Stony Brook University

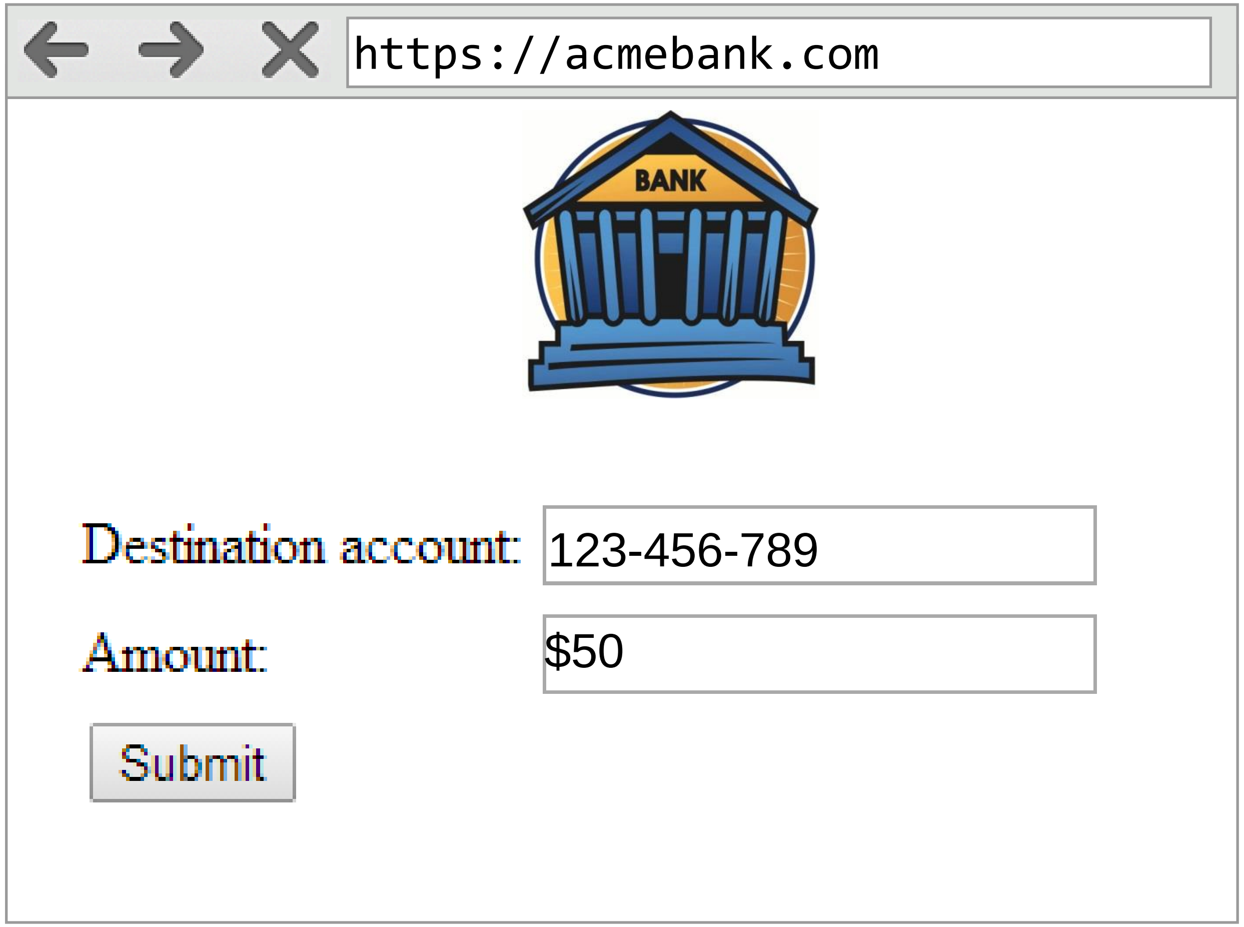
CSE 361: Web Security

CSRF, XSSI, SRI, and Sandboxing

Nick Nikiforakis

CSRF (Sea Surf)

Regular Web site usage



https://acmebank.com

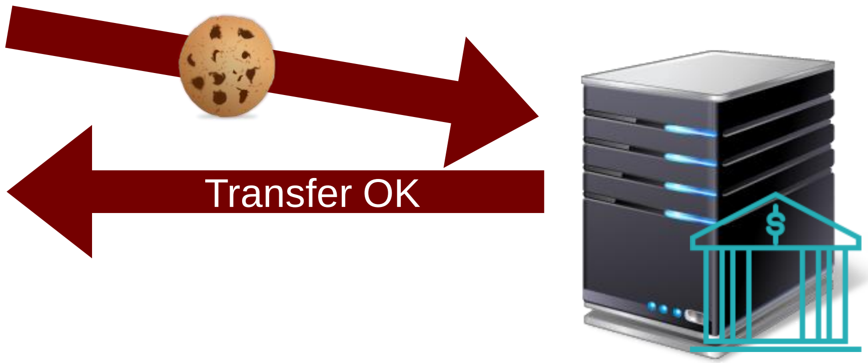
Destination account: 123-456-789

Amount: $50

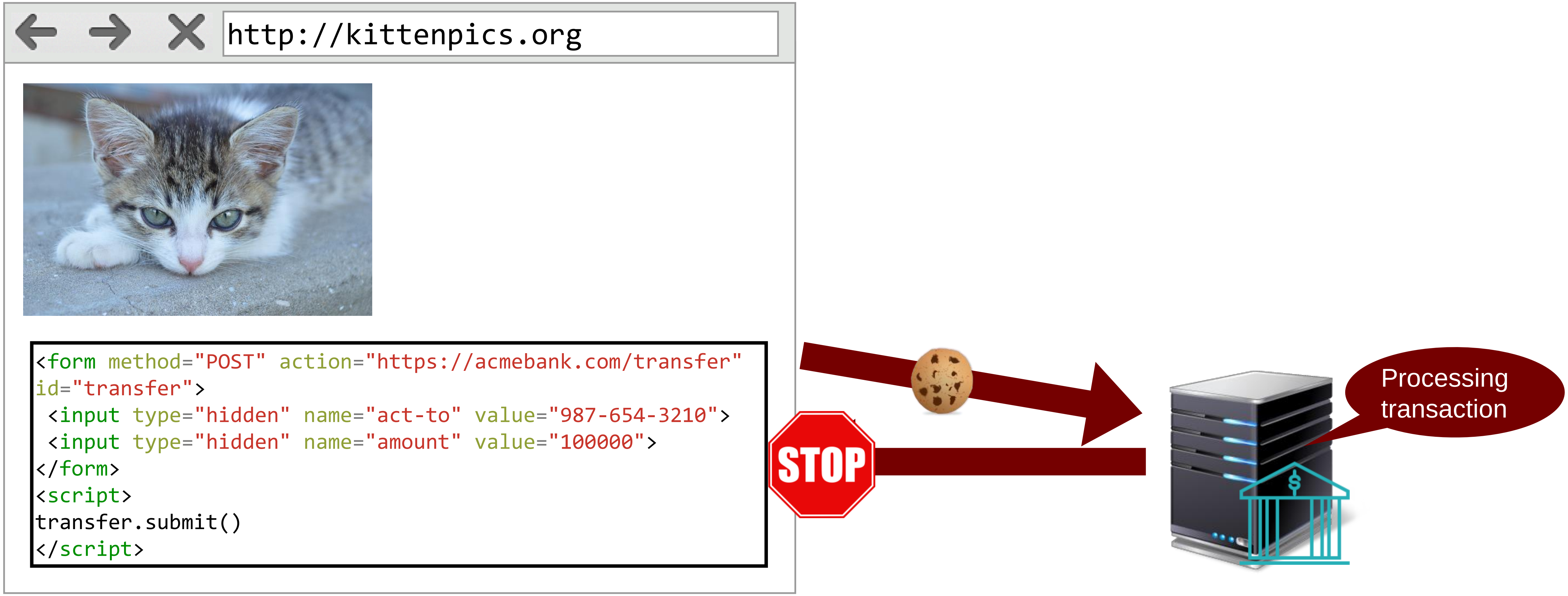
Submit

Behind the scenes

```
<form method="POST"
target=https://acmebank.com/transfer>
        <input type="text" name="acct-to">
        <input type="text" name="amount">
        <input type="submit">
</form>
```



Forcing browser to perform an action for the attacker



http://kittenpics.org

```
<form method="POST" action="https://acmebank.com/transfer"
id="transfer">
 <input type="hidden" name="act-to" value="987-654-3210">
 <input type="hidden" name="amount" value="100000">
</form>
<script>
transfer.submit()
</script>
```

Cross-Site Request Forgery (CSRF / "Sea Surf")

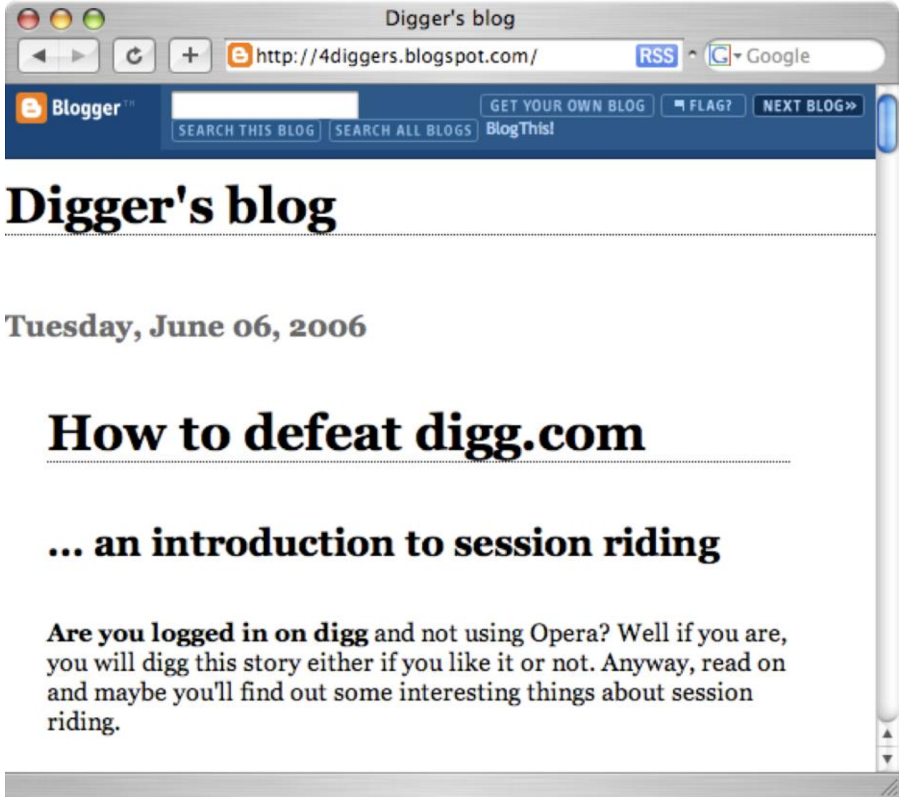
- Web application does not ensure that state-changing request came from "within" the application itself
- Attack works for GET ...
  - Image tag with src attribute:

```
<img src="https://acmebank.com/transfer?to=attacker&amount=10000">
```

  - Hidden iframes, css files, scripts, ...
- and POST
  - create iframe (or pop-up window)
  - fill created viewport with prefilled form
  - submit form

CSRF Examples: digg.com (2006)

- digg.com determines frontpage based on how many "diggs" a story gets
- vulnerable against CSRF, could be used to digg an URL of the attacker's choosing
- Guess which article made it to the front page...



Digger's blog

Tuesday, June 06, 2006

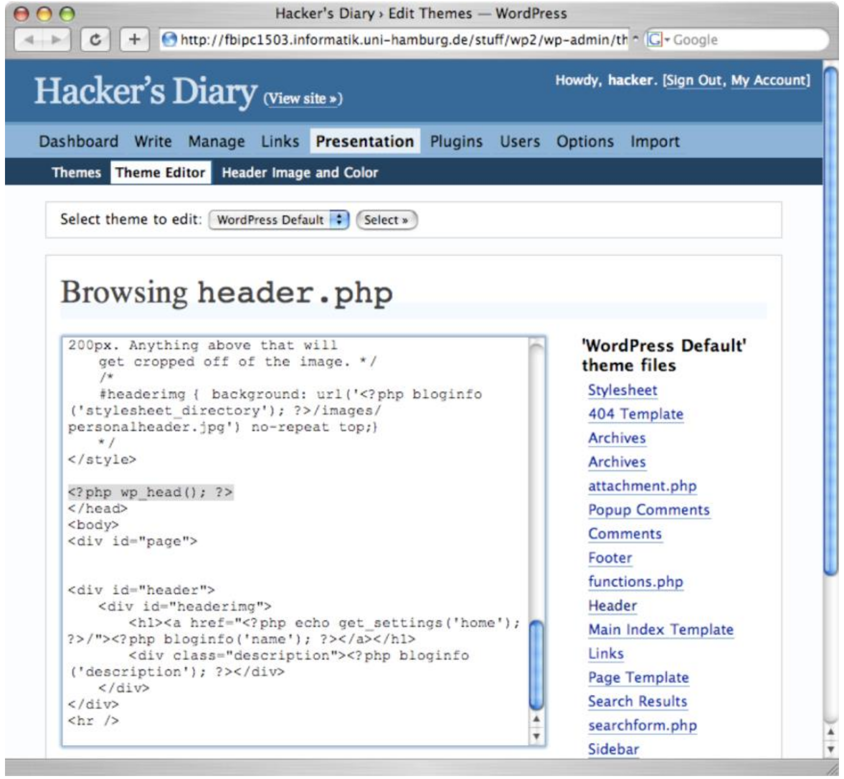
How to defeat digg.com

... an introduction to session riding

Are you logged in on digg and not using Opera? Well if you are, you will digg this story either if you like it or not. Anyway, read on and maybe you'll find out some interesting things about session riding.

CSRF Example: WordPress < 2.06 (2007)

- WordPress theme editor was susceptible
- WordPress themes are PHP files
- Attacker could modify files when logged-in admin visited his page
  - arbitrary code execution on targeted page

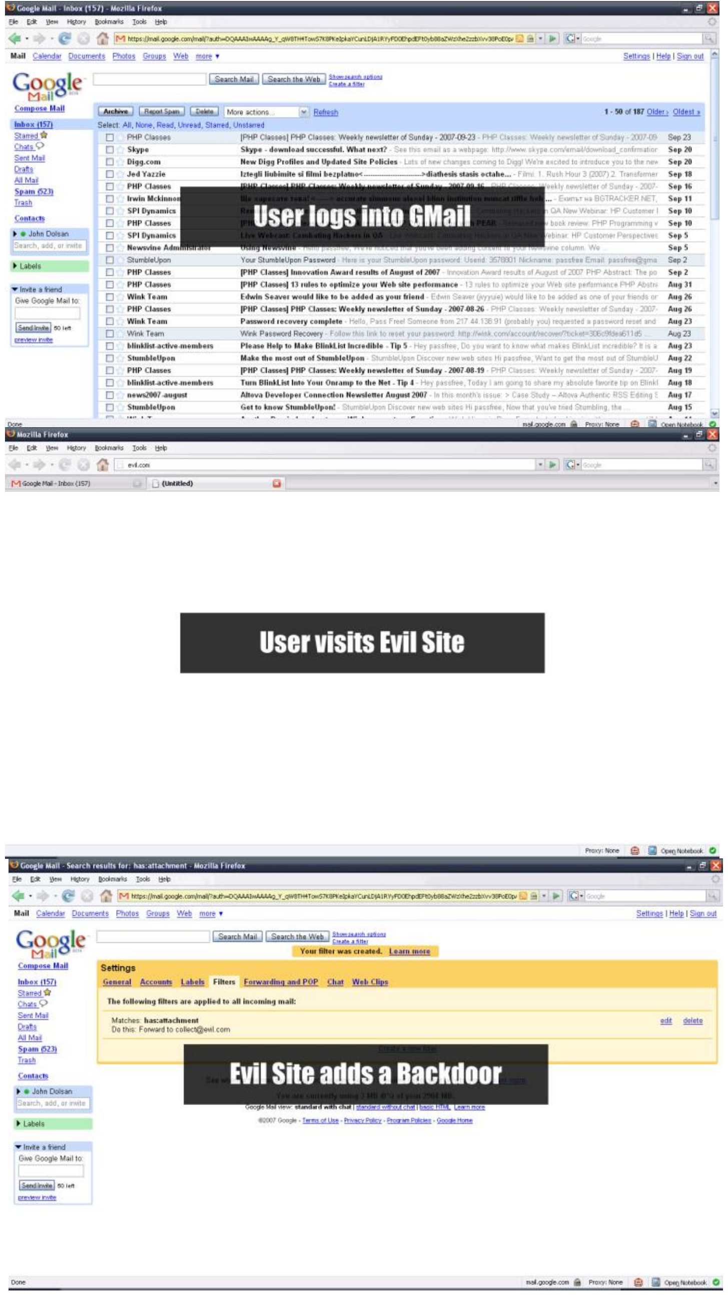


Browsing header.php

```
200px. Anything above that will
    get cropped off of the image. */
    /*
    #headerimg { background: url('<?php bloginfo
('stylesheet_directory'); ?>/images/
personalheader.jpg') no-repeat top;}
    */
</style>

<?php wp_head(); ?>
</head>
<body>
<div id="page">


<div id="header">
    <div id="headerimg">
        <h1><a href="<?php echo get_settings('home');
?>/"><?php bloginfo('name'); ?></a></h1>
        <div class="description"><?php bloginfo
('description'); ?></div>
    </div>
</div>
<hr />
```

CSRF Example: Gmail filters (2007)

- Google Mail insufficiently protected against CSRF
- Attacker could add mail filters
  - e.g., forward all emails to a certain address
- According to a victim, this led to a domain takeover
  - Attacker adds redirect filter
  - Attacker request AUTH code for domain transfer
  - Voila
  - Actually, this incident occurred after the bug was fixed…



Your filter was created. Learn more

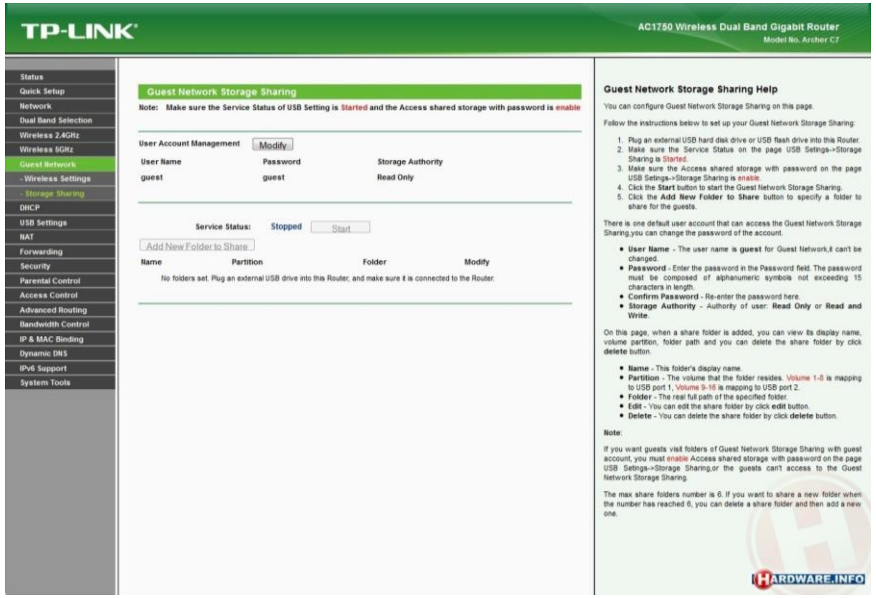
The following filters are applied to all incoming mail:

Matches: has:attachment
Do this: Forward to collect@evil.com

http://www.davidairey.com/google-gmail-security-hijack/

CSRF Example: TP-Link routers (CVE-2013-2645)

- TP-Link Web interface was vulnerable to configuration changes via CSRF
  - set root of built-in FTP server, enable FTP via WAN, ...
  - modify DNS server
- Exploited in the wild to change DNS server
  - redirects all DNS traffic to attacker's server
    - leaking all visited domains
    - allowing for trivial MitM attacks
- Only worked when user was logged in



Guest Network Storage Sharing

Note: Make sure the Service Status of USB Setting is Started and the Access shared storage with password is enable

User Account Management

User Name	Password	Storage Authority
guest	guest	Read Only

Service Status: Stopped

Name	Partition	Folder	Modify

No folders set. Plug an external USB drive into this Router, and make sure it is connected to the Router.

Guest Network Storage Sharing Help

You can configure Guest Network Storage Sharing on this page.

Follow the instructions below to set up your Guest Network Storage Sharing:

1. Plug an external USB hard disk drive or USB flash drive into this Router.
2. Make sure the Service Status on the page USB Setings->Storage Sharing is Started.
3. Make sure the Access shared storage with password on the page USB Setings->Storage Sharing is enable.
4. Click the **Start** button to start the Guest Network Storage Sharing.
5. Click the **Add New Folder to Share** button to specify a folder to share for the guests.

There is one default user account that can access the Guest Network Storage Sharing,you can change the password of the account.

- **User Name** - The user name is **guest** for Guest Network,it can't be changed.
- **Password** - Enter the password in the Password field. The password must be composed of alphanumeric symbols not exceeding 15 characters in length.
- **Confirm Password** - Re-enter the password here.
- **Storage Authority** - Authority of user: **Read Only** or **Read and Write**.

On this page, when a share folder is added, you can view its display name, volume partition, folder path and you can delete the share folder by click **delete** button.

- **Name** - This folder's display name.
- **Partition** - The volume that the folder resides. Volume 1-8 is mapping to USB port 1, Volume 9-16 is mapping to USB port 2.
- **Folder** - The real full path of the specified folder.
- **Edit** - You can edit the share folder by click **edit** button.
- **Delete** - You can delete the share folder by click **delete** button.

Note:

If you want guests visit folders of Guest Network Storage Sharing with guest account, you must enable Access shared storage with password on the page USB Setings->Storage Sharing,or the guests can't access to the Guest Network Storage Sharing.

The max share folders number is 6. If you want to share a new folder when the number has reached 6, you can delete a share folder and then add a new one.

CSRF in 2017 to 2019

- CVE-2017-7404 D-Link router, firmware upload possible
- CVE-2017-9934 Joomla! CSRF to XSS
- CVE-2018-100053 LimeSurvey Delete Themes
- CVE-2018-6288 Kaspersky Secure Mail Gateway Admin Account Takeover
- CVE-2019-10673 WordPress CSRF to change admin email, password recovery for full compromise

(Not really) Preventing CSRF: Refer(r)er Checking

- CSRF entails cross-domain requests
  - in theory, these should carry a referrer
  - server could decide based on header
- In practice, there are several problems
  - Middleboxes/proxies might strip (complete URL is sent, privacy concerns)
  - Attacker may strip Referer header by
    - using a `data:` URL
    - Referrer-Policy header
- Utility vs. Security trade-off
  - what do we do when the header is not present?

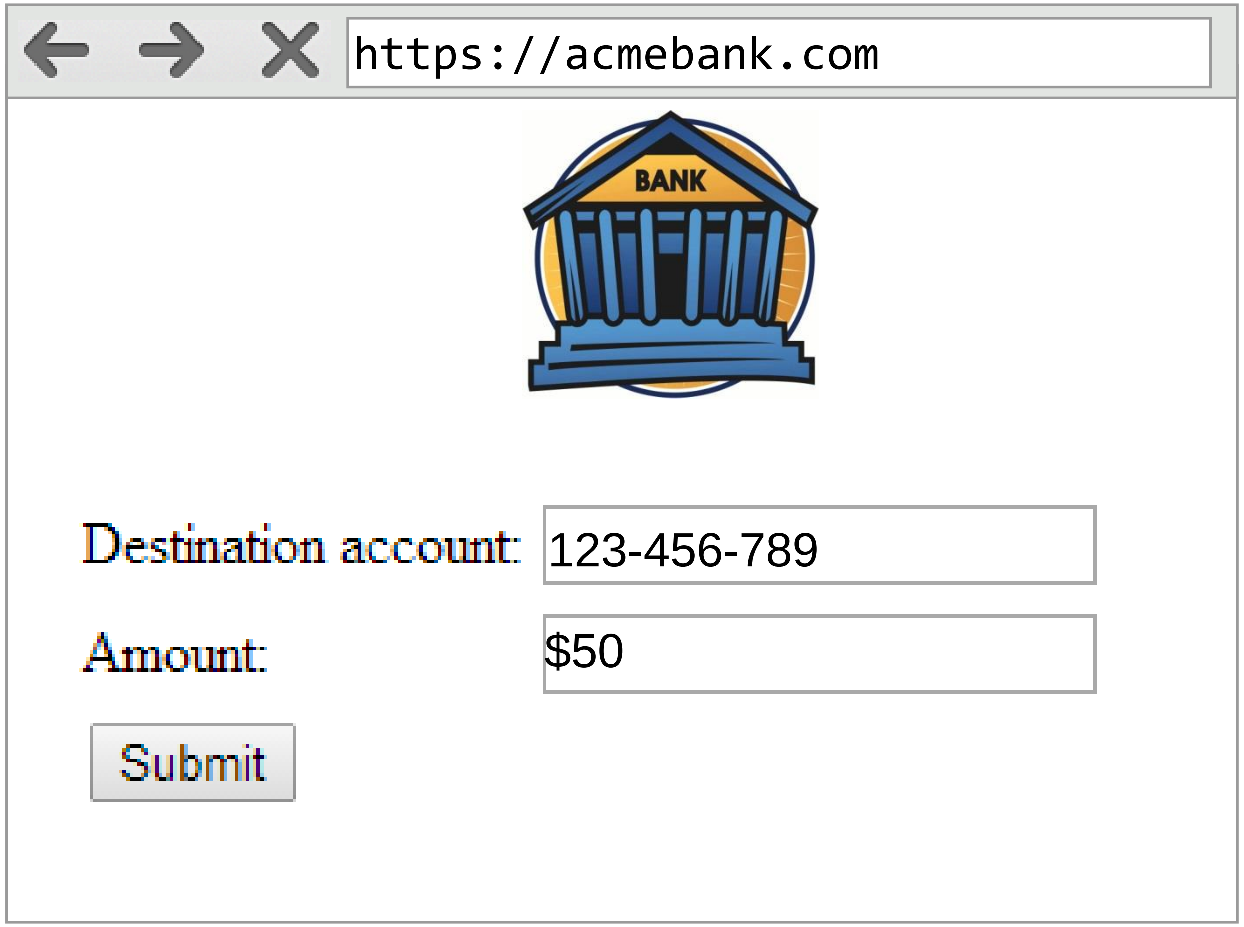
Preventing CSRF: Origin Header Checking

- Privacy-friendly version of Referer
  - Contains only the origin, not the complete URL
- Always sent along XMLHttpRequests and WebSockets
  - requires changing program logic to use these requests for state-changing operations
- In **modern browsers**, also sent along with any **cross-origin POST requests**
  - server should not necessarily rely on only having modern clients, though

What the third-party website receives

Mechanism	Sent URL
Referer	https://www.news.com/blahblah?foo=bar
Origin	https://www.news.com

Regular Web site usage



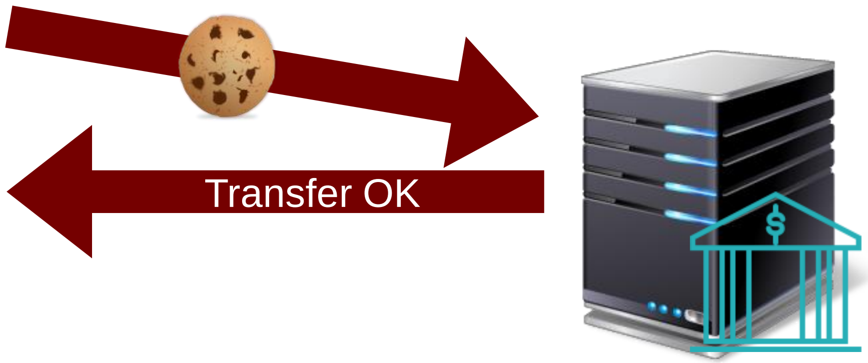
https://acmebank.com

Destination account: 123-456-789

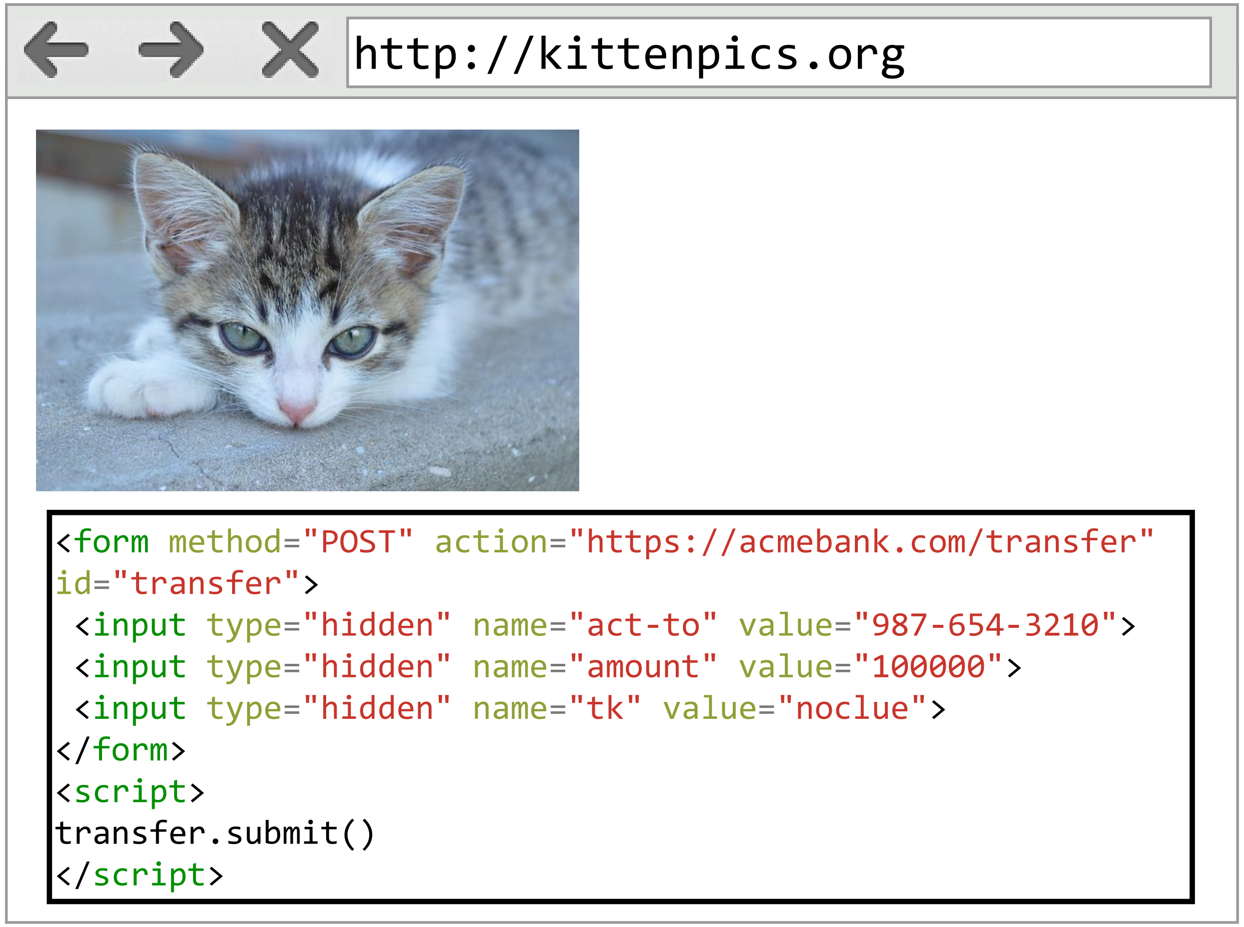
Amount: $50

Behind the scenes

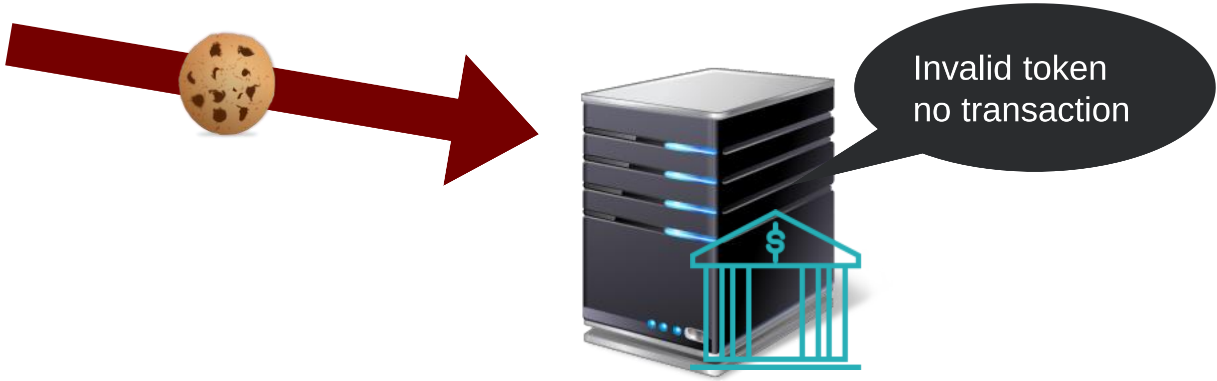
```
<form method=“POST”
target=https://acmebank.com/transfer>
   <input type=“text” name=“acct-to”>
   <input type=“text” name=“amount”>
  <input type=“hidden” name=“tk” value=“n73gn9ia345ntu”
   <input type=“submit”>
</form>
```



Preventing CSRF: Using CSRF tokens/nonces



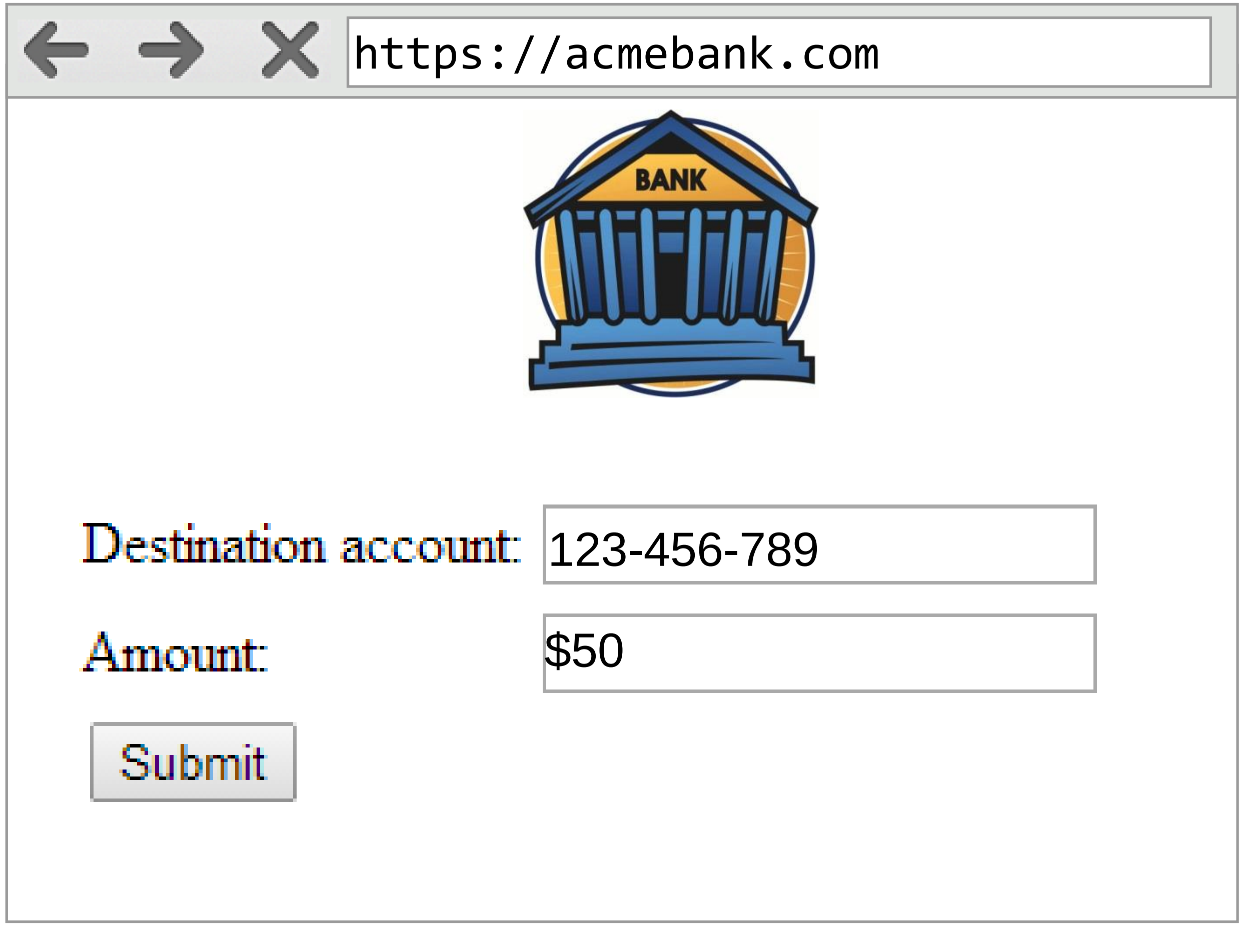
```
<form method="POST" action="https://acmebank.com/transfer"
id="transfer">
 <input type="hidden" name="act-to" value="987-654-3210">
 <input type="hidden" name="amount" value="100000">
 <input type="hidden" name="tk" value="noclue">
</form>
<script>
transfer.submit()
</script>
```



Preventing CSRF: Using CSRF tokens/nonces

- Server generates token randomly for user
  - stores currently valid token in session for user
- Tokens are placed in all forms
  - inaccessible to the attacker without an XSS due to the SOP
- On submission, checks server-side token against submitted token
  - only allows action if tokens match
- Assures that a request's origin must be in the same origin

Preventing CSRF: Double Submit Cookie

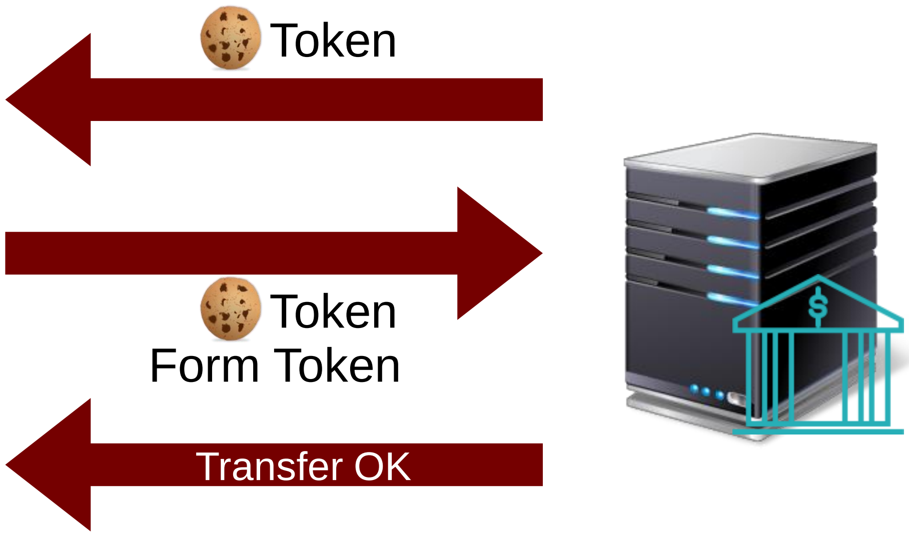


https://acmebank.com

Destination account: 123-456-789

Amount: $50

Submit



Preventing CSRF: Double Submit Cookie

- Require value in posted content to match value of certain cookie
  - generate token randomly on server, store in cookie
  - insert cookie's value into each form
    - server-side addition for protected forms or
    - via JavaScript after form was loaded
- Advantage: no server-side state required
  - just compare submitted form value against cookie
- Disadvantage: cookie tossing
  - If an attacker controls a subdomain, he might set token value
  - if the server only compares cookie and form token, CSRF protection is bypassed

Preventing CSRF: Custom Headers

- Idea: use XMLHttpRequests for all state-changing requests
  - and attach a custom header (e.g., "X-CSRF-Free")
  - only handle requests with that header on the server
- Protection by existing technologies
  - Same-domain requests are always allowed
  - Cross-domain requests with custom headers requires pre-flight CORS request
- Advantage: no server-side state or randomness required
- Disadvantage: applications must be changed

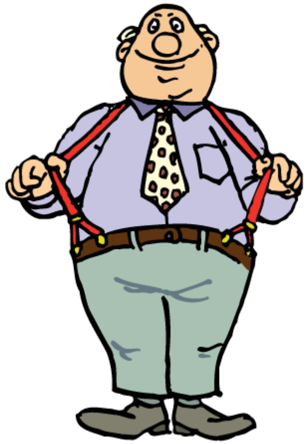
Preventing CSRF: Same-Site Cookies

- Two modes
  - Strict: even in top-level navigation, never send cookies with cross-origin request
    - if facebook.com set that, every user following a link there would not be logged in
  - Lax: non top-level navigation will not send cookies
    - cookies only send along with safe requests (GET, HEAD, OPTIONS, TRACE)
    - protects against POST-based CSRF, not against GET-based though

- Until May 2018 only supported by Chrome and Opera
- Since Chrome 80, defaults to SameSite=lax
  - SameSite=none only works with Secure flag

CSRF Conclusion

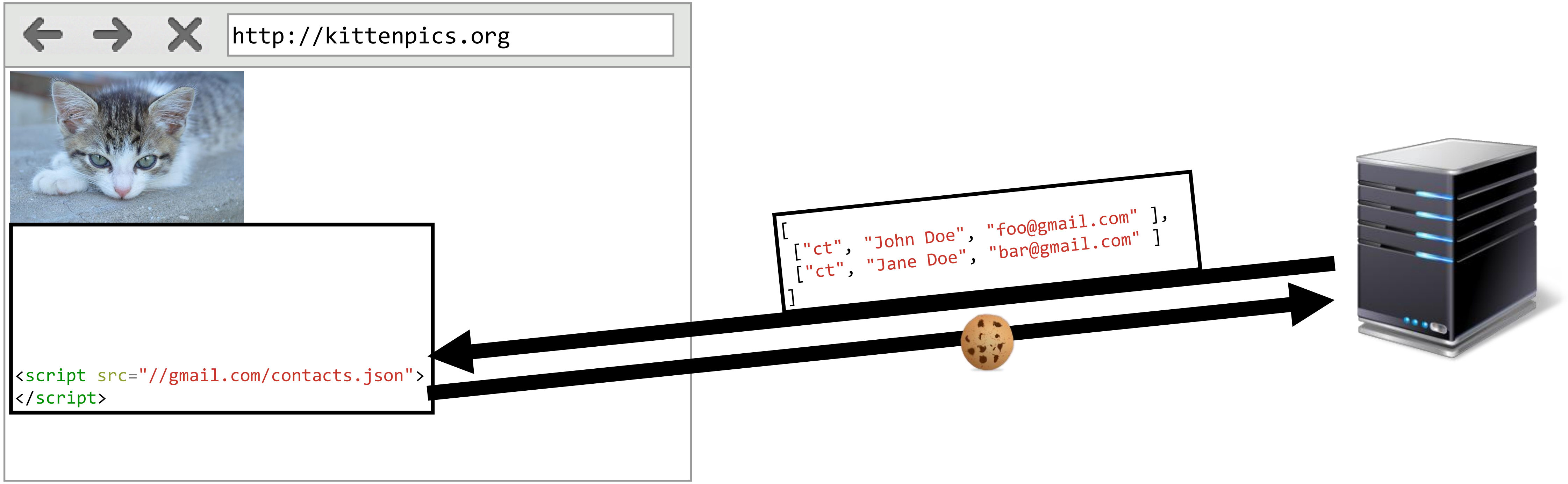
- CSRF caused by servers accepting requests from outside their origin
  - hard to determine based on Referer header though
- CSRF can have severe effects
  - compromised firmware, hijacked Web sites, ...
- Several options for fixing exist
  - CSRF tokens nowadays implemented in any (good) framework
  - protection can be achieved using well-established principles (SOP, CORS)
  - SameSite cookies also address the issue, already default in Chrome
- Support still varies (https://caniuse.com/?search=samesite)
  - Use defense in depth



Cross-Origin Data Leakage

JSON/JavaScript Hijacking (2006)

- Recall from previous lectures
  - script inclusion is exempt from SOP
  - all requests are made with cookies attached

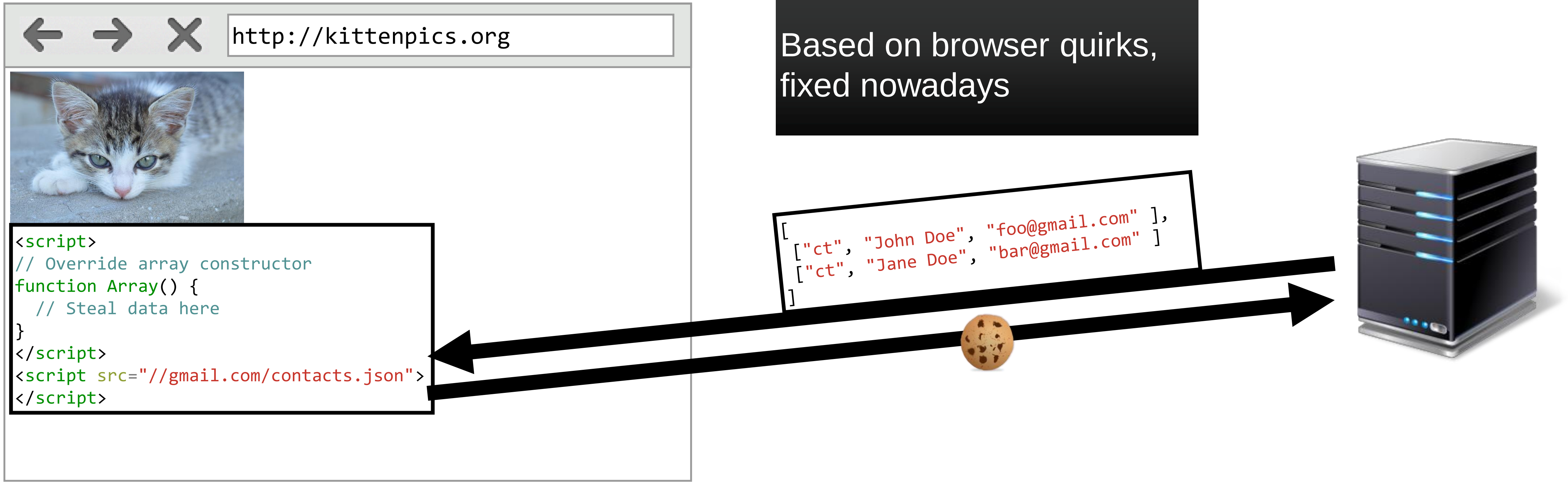


```
<script src="//gmail.com/contacts.json">
</script>
```

```
[
 ["ct", "John Doe", "foo@gmail.com" ],
 ["ct", "Jane Doe", "bar@gmail.com" ]
]
```

JSON/JavaScript Hijacking (2006)

- Recall from previous lectures
  - script inclusion is exempt from SOP
  - all requests are made with cookies attached



http://kittenpics.org

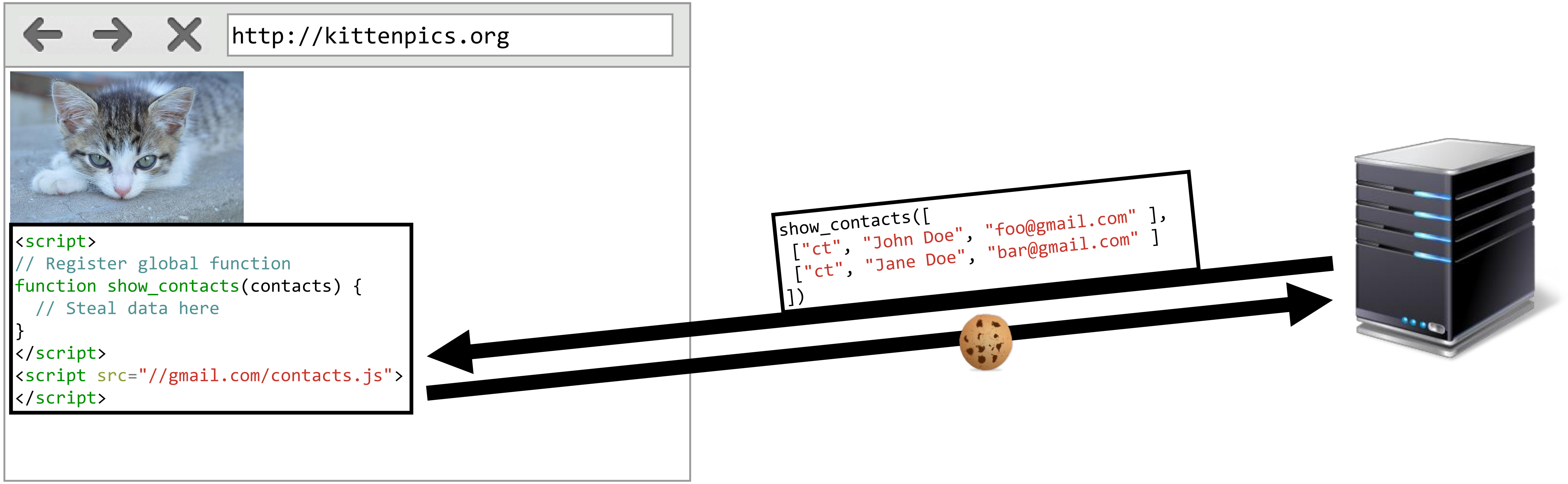
```
<script>
// Override array constructor
function Array() {
  // Steal data here
}
</script>
<script src="//gmail.com/contacts.json">
</script>
```

```
[
 ["ct", "John Doe", "foo@gmail.com" ],
 ["ct", "Jane Doe", "bar@gmail.com" ]
]
```

Based on browser quirks, fixed nowadays

Cross-Site Scripting Inclusion (XSSI)

- Regular scripts may also be dynamically generated
  - We cannot read the source code, but can observe side-effects



```
<script>
// Register global function
function show_contacts(contacts) {
  // Steal data here
}
</script>
<script src="//gmail.com/contacts.js">
</script>
```

```
show_contacts([
 ["ct", "John Doe", "foo@gmail.com" ],
 ["ct", "Jane Doe", "bar@gmail.com" ]
])
```

Exploiting XSSI

```
// Local variable at top level
var first_name = "John";
// Global variable due to missing var keyword
last_name = "Doe";
// Explicitly defined global variable
window.user_email = "john@doe.com";
```

```
console.log(first_name);
console.log(last_name);
console.log(user_email);
```

```
function example() {
  var email = "john@doe.com";
  window.MyLibrary.doSomething(email);
}

example();
```

```
window.MyLibrary = {};
window.MyLibrary.doSomething =
function(email) { console.log(email); }
```

Exploiting XSSI

```
function example2() {
  var secret_values = ["secret", "more secret"];

  secret_values.forEach(function(secret) {
    // do something secret in here
  });
}
example2();
```

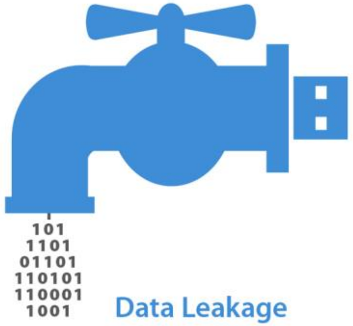
```
Array.prototype.forEach = function(callback) {
  // "this" is bound secret_values
  console.log(this);
}
```

```
(function() {
  function test(someInput) {
    var email = "john@doe.com";
    doNothingWithEmail(someInput);
  }

  test.call(someThing, "myInput");
})();
```

```
Function.prototype.call = function() {
  // "this" is bound test
  console.log(this.toString());
};
```

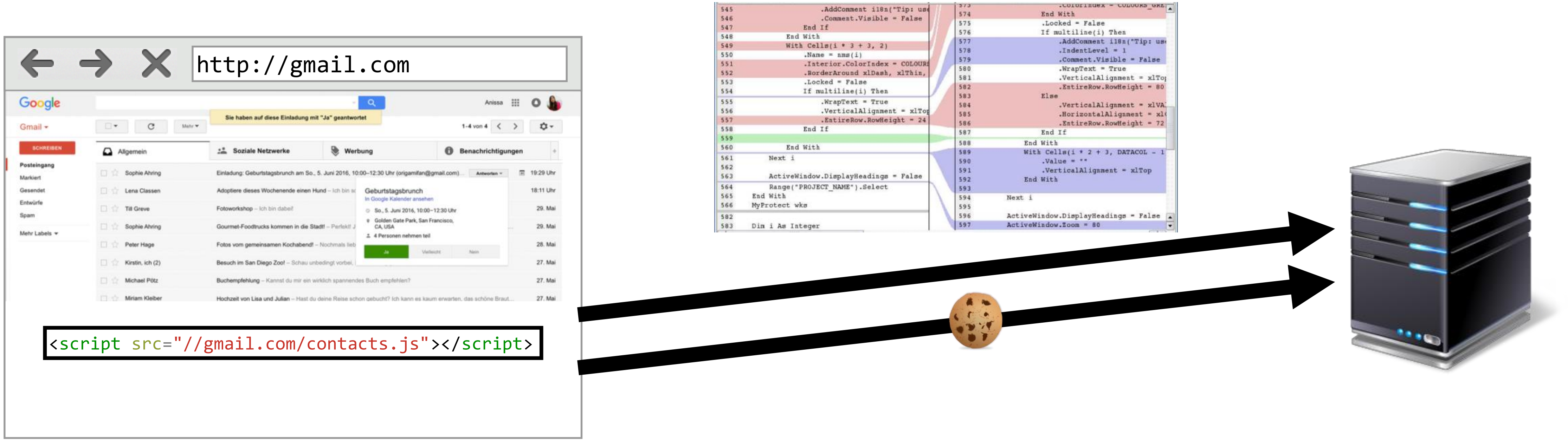
Exploiting XSSI

- Trivial case: global variables registered
  - simply access the variable (registered in global scope of site)
- Little more involved: global function called
  - overwrite function (if necessary, create object before)
- Local variables accessible if functions are called on them
  - overwrite prototype
  - e.g., forEach or call



Identifying potential XSSI [USENIX15]

- On each page visit, request included scripts twice
  - with and without cookies
- Diff the two results



```
<script src="//gmail.com/contacts.js"></script>
```

```
                    .AddComment i18n("Tip: use
                    .Comment.Visible = False
                End If
            End With
            With Cells(i * 3 + 3, 2)
                .Name = nms(i)
                .Interior.ColorIndex = COLOURS
                .BorderAround xlDash, xlThin,
                .Locked = False
                If multiline(i) Then
                    .WrapText = True
                    .VerticalAlignment = xlTop
                    .EntireRow.RowHeight = 24
                End If

            End With
        Next i

        ActiveWindow.DisplayHeadings = False
        Range("PROJECT_NAME").Select
    End With
    MyProtect wks

    Dim i As Integer
```

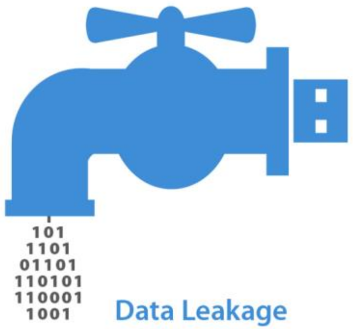
```
                    .ColorIndex = COLOURS_GRE
                End With
                .Locked = False
                If multiline(i) Then
                    .AddComment i18n("Tip: use
                    .IndentLevel = 1
                    .Comment.Visible = False
                    .WrapText = True
                    .VerticalAlignment = xlTop
                    .EntireRow.RowHeight = 80
                Else
                    .VerticalAlignment = xlVA
                    .HorizontalAlignment = xl(
                    .EntireRow.RowHeight = 72
                End If
            End With
            With Cells(i * 2 + 3, DATACOL - 1
                .Value = ""
                .VerticalAlignment = xlTop
            End With

        Next i

        ActiveWindow.DisplayHeadings = False
        ActiveWindow.Zoom = 80
```

XSSI in the Wild [USENIX15]

- Conducted a study of 150 highest-ranked sites with logins
  - sites for which we could create a login (not banks, for example)

	Domains	Exploitable
Dynamic scripts	49	40
Unique identifier	34	28
Other personal data	15	11
CSRF / auth tokens	7	4

- Several high impact flaws
  - leaked credit card info on my own bank
  - reading senders and subjects of emails
  - account hijacking for file hosting service

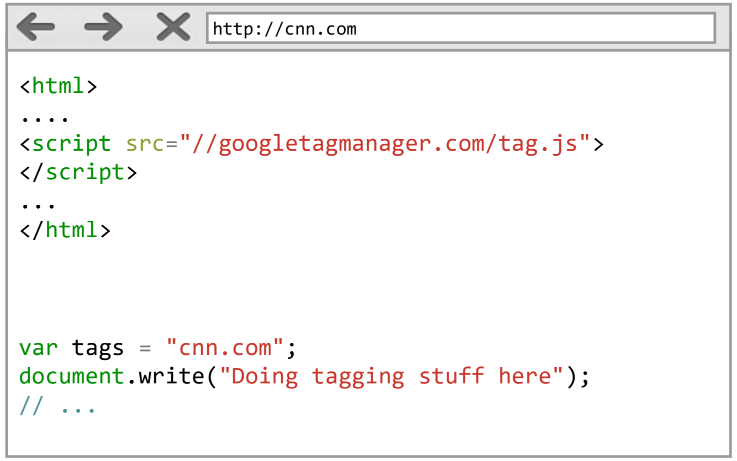


Preventing XSSI

- Scripts must not be loadable from other origins
  - referrer checking (recall the problems associated with that)
  - use of secret tokens (similar to CSRF)
- Only provide code in scripts, use provisioning service for data
  - use XHR to retrieve data
  - easily protectable by SOP or CORS
- Use inline scripts only
  - with CSP nonces, even possible to use with CSP
  - can not be included remotely, hence data is secure there

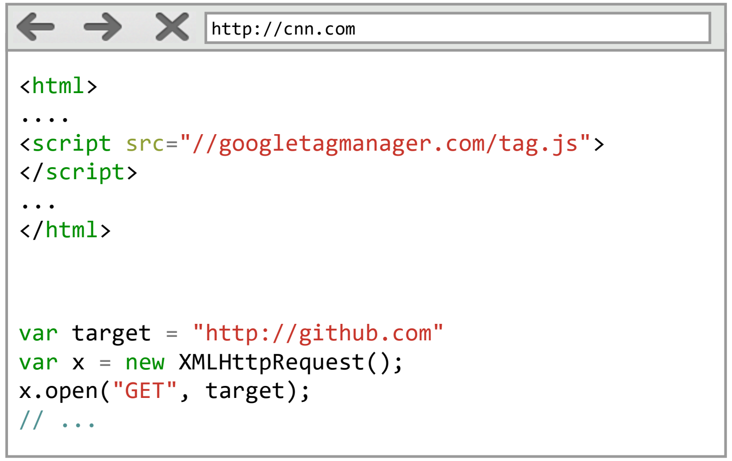
The Great Cannon

Including third-party resources on the Web

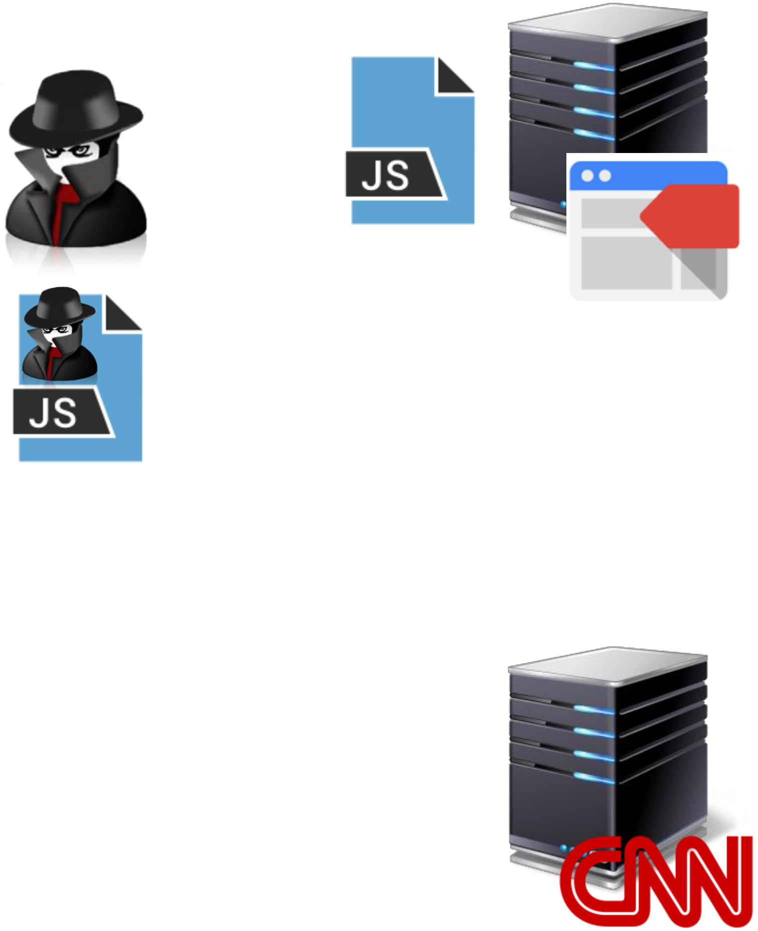


```
<html>
....
<script src="//googletagmanager.com/tag.js">
</script>
...
</html>



var tags = "cnn.com";
document.write("Doing tagging stuff here");
// ...
```

Including third-party resources on the Web (with MitM)

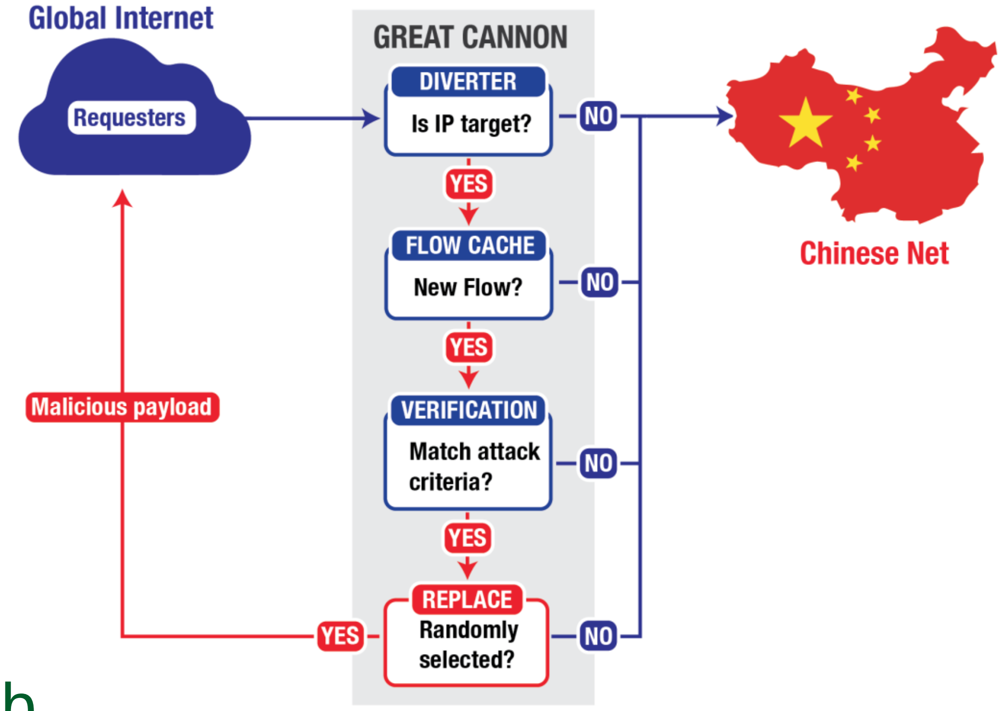


```
<html>
....
<script src="//googletagmanager.com/tag.js">
</script>
...
</html>


var target = "http://github.com"
var x = new XMLHttpRequest();
x.open("GET", target);
// ...
```



The Great Cannon

- China already has a powerful firewall
  - "The Great Firewall"
  - drops unwanted connections (e.g. NY Times)
- Mirror sites exists for blocked sites
  - e.g., greatfire.org and several GitHub repos
- Great Cannon injected JavaScript into content from, e.g., baidu.com
  - millions of users opened connections to GitHub, New York Times, greatfire.org
- Massively Distributed Denial of Service



https://citizenlab.ca/2015/04/chinas-great-cannon/

Subresource Integrity (SRI)

- To thwart such injection attacks, SRI was proposed
- Use cryptographic hash of remote resource
  - for scripts and style sheets
  - if hash does not match, resource is ignored

```
<script src="https://code.jquery.com/jquery-2.1.4.min.js"
integrity="sha384-R4/ztc4ZlRqWjqIuvf6RX5yb/v90qNGx6fS48N0tRxiGkqveZETq72KgDVJCp2TC"
crossorigin="anonymous"></script>
```

- Protects against malicious CDNs/MitM attackers
  - also allows to pin to a specific version of third-party libraries

-
  ```
  <script>window.jQuery || /* reload from own domain here */;</script>
  ```

Subresource Integrity (SRI)

- SRI resources must be CORS-enabled
  - otherwise, SRI could be used to test remote resource for certain content
- Integrity attribute can have multiple values
  - Only strongest hash is used

```
<script src="https://code.jquery.com/jquery-2.1.4.min.js"
integrity="sha384-R4/ztc4ZlRqWjqIuvf6RX5yb/v90qNGx6fS48N0tRxiGkqveZETq72KgDVJCp2TC sha256-
8WqyJLuWKRBVhxXIL1jBDD7SDxU936oZkCnxQbWwJVw="
crossorigin="anonymous"></script>
```

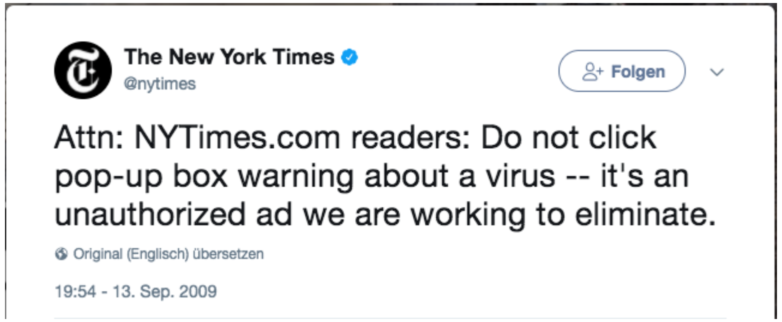
  - Cannot be used to allow different versions of a script

```
<script src="https://code.jquery.com/jquery-2.1.4.min.js"
integrity="sha256-t1X5SBfMY4/0kYdt8H1CP/90GgOi1G6U9UnjC6AVYHA=
sha256-8WqyJLuWKRBVhxXIL1jBDD7SDxU936oZkCnxQbWwJVw="
crossorigin="anonymous"></script>
```

Sandboxing Content

Multi-origin Web applications

- Modern Web applications use code from multiple origins
  - Analytics
  - Advertisement
  - Maps
  - ....
- Even framed content may, e.g., open a popup
  - or redirect the parent frame
- Necessity for control privileges of included content arises
  - putting everybody in their own little sandbox


The New York Times
@nytimes

Attn: NYTimes.com readers: Do not click pop-up box warning about a virus -- it's an unauthorized ad we are working to eliminate.

Original (Englisch) übersetzen

19:54 - 13. Sep. 2009

Sandboxing iframes

- Limits iframe's ability to conduct certain actions
  - e.g., disable JavaScript, putting them in an isolated origin
- Just adding sandbox to the iframe will restrict everything
  - rights have to be granted explicitly
  - `allow-forms`: allows for form submission in iframe
  - `allow-popups`: enables popups
  - `allow-pointer-lock`: enable PointerLock API to get raw mouse movements
  - `allow-scripts`: enable scripting
  - `allow-same-origin`: enable origin of included page, not isolated one
  - `allow-top-navigation`: enables navigating the top frame

Sandbox usage examples

```
<textarea id='code'></textarea>
<button id='safe'>eval() in a sandboxed frame.</button>
<iframe sandbox='allow-scripts' id='sbox' src='frame.html'>
</iframe>

<script>
  function evaluate() {
    sandboxed.contentWindow.postMessage(code.value, '*');
  }
  safe.addEventListener('click', evaluate);

  window.addEventListener('message', function (e) {
    if (e.origin === "null" && e.source === sbox.contentWindow)
      alert('Result: ' + e.data);
    });
</script>
```

Parent page

```
<script>
  window.addEventListener('message', function (e) {
    if (e.origin !== "https://main.com") {
      return
    }
    var mainWindow = e.source;
    var result = '';

    try {
      result = eval(e.data);
    } catch (e) {
      result = 'eval() threw an exception.';
    }
    mainWindow.postMessage(result, e.origin);
  });
</script>
```

frame.html

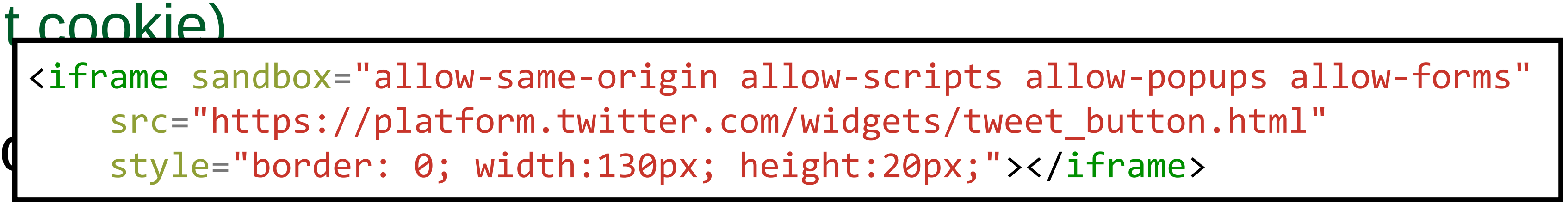
https://www.html5rocks.com/static/demos/evalbox/index.html

Determining least privilege

- Example: tweet button
  - opens popup window
  - submit a form
  - sends authenticated request to twitter.com (using and accesses document.cookie)
- Requires four permissions
  - allow-popups (well, it opens a popup..)
  - allow-forms (well, it is a form?)
  - allow-same-origin (JavaScript needs access to cookies)
  - allow-scripts (not too much of a surprise)

Determining least privilege

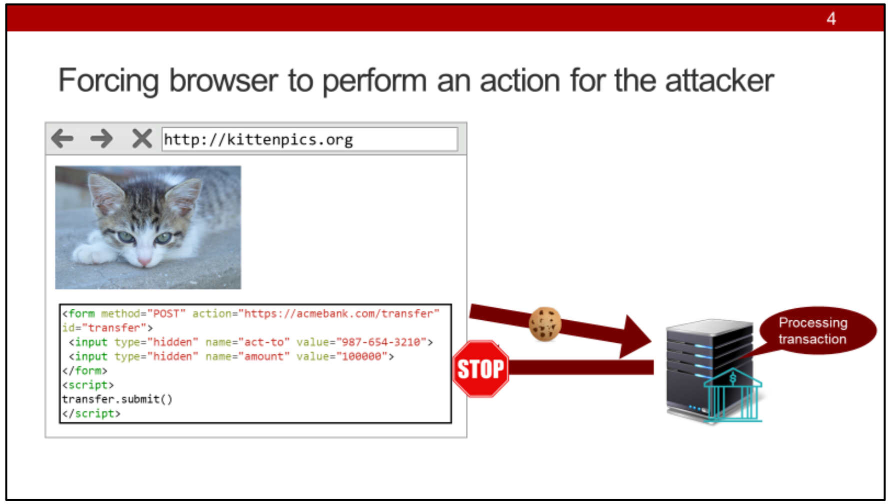
- Example: tweet button
  - opens popup window
  - submit a form
  - sends authenticated request to twitter.com (using and accesses document cookie)



```
<iframe sandbox="allow-same-origin allow-scripts allow-popups allow-forms"
    src="https://platform.twitter.com/widgets/tweet_button.html"
    style="border: 0; width:130px; height:20px;"></iframe>
```

- Requires fo
  - allow-popups (well, it opens a popup..)
  - allow-forms (well, it is a form?)
  - allow-same-origin (JavaScript needs access to cookies)
  - allow-scripts (not too much of a surprise)

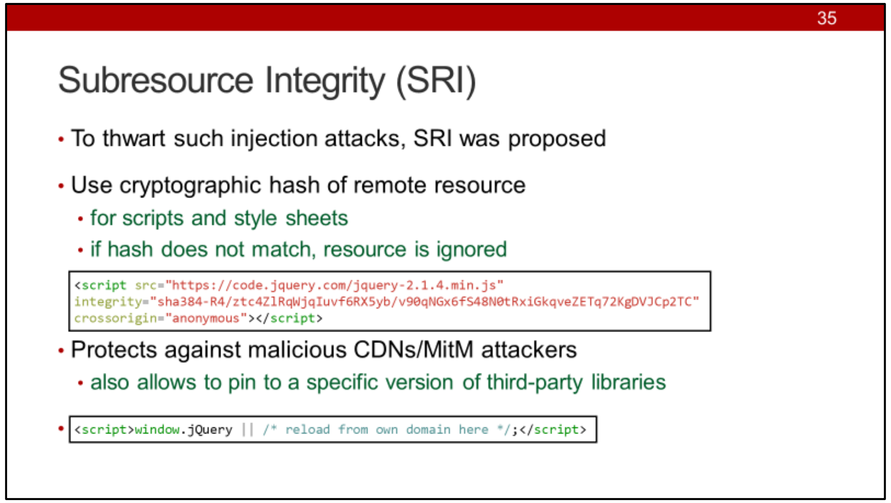
Summary



4

Forcing browser to perform an action for the attacker

```
<form method="POST" action="https://acmebank.com/transfer"
id="transfer">
 <input type="hidden" name="act-to" value="987-654-3210">
 <input type="hidden" name="amount" value="100000">
</form>
<script>
transfer.submit()
</script>
```



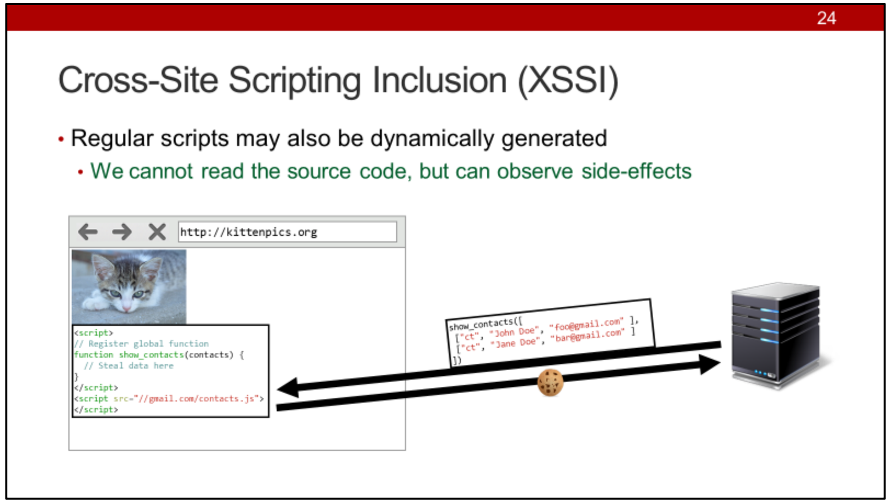
35

Subresource Integrity (SRI)

- To thwart such injection attacks, SRI was proposed
- Use cryptographic hash of remote resource
  - for scripts and style sheets
  - if hash does not match, resource is ignored

```
<script src="https://code.jquery.com/jquery-2.1.4.min.js"
integrity="sha384-R4/ztc4ZlRqWjqIuvf6RX5yb/v90qNGx6fS48N0tRxiGkqveZETq72KgDVJCp2TC"
crossorigin="anonymous"></script>
```

- Protects against malicious CDNs/MitM attackers
  - also allows to pin to a specific version of third-party libraries
- `<script>window.jQuery || /* reload from own domain here */;</script>`



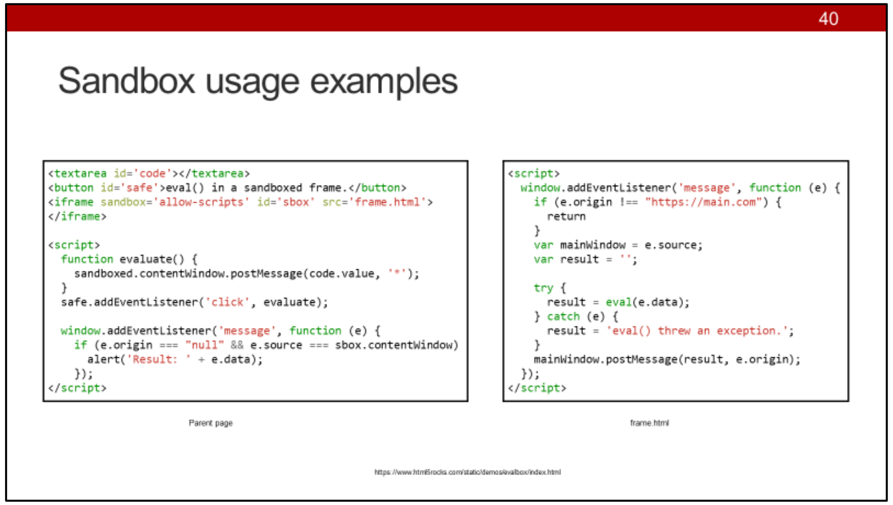
24

Cross-Site Scripting Inclusion (XSSI)

- Regular scripts may also be dynamically generated
  - We cannot read the source code, but can observe side-effects

```
<script>
// Register global function
function show_contacts(contacts) {
  // Steal data here
}
</script>
<script src="//gmail.com/contacts.js">
</script>
```

```
show_contacts([
 ["ct", "John Doe", "foo@gmail.com" ],
 ["ct", "Jane Doe", "bar@gmail.com" ]
])
```



40

Sandbox usage examples

```
<textarea id='code'></textarea>
<button id='safe'>eval() in a sandboxed frame.</button>
<iframe sandbox='allow-scripts' id='sbox' src='frame.html'>
</iframe>

<script>
  function evaluate() {
    sandboxed.contentWindow.postMessage(code.value, '*');
  }
  safe.addEventListener('click', evaluate);

  window.addEventListener('message', function (e) {
    if (e.origin === "null" && e.source === sbox.contentWindow)
      alert('Result: ' + e.data);
    });
</script>
```

Parent page

```
<script>
  window.addEventListener('message', function (e) {
    if (e.origin !== "https://main.com") {
      return
    }
    var mainWindow = e.source;
    var result = '';

    try {
      result = eval(e.data);
    } catch (e) {
      result = 'eval() threw an exception.';
    }
    mainWindow.postMessage(result, e.origin);
  });
</script>
```

frame.html

https://www.html5rocks.com/static/demos/evalbox/index.html

Credits

- Original slide deck by Ben Stock
- Modified by Nick Nikiforakis