmarklet



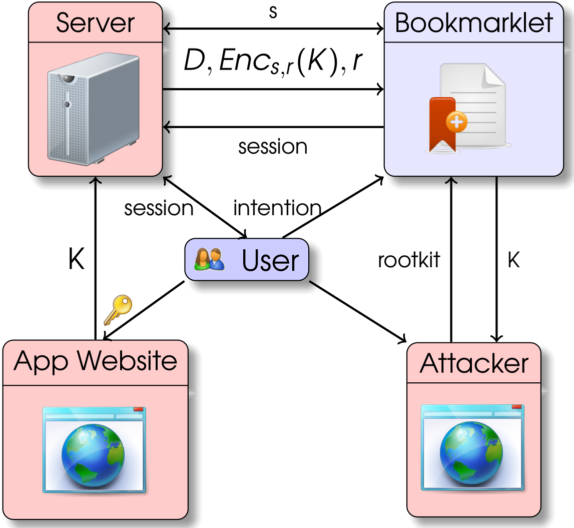
LastPass login bookmarklet

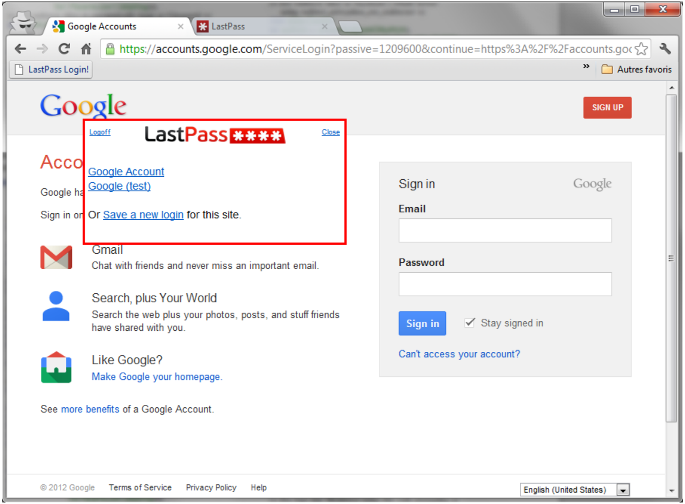


LastPass login bookmarklet



LastPass login bookmarklet





Google Accounts
LastPass
https://accounts.google.com/ServiceLogin?passive=1209600&continue=https%3A%2F%2Faccounts.goo
LastPass Login!
Autres favoris
Google
SIGN UP
Logoff
LastPass ★★★★
Close
Acco
Google Account
Google (test)
Google ha
Sign in or
Or Save a new login for this site.
Gmail
Chat with friends and never miss an important email.
Search, plus Your World
Search the web plus your photos, posts, and stuff friends have shared with you.
Like Google?
Make Google your homepage.
See more benefits of a Google Account.
Sign in
Google
Email
Password
Sign in
Stay signed in
Can't access your account?
© 2012 Google
Terms of Service
Privacy Policy
Help
English (United States)

Key recovery by rootkiting

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.

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.

Available building blocks

Server side

- `Origin` HTTP header.
- `Access-Control-Allow-Origin`.
- Content Security Policy.

Client side

- Browser extension sandboxing.
- Web Crypto API
- ECMA 5
- 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 5
- 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

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.

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 Authrorization by Formal Analysis
CSF 2012 (to appear)

Akhawe, Barth, Lam, Mitchell, Song
Towards a Formal Foundation of Web Security
CSF 2010

Thank you.

... as well as ERC and USENIX Sponsors.