HTML5 Security & Headers
- X-Crawling-Response-Header-

JPCERT Coordination Center
Watch and Warning Group, Information Security Analyst
Tomoyuki Shigemori
Overview of JPCERT/CC

- Japan Computer Emergency Response Team Coordination Center
  - Founded in 1996
  - An independent, non-profit organization
  - National CSIRT (Computer Security Incident Response Team)
  - Coordination center

- JPCERT/CC, as a national CSIRT, monitors computer security incidents at a national level, identifies and handles incidents that could affect the economy and critical infrastructures, and warns critical stakeholders and the nation about computer security threats.

- JPCERT/CC, as a coordination center, provides technical support in response to computer security incidents through coordination with other local and overseas CSIRTs.
Overview of JPCERT/CC - 3 pillars and 4 foundations -

Prevent
- Vulnerability Information Handling
  - Coordinate with developers on unknown vulnerability information
  - Secure Coding

Monitor
- Information gathering / analysis / sharing
- Internet Traffic Monitoring
  - Alerts / Advisories

Respond
- Incident Handling
  - Mitigating the damage through efficient incident handling
  - Information sharing to prevent similar incidents

Early Warning Information
Information sharing with critical infrastructure enterprises, etc.

CSIRT Establishment Support
Capacity building for internal CSIRTs in enterprises / overseas national CSIRTs

Artifact Analysis
Analysis on attack methods / behavior of malware (unauthorized program)

International Collaboration
Collaboration with overseas organizations for smoother handling of incidents and vulnerabilities
What is HTML5?

- HTML5 is the next HTML standard.
- HTML5 enables us to build more flexible and highly convenient websites. It allows us to store data within the visitor’s browser, enables full-duplex communication between the visitor’s browser and web servers, obtain location information of the visitor, etc.
- It is becoming widely utilized in Asia-Pacific region including Japan.
- Major browsers are compatible with HTML5 (Some functions are not enabled).
What is HTML5?

- HTML5 is the next HTML standard.
- HTML5 enables us to build more flexible and highly convenient websites. It allows us to store data within the visitor’s browser, enables full-duplex communication between the visitor’s browser and web servers, obtain location information of the visitor, etc.
- It is becoming widely utilized in Asia-Pacific region including Japan.
- Major browsers are compatible with HTML5 (Some functions are not enabled).
- While HTML5 enhances browser capabilities, it also brings new security concerns which were not an issue in previous versions of HTML.
- The possible impacts are not fully verified or widely known, which may affect users by attackers leveraging the security flaws of these new architectures.
Japanese version was published on October 2013 and English version will be coming up soon.

https://www.jpcert.or.jp/research/html5.html (Japanese ver.)
Research Report Terms of Use

The report is expected to be utilized as…
- Technical paper, basic information for guidelines
- Material for study meetings
- Reference for seminars

For citing or reproducing the original document, please refer to the following:

JPCERT/CC Terms of use
https://www.jpcert.or.jp/guide.html (Japanese only)

Sample
Source: JPCERT Coordination Center
“Technical research report on security issue of web applications utilizing HTML5”
https://www.jpcert.or.jp/research/HTML5-20131030.pdf
JavaScript API
XMLHttpRequest
**XMLHttpRequest (XHR) Summary**

- **XMLHttpRequest (XHR):**
  - an API to communicate with HTTP using JavaScript
  - enables interactive web contents by asynchronous communication
  - widely used due to increased usage of AJAX
  - this function could only communicate with same origin in versions earlier than HTML5

> 'Origin': loaded by the original source

> 'Access-Control-Allow-Origin': authorized origin or '*' (wildcard)

In order to allow Cookie, “Access-Control-Allow-Credentials” needs to be configured

※Origin : combination of host name, protocol and port number
Current Status of HTML5
Previous Survey Results

- Survey against 2,359 sites was conducted on November 2012

How about Cross-Origin Resource Sharing (CORS) headers with values other than 'Access-Control-Allow-Origin'?


0.25% of websites supports CORS
Fact-Finding on Websites utilizing HTML5

Following points were checked:
--- CORS headers (*1)
--- Headers with security features (*2)

Approach to fact-finding
--- Crawl top page of websites on Alexa Top 1,000,000 (*3)
--- Check the HTTP header of the curl command response
--- If the website redirects to other points, final redirection point will be surveyed
--- Origin request header will be added to the sending request
--- Survey was conducted from 2013/12/26 to 2013/12/30

(*1) Response header which starts with 'Access-Control' on headers used by CORS
(*2) Specific headers with security features introduced in the research report
CORS Headers

- **Access-Control-Allow-Origin**
  - Specifies the Origin that is allowed to access to resource

- **Access-Control-Allow-Credentials**
  - Configured ‘true’ for permission on accessing to response towards requests with authentication information such as Cookie

- **Access-Control-Expose-Headers**
  - Specifies the header which the browser can use

- **Access-Control-Allow-Methods (preflight)**
  - Specifies the method which allows transmission

- **Access-Control-Allow-Headers (preflight)**
  - Specifies the header which allows transmission

- **Access-Control-Max-Age (preflight)**
  - Specifies the time to cache a preflight response
Headers with Security Features

- **X-XSS-Protection**
  - Protects from XSS attack

- **X-Frame-Options**
  - Manages contents in accordance with the configuration in Content-Type header

- **X-Frame-Options**
  - Restricts embedding into a frame such as iframe

- **Content-Security-Policy**
  - Restricts the sources of contents to be loaded

- **Content-Disposition**
  - Controls download dialog of a file

- **Strict-Transport-Security**
  - Enforces the use of secure HTTPS connections
Key Findings
Number of websites which responded was 949,217 sites.
Number of CORS enabled websites(*) was 7,218 sites.
Only 0.76% of the websites had CORS enabled.

(*) Websites which returned headers related to CORS
5.4% of the websites were configured improperly.

If improperly configured, the value will be invalid and “Access-Control-Allow-Origin” will not work properly.
Access-Control-Allow-Origin

**Proper Configuration**
- `scheme://host[:port][ scheme://host[:port]]*` (※1)
- `null`
- `*`

**Improper configuration sample** (either of the settings will be invalid)
- No scheme: `example.com`
- Multiple configuration with comma breaking: `http://example1.com,http://example2.com`
- Wildcard usage: `http://*.example.com`
- ‘/’ inserted at the end of origin: `http://example.com/`
- Multiple header: `http://example1.com`
  `http://example2.com`

**Attention!! Without the knowledge of proper configuration on how to add headers, it may weaken the authorization and create vulnerabilities.**

※1: Some browsers forbid origin configuration with space breaking
Websites with proper ‘Access-Control-Allow-Origin’ were configured as follows:

— Approximately 71% configured to ‘*’
  ⇒ allows CORS from any website
— Approximately 29% configured ‘origin’
  ⇒ CORS is only allowed from the website specified by the origin header
Websites enabling Access-Control-Allow-Credentials were 22% out of all of the CORS websites.

Value set to ‘true’ is only valid. Value other than that operates as it is not set to ‘true’. Furthermore, there were several websites which value is set to ‘false’.
In case ‘true’ is specified on Access-Control-Allow-Credentials, the value ‘*’ cannot be used for Access-Control-Allow-Origin. However, 163 sites were configured to ‘*’.

Please check beforehand whether it operates as it is intended!!
Within the 68% of the websites for which the origin was set in Access-Control-Allow-Origin, ‘true’ was configured in Access-Control-Allow-Credentials.
If the response header returned is as shown above, JavaScript loaded from http://example.com will operate as same as the client browser.

⇒ When arbitrary JavaScript can be executed from a website specified in Access-Control-Allow-Origin, it may become very dangerous.

Then, is it safe if this vulnerability does not exist?
Mashup Example

A-C-Allow-Origin: siteA
A-C-Allow-Credentials: true

Mashup origin site

Mr. X’s image

A-C-Allow-Origin:siteB
A-C-Allow-Credentials: true

Mashup site B

In order to allow Cookie added CORS to each site, generate value in Access-Control-Allow-Origin dynamically.

Use favorite image as background.

Provide favorite image related products.
How can attacker leverage this?

A-C-Allow-Origin: http://evil.example.com
A-C-Allow-Credentials:true

Mashup origin site

Mr. X’s photo

evil web site (evil.examle.com)

Origin: http://evil.example.com

Cookie

send outside

Information Leakage

Mr. X
Cross Origin Request utilizing XHR2  Sample Advisory

XHR2 を突破できるケース

セキュリティの配慮がある XHR2 であっても、次のような条件が成立する場合にはリクエスト強要攻撃が成立し得る。

- クライアントから送られてきた Origin: origin リクエストヘッダに対して、サーバが必ず Access-Control-Allow-Origin: origin レスポンスヘッダを返すようになっているかつ、
- サーバが Access-Control-Allow-Credentials: true レスポンスヘッダを発行するとともに、Cookie を用いたセッション維持を行っている。

図8-16: XHR2 を突破できるケース

IPA Lectures on secure coding   Web Application course
194 sites inherited the value from the original origin header and configured in Access-Control-Allow-Origin header and sent it back!!

Data may be intercepted if the user authenticated web page is configured as aforementioned.

Furthermore, it may bypass CSRF token protection.
Password Change by CSRF

Issue a token

GET /passwd HTTP/1.1
Origin: http://evil.com

Cookie

Access-Control-Allow-Credentials: true

Token=2a8bc84e

Verify the token

POST /passwd HTTP/1.1
newpwd=infected&token=2a8bc84e
Origin: http://evil.com

Cookie

Obtain token by JavaScript

XHR

Obtain token by JavaScript

Java Script

Password Changed
Proper Configuration

- **Restriction on permitted page**
  - Access-Control-Allow-Origin and Access-Control-Allow-Credentials should not be added to a web page which is not intended to be open to the public. (i.e. Configuration Change Page, Password Change Page)

- **Verify origin header**
  - In case of allowing particular web sites, verify the value of the Origin and then only add Access-Control-Allow-Origin and Access-Control-Allow-Credentials as necessary.

GET /passwd HTTP/1.1
Origin: http://evil.com

Cookie

Access-Control-Allow-Credentials: true

token=2a8bc84e

JavaScript cannot access to token

Issue a token
Verify & Restriction

XHR

Java Script
Configuration examples

Restriction on permitted page

Access-Control-Allow-Origin and Access-Control-Allow-Credentials should not be added to a web page which is not intended to be open to the public.
(i.e. Configuration Change Page, Password Change Page)

Ex: Allow usage of the resource only by an image file

```xml
<?FilesMatch "\.(jpeg|gif|png|img)$">
  SetEnvIf Origin "^https?://.*$" ORIGIN=$0
  Header set Access-Control-Allow-Origin %{ORIGIN} env
</FilesMatch>
```

Verify the origin header

In case of allowing particular web sites, verify the value of the Origin and then only add Access-Control-Allow-Origin and Access-Control-Allow-Credentials as necessary.

Ex: Only allow requests from sub domains of example.com to use the resource

```bash
SetEnvIf Origin "^https?://.*\.(example\.|com)$" ORIGIN=$0
Header set Access-Control-Allow-Origin %{ORIGIN} env
```
Headers with Security Features
Most sites with X-XSS-Protection enabled were configured to ‘1’:mode=block (Enable protection filter, blank page was displayed upon detection)

514 websites configured to ‘0’ (disabled)

Attention!! Disabling the XSS Protection should be restricted to specific web pages such as a page having false-positive issues.
20,720 sites out of all sites were redirected to HTTPS.
1,754 sites configured HSTS.
Over half of the sites configured max-age to 1 year.
Attention!! If max-age is configured to ‘0’, this website will be deleted from HSTS list of the browser.
Conclusion
Conclusion

While HTML5 enhances browser capabilities, there are multiple architectural issues which we should be aware of upon use.

Please refer to “Technical research report on security issue of web applications utilizing HTML5” for developing secure web applications.

Any inquiries are welcome. Please contact to

JPCERT Coordination Center
Email: ww-info@jpcert.or.jp
Tel: +81-3-3518-4600
Web: https://www.jpcert.or.jp/
Appendix
Custom HTTP header

- X-Recruiting: We’re looking for talented people, join us: <URL>(We have cookies!)

- X-Recruiting: Like HTTP headers? Come write ours: <URL>

- x-poetry: Choose Life. Choose a job. Choose a career.

- X-<CompanyName>-jobs: you’re reading this … come work at xxx!

- Were-currently-looking-for-devs-like-you: Tweet @xxx for job details.
X-Want-A-Job-With-Us response header

X-Want-A-Job-With-Us: QlpoOTFBWskWSRfWskAAAoVgEAAzkJ3cwlABliZDQmU2lCmTeYoWgdWyrXBr3fsUaloRtl5IGIlpyj18dBoXckU4C5AkX1rJA==

? decode by Base64

Binary !!!
discem file type

output to a file

Compressed file…. :-c
extraction by bzip2

extraction

confirmation file type

# bunzip2 -v decoded
bunzip2: Can't guess original name for decoded -- using decoded.out
  decoded: done

# file decoded.out
decoded.out: ASCII text, with no line terminators

ASCII! Readable!!
finish?

```
# cat decoded.out
frac q@e na rznvy jvgu gur jbeq va vg#
```

can’t read … :-(

```
frac q@e na rznvy jvgu
```

rotate …
perl

```
# perl -ne 'split(/,\$\_); for \$i(1..25){print"\$i \"; foreach(\$\_){if(/[a-z]/){\$c=\$i+ord \$\_; \$c=-26 if(\$c>ord(\'z\')); print chr \$c;}else{print;}}}'' decoded.out
```

**ROT13!!**

Send an email with the word in it
Thank you for your attention