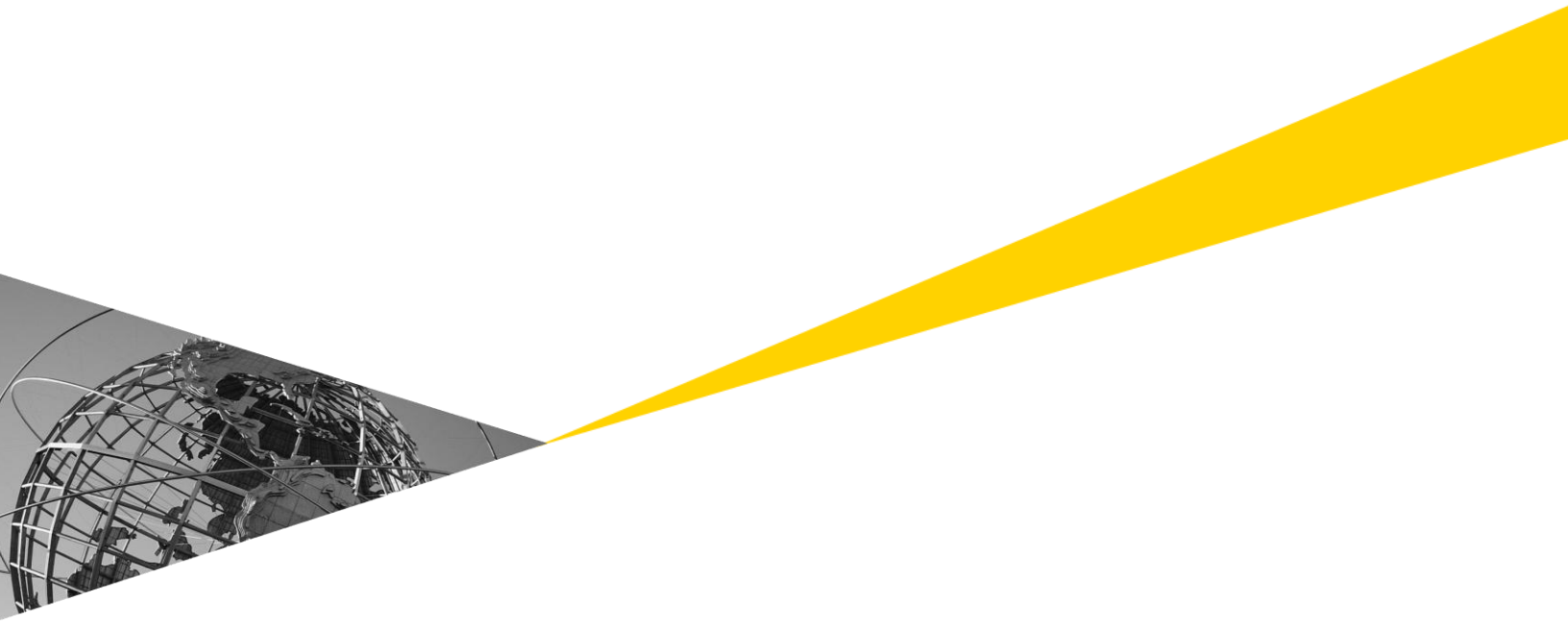


AJAX Hammer

Harnessing AJAX for Dynamic CSRF

Oren Ofer
Hacktics Advanced Security Center, E&Y

January 9th, 2012

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

ERNST & YOUNG

TABLE OF CONTENTS

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

1. Introduction

It started as a simple penetration test, which yielded common exposures… but in this case, the client was not willing to settle for a simple explanation as to why header verification is an incomplete solution to CSRF... This client **demanded** a fully working proof of concept which negates the header verification technique, in order to justify the development of the suggested anti-CSRF prevention mechanism - a request that led us to an interesting discovery.

The following paper presents new methods that can be used to harness AJAX for dynamic CSRF attacks (Cross-Site-Request-Forgery [i]), regardless of CORS implementations, and as opposed to today`s perceived limitations.

The methods presented can be used in-spite of the restrictive **Same Origin Policy**[ii] enforced by the browser on AJAX calls (assuming the right conditions are met), and have a greater impact than "static" CSRF attacks, since they also enable **content theft** and **response analysis**.

In addition, the whitepaper will prove that web applications cannot rely on certain technologies (JSON) or incomplete CSRF prevention mechanisms (such as custom header verification or partial of CSRF tokens), at least not anymore.

The various claims provided in this whitepaper will be backed by POC code and demonstration movies.

For a quick introduction to the attack, view the online **prezi presentation**, and the online **demonstration movie**.

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

2. Background – CSRF and AJAX

As penetration testers, we face a variety of security mechanisms in each test, and mechanisms that rely on server-side custom header verification, non-standard AJAX based content delivery method or a location-specific token verification to prevent CSRF attacks are not uncommon.
But before we begin to explain how to bypass these mechanisms using dynamic CSRF attacks that harness AJAX (which is supposed to be restricted due to the same origin policy), let's set a good foundation by explaining the basics:

2.1. Cross-Site Request Forgery (aka Session Riding)

As the security level of web sites and browsers improves, hackers find creative ways to achieve their goals - small loopholes within the browsers security rules.
One of these ways is the Cross-Site Request Forgery attack, abbreviated as CSRF or XSRF – an attack that enables malicious 3rd party websites to perform operation on behalf of users, by redirecting them to action-performing links in vulnerable web sites.
The following pre-conditions must be met for a successful CSRF attack:

- The victim must be authenticated to the target vulnerable web site.
- The authentication mechanism should be based on session identifiers stored in a cookie, HTTP authentication (BASIC, DIGEST or NTLM) or certificate based authentication.
- The victim must use the same browser instance, or use a browser instance that shares certificates and/or persistent cookies with the browser instance used for authentication.

After the authentication process, the browser "stores" the identifiers (session id, certificate or credentials) in the browser memory / hard drive, and associates them to the web site's domain. By default, any access to the domain will cause the browser to automatically send the identifiers to the server, thus, the user's identity will be "delegated" to the server.
Malicious websites can abuse this behavior and instruct the browsers of unsuspecting users to access action-performing URLs in external web sites, relying on existing authentication between the users and the target external web sites, potentially performing operations on user's behalf.
It is important to mention that CSRF attacks are considered "static" - a shot in the dark, since the attacking web sites are unable to directly analyze the response of this redirection.

2.2. AJAX and XMLHttpRequest

One way for making websites more dynamic is using "Asynchronous JavaScript And XML", or AJAX, in short. AJAX allows website pages to incorporate JavaScript dynamic objects, such as the XMLHttpRequest, that can interact with the website's host using the HTTP protocol.
By using XMLHttpRequest it is possible to affect the content within the webpage without refreshing the entire web page, and access the server with advanced content delivery methods such as XML and JSON. By avoiding the need to refresh the entire web page, users benefit of a faster response and website owners benefit from lesser bandwidth overloads.

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

2.3. Same Origin Policy on AJAX

The same origin policy dictates that an AJAX object's ability to fully communicate on the user's behalf is possible assuming the following conditions are met:

- The protocol used by the AJAX object must be identical to the protocol of the origin page.
- The target port of the AJAX object must be identical to the port of the origin page.
- The domain of the host and the domain of the AJAX object's target host must be identical (Sub domains are governed by slightly different restrictions which will not be covered in this whitepaper).

Confused? The example can be used to understand the various restrictions:
A user that surfs to a website that resides in http://abstractSite.com:5656, incorporates an XMLHttpRequest object which tries to send a request to the following address: http://www.fictionSite.com:5656. Assuming the target web site does not implement any permissive CORS policies, the browser will perform the following tests before allowing the AJAX object to communicate with the target web site:

- Same Protocol Verification – both the host and target websites protocols are using the HTTP protocol, thus, this check will pass.
- Same Port Verification - both the host and target websites are using the same TCP port (5656), thus, this check will pass.
- Same Domain Name – The XMLHttpRequest origin domain is "abstractSite.com", and the target domain is "www.fictionSite.com". Since the domains are different, and no CORS policy is set, this check will fail.

Clearly, the origin domain is different, and according to the W3C RFC[iv] , the browser ***should*** terminate this request. The reader probably wonders "why should?"... and the answer to this question lies in the impact of the scenario described in the following sections.

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

3. CSRF with AJAX Attack Methodology

3.1. CSRF with AJAX – What can it do, how is it possible?

Since an AJAX call (unlike a simple redirection) can analyze the content returned from a request, it can be used to enhance the capabilities of a "static" CSRF attack, and make dynamic. AJAX calls can enable malicious web sites to:

- Analyze the content returned and locate sensitive information.
- Locate anti-CSRF tokens in pages that precede a CSRF protected entry point.
- Dynamically locate the CSRF target entry points, instead of constructing the CSRF payload in advance.
- Overcome custom header requirements, and bypass incomplete CSRF prevention mechanisms.
- Perform CSRF on entry points that require JSON, XML or different content delivery methods.

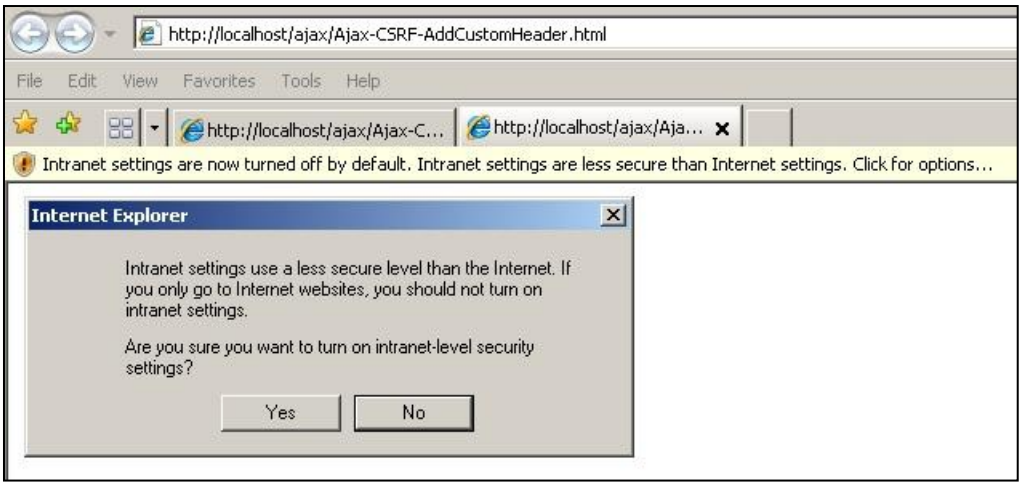
In order to generate a cross domain request on the users behalf while using an AJAX object, the following conditions must be met:

- Same Port – the malicious website and the vulnerable website reside on the same port.
- Same Protocol – the malicious website and the vulnerable website must use the same protocol.
- The victim must use a "**permissive browser**", meaning a browser that supports **permissive intranet settings** (a concept which will be described in the following section).
- The malicious web site must be perceived as "**Internal**" by the browser – the user should access the attacking web site while using an Intranet address. This perquisite can be achieved in multiple ways:
  - The attacker can to setup the malicious website in the **intranet** of the victim's network (assuming he is a legitimate or rouge member of that network).
  - The attacker can inject the AJAX scripts into a vulnerable third party application hosted in the internal network (Outlook Web Access, SharePoint, administrative interfaces, etc).
  - The attacker can lure the user to change the host's definitions for the malicious website or perform it himself using public infrastructures (internet-cafe shops, etc).

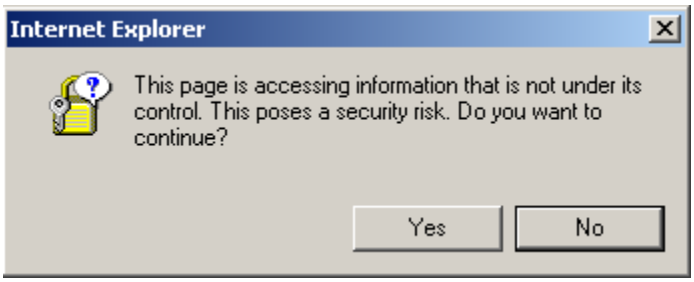
Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

3.2. Permissive Browsers

Browsers are shipped with pre-defined security zone settings which vary based on the location of the websites. For instance, if the user enabled intranet settings in the browser (initial popup confirmation or intentional definition), than the security levels for websites which are located in the Internet are stricter then the security level for websites which are located in the Intranet. One of the restrictions enforced through these settings is the same origin policy.
This restriction can be "bended" when the intranet settings are enabled in the browser (once):



The browser may permits cross domain requests which require a single user confirmation to a small notification (once per domain):



This scenario will occur in various browsers, such as Safari 4, Internet Explorer 6-8 and in an Internet Explorer 9 instance which is configured to allow the inclusion of all Intranet sites in the local security zone settings (and potentially, in other browsers as well).
But what is the probability that user will confirm the warning?
Researchers from the Carnegie Melon University conducted a research called the “Crying Wolf: An Empirical Study of SSL Warning Effectiveness”[iii]. Part of the research covered how many users ignore the following security warning that Internet Explorer 7 provides when they are surfing different websites.

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

The research results show that even though users are surfing an online banking website, **90%** of the users ignored the security warning (!). Although a similar empirical research was not performed on the security warning presented in the AJAX cross domain case, the cross domain notification is less obvious, and I strongly believe that the results would be similar and maybe higher due to the following reasons:

1. Title – the AJAX cross domain request warning title does not imply anything which is related to a security threat.
2. Common Behavior Pattern – The common user behavior includes “skimming”, a process in which users read the first line, and with the absence of an interesting "trigger", simply skip to the last line in the paragraph. Thus, users will usually read the “Do you want to continue?” sentence, and skip the middle sentence which declares this notification as a possible security risk.
3. Website Content – The aforementioned research shows that users tendency to ignore security warnings when they surf a website which appears innocent, and does not contain any important data of the users (for example, the library's website). For example, the research showed that in the library's website, **100**%(!) of the users ignored the security warning. This behavior pattern can be used by the attacker, by setting the CSRF performing website to look as an innocent website without sensitive information, lowering the users’ psychological awareness.

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

3.3. Demonstrating CSRF with AJAX

The following section demonstrates how attackers can create an innocent looking website, armed with a dynamic CSRF generating mechanism.

For examples that include **content theft**, **custom header additions** and **repayable parameter analysis** (VIEWSTATE, EVENTARGS and EVENTARGUMENT), please refer to the content presented in the 2012 local chapter meeting of OWASP IL: http://www.youtube.com/watch?v=JHJ1WW4Fcvw[vii] and http://prezi.com/6vnl6so07b-c/ajax-hammer/[vi].

In the following example, the malicious website is called “abstract”, and is located in the victim's intranet, while the simulated vulnerable target web site is named www.cnn.com.

The malicious website protocol and port are set accordingly to the target website, HTTP and port 80 respectively.

Step 1 - The user authenticates in front of the vulnerable website, which populates the session memory associated with the browser' cookie with the user identity and permissions.

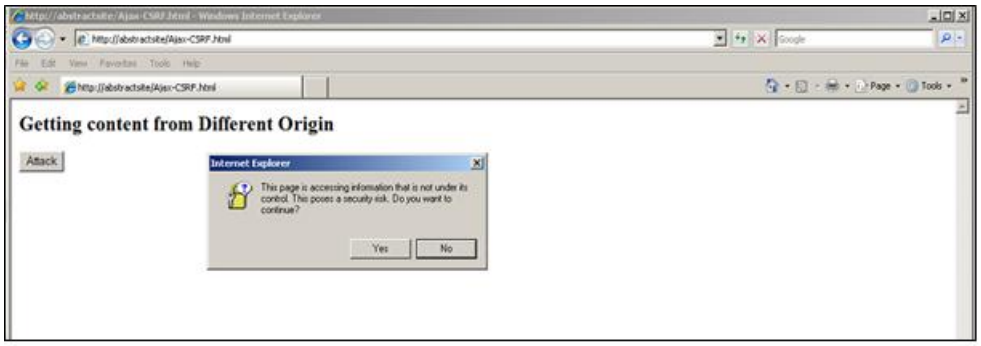
Step 2 – The authenticated user uses a second tab to surf to the malicious website http://absractSite/Ajax-CSRF.html. So far, the first and second phases are also the perquisite steps for a static CSRF attack.

The malicious web site contains the following AJAX-using function:

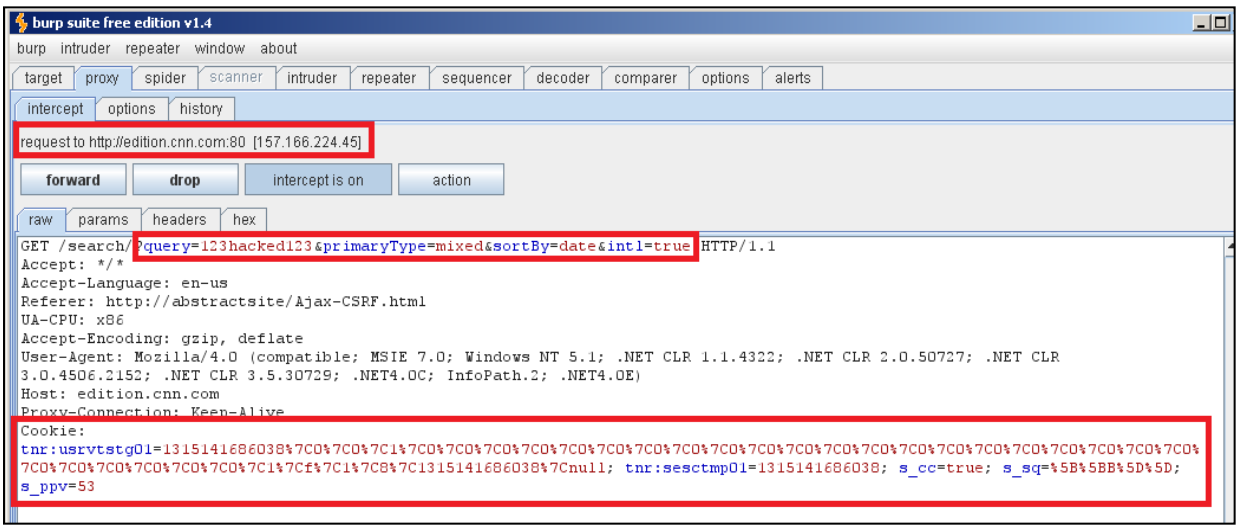
```
function csrfAjax()
{
if (window.XMLHttpRequest)
  {
     xmlhttp=new XMLHttpRequest();
  }
else
  {
     xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
       document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","http://edition.cnn.com/search/?query=123hacked123&primary
Type=mixed&sortBy=date&intl=true",true);
xmlhttp.send("");
}
```

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

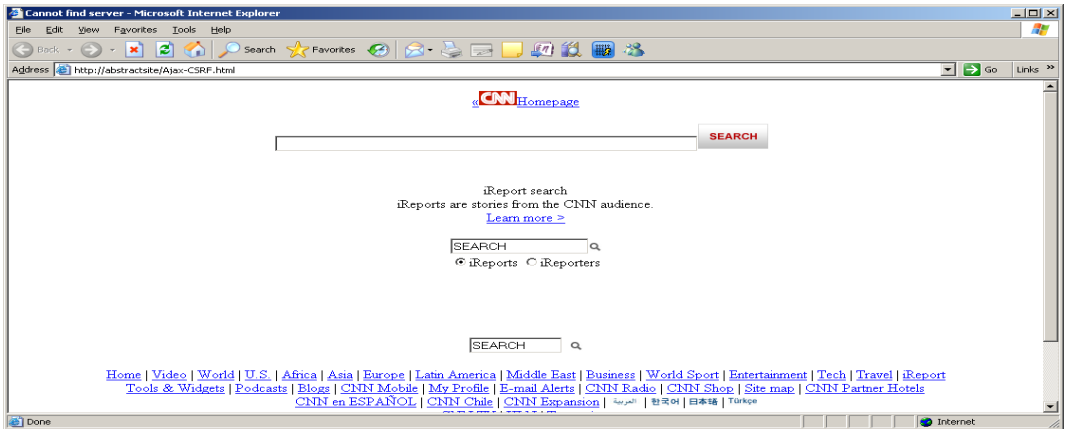
In the following screenshot it is possible to see the innocent looking popup:



Step 3 – The user clicks "yes" and the following request is generated (intercepted with a proxy):



The browser sends the request to the target while automatically adding the users` cookie, and thus, causes the victim to perform the action that the attacker intended. Now, the malicious website is able to analyze the response, and presents it under the malicious domain:



Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

3.4. Dynamic AJAX CSRF – CSRF in New Territories

To summarize how AJAX flexibility enhances the capabilities of current CSRF attacks, let's see what can be done it that wasn't there before.

Response Content Theft – The XMLHttpRequest object allows the malicious site to analyze the response using the "responseText" method. This capability enables new uses for CSRF – CSRF for information gathering, content theft and internal domain & IP enumeration (as long as the enumeration is mapping identical ports).

Direct Dynamic CSRF – Attackers are no longer bound to static CSRF attack scenarios, or to dynamic CSRF attack scenarios that rely on indirect information gathering[viii]. Using this AJAX based technique it is possible to dynamically identify the target URL, gather information from the response, and continue to the next step of the CSRF attack according to the information gathered:

- Locate anti-CSRF tokens in pages that precede a CSRF protected entry point.
- Replay obligatory dynamic fields such as VIEWSTATE, EVENTTARGET and EVENTARGUMENT.
- Dynamically locate the CSRF target entry points, instead of constructing the CSRF payload in advance.
- **Send notifications on the application structure to a remote attacker, creating an HTTP "command line" scenario**!

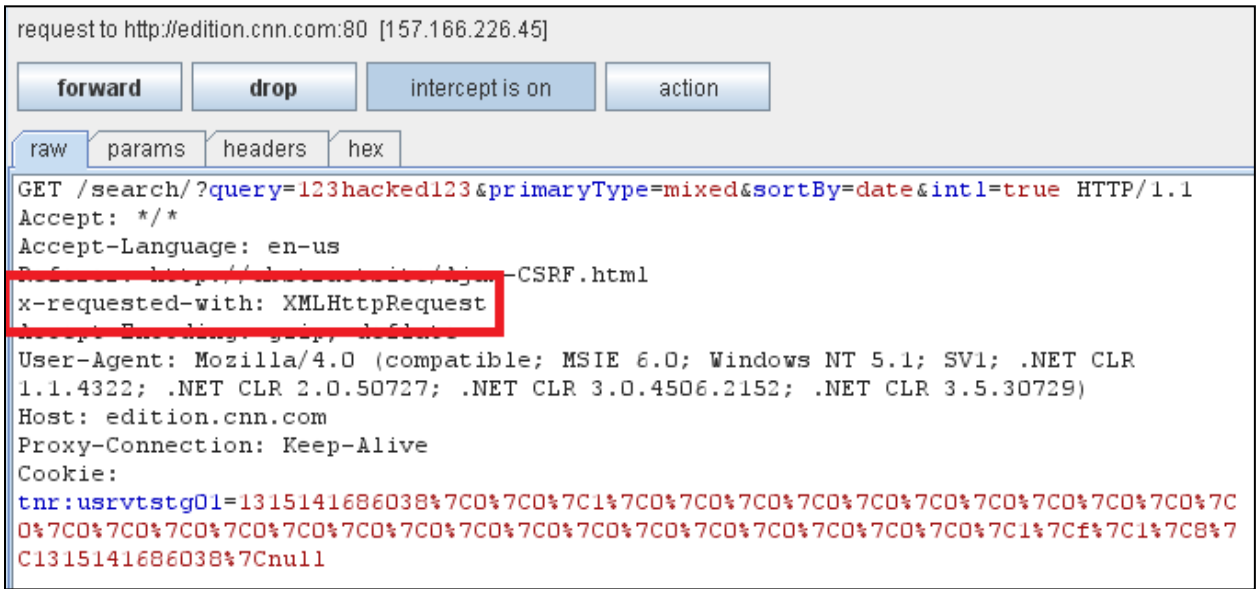
Server Side Custom Headers Verification – It is a common misconception that performing validation of custom headers on the server side can prevent CSRF attacks. By using the XMLHTTPRequest object, an attacker can add custom headers of his choosing. The following code segment, builds on the previous code sample, and adds a custom header:

```
function csrfAjax()
{
if (window.XMLHttpRequest)
  {
     xmlhttp=new XMLHttpRequest();
  }
else
  {
     xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
```

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

```
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
       document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","http://edition.cnn.com/search/?query=123hacked123&primary
Type=mixed&sortBy=date&intl=true",true);
xmlhttp.setRequestHeader("X-Requested-With", "XMLHttpRequest");
xmlhttp.send("");
}
```

In the following screenshot it is possible to see now that the custom header is added to the CSRF request:



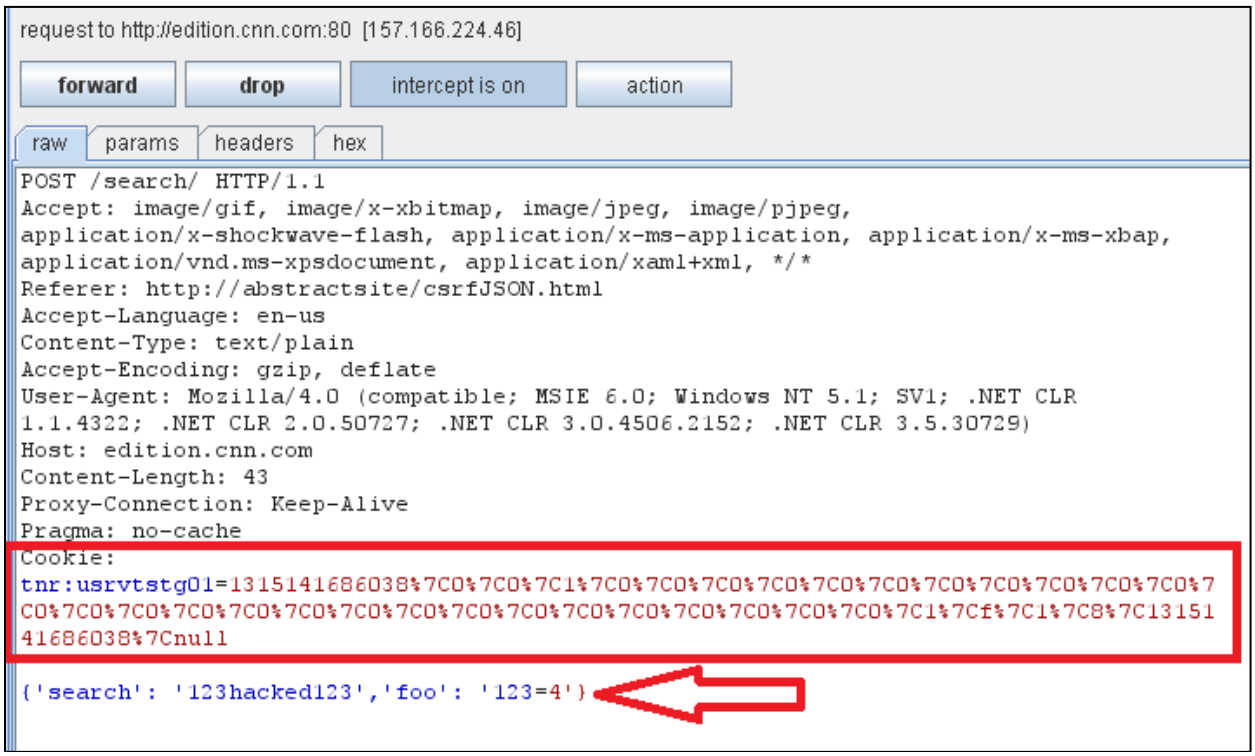
Special Formats – sometimes developers rely on complex formatted requests in attempt to foil CSRF attacks, include XML input, JSON, etc. Some of these formats can be forged using HTML forms, but with some inherent restrictions. For instance, generating JSON requests using an HTML form can be done in the following manner:

```
<html>
<head>
</head>
<body>
Welcome to my Hacker Site!</br>
<form enctype="text/plain" action="http://edition.cnn.com/search/"
method="post">
```

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

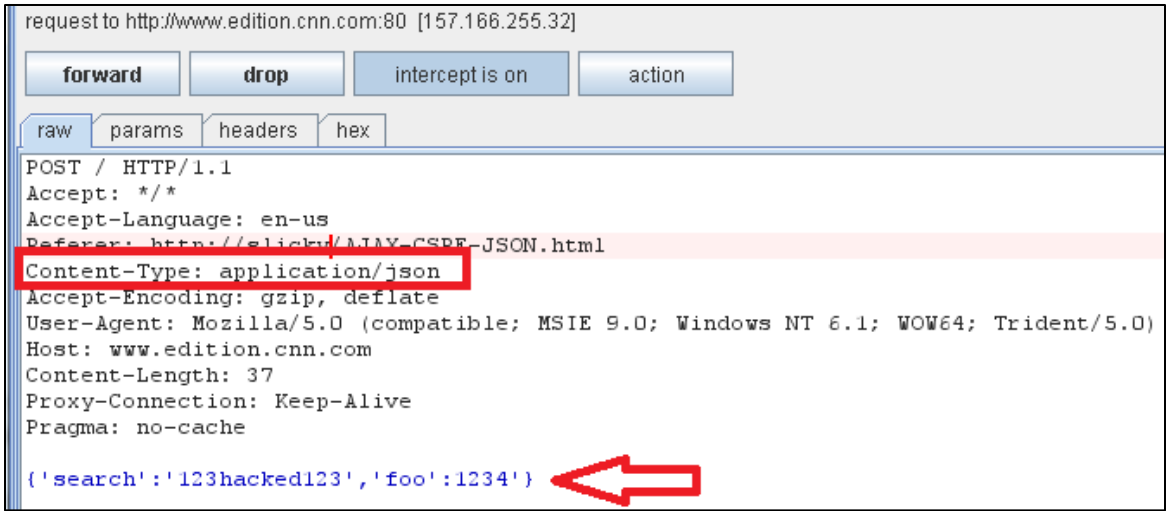
```
<input type="hidden" name="{'search': '123hacked123','foo': '123" value="4'}"
/>
</form>
<script>
document.forms[0].submit();
</script>
</body>
</html>
```

The following screenshot shows that a valid JSON request is generated by the user, and the cookie is automatically added to the request:



The method is limited due to two major reasons:
The **first** reason is inherent in the equal sign which is added as due to the structure of the HTML form “parameter = value” pair - thus, applications that perform input validation or enforce a strict JSON structure will detect the equal sign.
The **second** reason lies in the fact that by using form submission, attackers cannot dictate the request’s content type. Thus, applications that validate that the “Content-Type” header will see that it is not “application/json".
Generating the request using AJAX will overcome all these problems, as can be seen in the following screenshot:

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com



3.5. Additional Ideas for Future Research

It seems that further research is required in order to better assess the potential that lies in the usage of AJAX for CSRF attacks. Such research should encompass an empirical test of browsers from various vendors, multiple versions and even different platforms (i.e. mobile browsers and desktop browsers). The following subjects may provide additional leads:

- Enhancing the capabilities of Dynamic AJAX CSRF while using different HTTP Methods – tests should be conducted in order to determine how will web servers and application servers behave if the page is accessed with GET/HEAD/POST methods, instead of the original methods, similar to the case described in "Bypassing VBAAC With HTTP Verb Tampering" [v].
- Creating an AJAX CSRF notification system, similar to BeEF (XSS).

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

4. Mitigations

In order to mitigate AJAX based Dynamic CSRF, is it recommended to implement the following mitigations:

- Disable the intranet settings in all the organization browsers.
- Use an IPS/WAF to verify the "referer" header of incoming requests (since this is the only header that cannot be forged using this technique, at least using the information gathered in the current research).

Furthermore, applications should implement an efficient Anti-CSRF mechanism for preventing external sources from accessing internal pages on behalf of users. The feature should be implemented in the following manner:

- The proposed access verifier mechanism (a.k.a Anti-CSRF mechanism) should generate a designated random token for each user that undergoes a successful login process (in addition to the session identifier, and preferably in the same module that verifies the login request), store this token in the user-associated server side session memory, and ***append*** this value to a designated HTTP parameter in every link, AJAX request, redirection instruction or HTML form embedded into the content presented to the user.
  Note - Redirecting modules are modules that **must** receive the token as a POST parameter, and not in GET, in order to prevent the token from being disclosed to third party web sites in the "referer" header.
- The token should be long (at least 16 characters), random (generated using a random seed, such as the size of the log file), and consist of at least 3 types of characters (letters, capital letters and numbers).
- Each internal page in the application (apart from the login page) must verify the existence of this random value and make sure it is the same value issued for the user, by comparing it to the value the stored in the user's server side session memory; the verification must be performed immediately after verifying the user's authentication and authorization level (in case this verification is not performed in any global module).
- It is recommended to perform the token verification in a global module, such as a filter or a global event.

It is important to mention that the random value should be sent as either a GET or POST parameter (apart from redirecting modules), and NOT within the cookie (which will be sent in every request to the server, alongside the value). This will prevent an attacker from accessing internal pages in the application, since he won't be able to figure what is the currently expected random variable value (at least using static CSRF, and in pages that require token for access, neither with dynamic CSRF).

External Implementations

The OWASP foundation implemented an external module called CSRFGuard, which can be utilized to implement the required mechanism. Additional information can be found in the following address:
http://www.owasp.org/index.php/Category:OWASP_CSRFGuard_Project

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

5. Credits

5.1. Bibliography

i. https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)
ii. http://www.w3.org/Security/wiki/Same_Origin_Policy
iii. http://lorrie.cranor.org/pubs/sslwarnings.pdf
iv. http://www.w3.org/TR/XMLHttpRequest/#the-open-method
v. http://dl.packetstormsecurity.net/papers/web/Bypassing_VBAAC_with_HTTP_Verb_Tampering.pdf
vi. http://prezi.com/6vnl6so07b-c/ajax-hammer/
vii. http://www.youtube.com/watch?v=JHJ1WW4Fcvw
viii. http://www.blackhat.com/presentations/bh-usa-09/HAMIEL/BHUSA09-Hamiel-DynamicCSRF-PAPER.pdf

5.2. Additional Contributors

I would like to thank the following individuals for their contribution to the release of this white paper:

- **Oran Hollander and Eran Tamari –** for believing in AJAX.
- **Shay Chen –** for suffering in the ugly QA & publication process.
- **Oren Hafif and Niv Sela –** QA and additional fine tuning.
- **Genady Podgaetsky –** for insisting, and setting high standards.
- **Snir Karat –** for covering the major desktop's Permissive Browsers

Ernst & Young LLP
1401 McKinney Street, Suite 1200
Houston, TX 77010
Tel: +1 (713) 750 1200
www.ey.com

6. About Ernst & Young

Ernst & Young
Assurance | Tax | Transactions | Advisory

About Ernst & Young

Ernst & Young is a global leader in assurance, tax, transaction, and advisory services. Worldwide, our 144,000 people are united by our shared values and an unwavering commitment to quality. We make a difference by helping our people, our clients, and our wider communities achieve their potential.

For more information, please visit www.ey.com.

Ernst & Young refers to the global organization of member firms of Ernst & Young Global Limited, each of which is a separate legal entity. Ernst & Young Global Limited, a UK company limited by guarantee, does not provide services to clients.

The Ernst & Young organization is divided into five geographic areas and firms may be members of the following entities: Ernst & Young Americas LLC, Ernst & Young EMEIA Limited, Ernst & Young Far East Area Limited and Ernst & Young Oceania Limited.

About Ernst & Young's Advisory Services

The relationship between risk and performance improvement is an increasingly complex and central business challenge, with business performance directly connected to the recognition and effective management of risk. Whether your focus is on business transformation or sustaining achievement, having the right advisors on your side can make all the difference. Our 18,000 advisory professionals form one of the broadest global advisory networks of any professional organization, delivering seasoned multidisciplinary teams that work with our clients to deliver a powerful and superior client experience. We use proven, integrated methodologies to help you achieve your strategic priorities and make improvements that are sustainable for the longer term. We understand that to achieve your potential as an organization, you require services that respond to your specific issues, so we bring our broad sector experience and deep subject matter knowledge to bear in a proactive and objective way. Above all, we are committed to measuring the gains and identifying where the strategy is delivering the value your business needs. This is how Ernst & Young makes a difference.