Finding and Preventing Cross-Site Request Forgery

Tom Gallagher

Security Test Lead, Microsoft

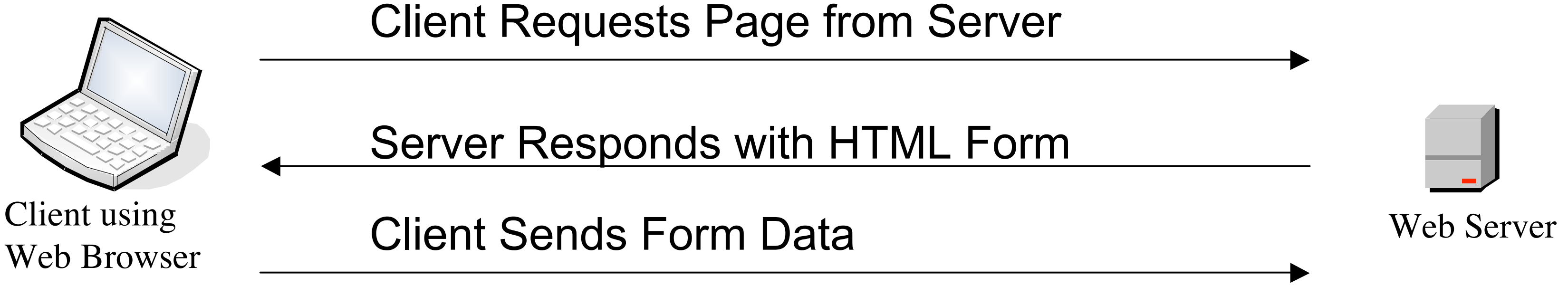
Agenda

- Quick reminder of how HTML forms work
- How cross-site request forgery (CSRF) attack works
- Obstacles and how attackers work around them
- Demo of attack
- Common proposals for prevention
- Demo of detected attack
- How to pen-test the prevention mechanism
- Built-in features to prevent attacks
- Impact on SOAP
- Automated testing

Why is CSRF interesting?

- Allows an attacker to take arbitrary actions as the victim against a web site.
- Similar to cross-site scripting
- Often missed by web pen-testers
- Not a small coding error like XSS
- Many platforms do not have built-in features to prevent the attack
- Preventing the attack may require implementing a new feature and often isn't trivial to do right
- Can apply to SOAP

Refresher on HTML Forms


Client Requests Page from Server
Server Responds with HTML Form
Client Sends Form Data
Client using Web Browser
Web Server

User completes HTML form and submits it

Server authenticates and authorizes user. If success, performs requested action.

Okay, what's the security problem?
Only authorized users can perform actions.

Reminder: Browser Security Model

- Browsers prevent cross-domain read access to data.
  - Example: http://www.contoso.com cannot read http://www.microsoft.com
- Cross-domain form submissions are allowed.
- CSRF Attack - Attacker coerces victim to submit attacker's form data to the victim's web server. (Cross-site form submission)

Anatomy of CSRF Attack

- Step 1: Attacker hosts web page with pre-populated HTML form data.
- Step2: Victim browses to attacker’s HTML form.
- Step 3: Page automatically submits pre-populated form data to a site where victim has access.
  - Remember: Javascript can automate posting forms.
- Step 4: Site authenticates request (with attacker’s form data) as coming from the victim.
- Result: Attacker’s form data is accepted by server since it was sent from legitimate user.

Demo of Attack

Obstacles for Attacker

- Needs to know victim's server
  - Knowing victim's server is not hard in a targeted attack or a commonly used server. Example: Famous banks, auction sites, etc.
- Needs to get victim to browse to attacker's site (pre-populated form)
  - Getting victim to load the attacker's form isn't hard. (Phishing is often successful.)
- Needs victim to log into server
  - Victim might already be logged into a site or might have automatic log-in enabled.
    - Examples: SSO, cookie, or Windows Integrated Authentication.
    - Windows Integrated Authentication is very popular on intranets.

Easier for GET Operations

- RFC 2616 (HTTP 1.1) states, "…the GET and HEAD methods SHOULD NOT have the significance of taking an action other than retrieval."
- In practice this is not followed. Many GET operations perform operations that do things besides/in addition to data retrieval.
  - Example: http://server/DeleteMessage.asp?ID=1
- Even easier to attack than POST, because victim can be attacked any place a URL is evaluated.
  - Example: Picture in email.

Common Prevention Ideas

- Check HTTP Referer
  - For privacy, Referer might not be present.
  - Redirects on the site might allow for correct Referer even if only redirecting to it's own site.
    - Example: http://server/redir.aspx?url=/delete.aspx?id=100
- Store state
  - When user browses to the form, record state, check it when it is submitted.
    - Examples: Server-side state or cookies (Attacker cannot set cookie for another user on victim's site without another security bug.)
  - Still vulnerable! Attacker can force the victim's browser to load the form from the trusted site and then submit the form from the attacker's site. (State will be correctly set.)
- Hidden HTML form field storing state
  - This works if done correctly.
  - Let's look at how to test for correctness.

Demo of Attack Detection

Pen-testing CSRF Validation Fields

Test 1: Verify validation field is unique for each user.

- Developers sometimes believe if the validation field is only good for a few minutes it’s good enough. Remember, an attacker’s pre-populated HTML form could use server-side code to grab a validation field on the fly.

Pen-testing CSRF Validation Fields

Test 2: Verify the validation field cannot be determined by other users.

- If the attacker can create a correct value of the validation field for another user, there is no value in the validation field.
- Validation field should also be unique for each site.
  - Example: If a hosting company deploys a web site for contoso.com and wingtiptoys.com, the validation field for user1 on one site should not work on the other. Especially bad if accounts can have different owners.
  - Even worse if problem present in commercial off the self software.

Pen-testing CSRF Validation Fields

Test 3: Verify the validation field is never sent on the query string.

- This data could be leaked in places like the HTTP Referer to the attacker.

- Also remember only data retrieval should occur through GET requests.

Pen-testing CSRF Validation Fields

Test 4: Verify request fails if validation field is missing.

- Nulling out data often bypasses checks.
- Validation field may only be checked on certain operations.
  - Example: During threat modeling, a team only planned to check validation field on database updates. They also send mail as the user but didn't plan on checking there!

ASP.NET Built-in Solution

- ASP.NET 1.1 introduced the ViewStateUserKey property.
- ViewStateUserKey can be set with user specific information.
- Information used to create unique __VIEWSTATE field.
- ASP.NET automatically checks when page is submitted
- Almost painless for developers.
- Other languages like C++, PHP, Classic ASP, do not currently have built-in support.

Impact on SOAP

- SOAP request are POSTs with XML contents
- Client should prevent cross-domain SOAP requests
- SOAP requests should be strictly validated so sending SOAP XML as form data should fail.
- This generally means SOAP doesn't have CSRF issues.
- However, SOAP 1.2 may allow it through both POST and GET…

GET/POST Binding

- SOAP method can be called using an HTTP GET or HTTP POST (normal form payload – not XML payload).
- SOAP 1.2 spec includes GET binding
- WSDL 1.1 spec includes GET and POST binding.

SOAP: HTTP GET Binding

- Usually takes 1 of 2 formats
- Format 1:

SOAP Method Name

http://server/auction.asmx/bid?id=5089&value=1000

Parameter Name

Parameter Value

- Format 2:

SOAP Method Name

http://server/auction.asmx?method=bid&id=8&value=10

Parameter Name

Parameter Value

SOAP: HTTP POST Binding

- Same data as HTTP GET except form data is send in POST payload instead of query string.
- Both POST and GET can often be disabled.
  - Example: ASP.NET contains 2 properties HttpPost and HttpGet to toggle this. Disabled by default.
  - Often enabled for debugging, and mistakenly left on.
  - Sometimes only enabled for localhost. (Still attackable but more difficult.)

Automated Testing

- Only interested in authenticated requests that perform actions besides data retrieval.
- Testing normal HTML forms:
  - Write code to find forms that are missing the known hidden validation forms.
    - Example: __VIEWSTATE
  - Existing functionality testing automation can be leveraged. Write code to hook the onbeforesubmit event and modify data accordingly.
  - Useful to have multiple accounts when testing.
- Testing SOAP GET/POST Binding
  - Local access - investigate server config (example: web.config for HttpGet/HttpPost setting)
  - Local access – If config allows GET/POST, examine methods for non-retrieval operations.
  - Remote access – Visit WSDL and try forming HTTP GET/POST requests based on information exposed. Success should return the same data as an XML SOAP request.

Questions?

- Related attacks covered in Hunting Security Bugs from Microsoft Press - http://www.microsoft.com/MSPress/books/8485.asp.
- Pick up a copy. ☺