GridPP
UK Computing for Particle Physics

Preventing CSRF attacks with GridSite

Andrew McNab

University of Manchester

MANCHESTER 1824

Outline

- How website logins work
- CSRF attacks
- How this affects “grid websites”
- GridSite solutions
- Other benefits from this

How the web works

- A web browser (Firefox) connects to a web server (Apache) and says:

  GET /hellopage.html HTTP/1.1

- The server replies:

  HTTP/1.1 200 OK

  Date: Thu, 25 Oct 2007 14:30:00 GMT

  Content-Type: text/html

  Hello!

How forms work

- Say we want to tell the server something, rather than just get a page:

  POST /helloprogram.cgi HTTP/1.1

  firstname=andrew&surname=mcnab

- The server might reply:

  HTTP/1.1 200 OK

  Content-Type: text/html

  Hello andrew mcnab!

How logins work

- We now want to login, so server knows it's really me:

  POST /loginprogram.cgi HTTP/1.1

  username=andrew&password=topsecret

- The server might reply:

  HTTP/1.1 200 OK

  Content-Type: text/html

  Set-Cookie: sessionid=1234567890

  Welcome back andrew!

Cookies

- The “cookie” I got back is a name-value pair chosen by the server, which is stored in my web browser.
- My browser remembers which website it's from.
- When I look at pages from that site, my browser mentions the cookie in case it matters:

  GET /anotherpage.cgi HTTP/1.1

  Cookie: sessionid=1234567890

- The server can show me extra options as a result.

Cookie security

- Cookie session IDs are either random numbers stored by the server, or cryptographic hashes of other info that only the server could have calculated.
- So trying to steal or make use of cookies is one of the major objectives of “front door” attacks on websites.
- Browsers implement a “Same Origin Policy” for cookies, to stop rogue websites reading cookies from other websites
  - This is applied to Javascript running in the browser too

CSRF

- CSRF is Cross Site Request Forgery
- For example, a direct link, button etc to the victim website's script that “does something”
- It relies on the victim user **already having a login cookie**, and the attacker knowing how the victim website works.
- Since some websites will “do” things with a link rather than a button, you can even embed an action link as an image in another website, or HTML email

Preventing CSRF

- From the user's side, there isn't much you can do about CSRF: you click the fake button on the attacker website, whoosh! you're on the victim website, the damage is done
- There are techniques, like double submit cookies, that developers can use to protect against CSRF
- **But in practice, CSRF quite difficult to set up because of hidden session information, and good (95%+) work-arounds like checking "Referer:" exist for other cases**

XMLHttpRequest

- Javascript is a C-like language that can be embedded in HTML web pages.
- XMLHttpRequest() is a Javascript function that can be used to fetch files over HTML by code embedded inside another page, without having to display them.
- Interactive websites like Google Maps and Gmail rely on XMLHTTPRequest: so it's good?
- XMLHTTPRequest can do CSRF **silently**: so it's bad too!

XMLHttpRequest

- When XMLHttpRequest was invented, people vaguely realised something bad might be possible.
- So browsers follow a similar “Same Origin Policy” as they do for cookies: only the original website that delivered the Javascript can be contacted by XMLHttpRequest.
- Unfortunately, Internet Explorer “enforces” this by putting up a rather weakly worded warning that you can click to ignore.
  - If you say yes, then the CSRF attack can proceed.

CSRF and Grids

- Many Grid project websites use X.509 user certificates
- These are equivalent to cookies that last a year
- So now the CSRF attacker doesn't even have to be lucky and pick a time when you happen to be logged in
  - You're always logged in!
- We tried to get away from passwords because of “phishing” attacks with faked-up websites: CSRF is almost the equivalent for X.509

GridSite

- GridSite consists of
  - A grid security toolkit for C/C++
    - Parses grid security objects, like GACL policies, X.509, GSI, VOMS credentials
  - An Apache module which adds support for these credentials
    - This lets people host webservices for Grids, written in C/C++/scripts/Java etc etc.

GridSite CSRF solution

- Remember the same origin policy used with cookies:
  - Only the original website pages can “see” its cookies
- When people arrive with an X.509 user certificate, GridSite 1.6 now gives them a cookie and relies on that rather than the certificate itself
- When they submit a form to “do” something, they must include the value of this “double submit cookie” in the form as well as the HTTP request (automatically via Javascript)

What this means

- This procedure cannot be faked with a button on the attacker's website (they cannot discover the cookie.)
- An XMLHttpRequest from Javascript on the attacker's site will fail in Firefox etc or produce that warning in Internet Explorer.
- This all reuses the passcode cookies developed by the GridSite Project for the GridHTTP protocol
  - Introduces lots of new positives too

How else to use it

- Since cookies rather than X.509 certificates are what really matters, can now login via other methods.
  - eg use Shibboleth username/password to access rights normally associated with an X.509 name
  - Use Kerberos login to as mcnab@hep.man.ac.uk rather than /C=UK/.../CN=andrew mcnab
- This integrates with GridSite 1.5.x support for non-X.509 credential types, and with existing GridHTTP

Attackers at home

- What if the attacker is closer to home, say with write access to your website?
  - maybe with a stolen credential that they want to escalate
- Cookies can be restricted to only some zones of website
  - eg with path=/securepages/
  - code on page in /userpages/... cannot read them.
- But Javascript XMLHttpRequest is still allowed since it's the same website, so attacker can steal a “securepages” cookie

Zones

- GridSite solution is allow login page to be on another virtual server: login.www.example.com for www.example.com
- Cookies are issued for “domain” www.example.com and the desired zone of website, eg with path=/securepages/
- XMLHttpRequest cannot be used to get round this since login.www.example.com and www.example.com count as different origins.
- **All done with a single-login click for a user with X.509**.

GridSite API

- GRIDHTTP_PASSCODE is name of cookie
  - Generated by mod_gridsite and saved in sessions store on server side
  - Or can be created by CGI/PHP/JSP login pages
- GRST_PASSCODE_COOKIE exported to CGI etc
  - Makes it easy to implement double-submit cookie checking
- GridSiteZoneSlashes directive in httpd.conf
  - Used to create zones based on directory hierarchy

Summary

- There are a set of CSRF attacks from the general web world which may become relevant to the grid world
- The existing double submit cookie solution and GridSite's cross-site login procedure defeat this class of attacks
  - even from within the victim website itself!
- We've implemented this in GridSite 1.6 in a way which other developers can build on