OWASP Meeting
May 25, 2006
San Antonio, TX
Dan Ross

Forceful Browsing

Shrek: Ogres are like onions.
Donkey: They stink?
Shrek: Yes. No.
Donkey: Oh, they make you cry.
Shrek: No.
Donkey: Oh, you leave em out in the sun, they get all brown, start sproutin' little white hairs.
(Shrek sighs)
Donkey: Oh, you both have layers.
(Pause)
Donkey: You know, not everybody likes onions.

Introduction

- April 2003: Al-Qaeda supporters hack into student's Web site... buried within a folder in his personal Web site... http://seclists.org/lists/isn/2003/Apr/0009.html
- December 2003: For months, access to a massive database of police files was available to anyone with a rudimentary knowledge of computers and an Internet link, according to a man who said he looked up files on the system several times... The man said he accessed the system by simply adding the words "PersonSearch/PersonSearch.asp" to the end of the link's normal Web address, http://www.mjno.state.mn. http://lists.jammed.com/ISN/2003/12/0014.html
- February 2004: Lee Gomes of the Wall Street Journal recently exposed this flaw at Gateway Computer. As Gomes reported, users could change their cookies and, when they went back to www.gateway.com, they were greeted with "Welcome back, x" and were able to access x's credit card information, address the works. Change the cookie again and you would be 'recognized' as someone else!
http://snafu.mit.edu/~fubob/pubs/wsj-gomes2.txt

- February 2004: A remote attacker could send a specially-crafted HTTP request containing hexadecimal URL encoded "dot dot" sequences (..%5c) to traverse directories and view any file on the Web server.
  http://www.securityfocus.com/bid/9733
- February 2005: Online payroll service provider PayMaxx shuttered its automated W-2 site on Wednesday after a researcher claimed that two security holes had exposed data on more than 25,000 people. PayMaxx's database contained a record for testing purposes that contained a Social Security number of 000-00-0000 and a password of all zeros. That could allow anyone to log into the site and then use the lack of authentication to sequentially download all the W-2 forms
  http://news.com.com/Payroll_site_closes_on_security_worries/2100-1029_3-5587859.html
- March 2005: Harvard MBA applicants rejected after "hacking" to check the status of their applications at ApplyYourself.com.
  http://www.businessweek.com/bschools/content/mar2005/bs2005039_7827_bs001.htm
  http://news-service.stanford.edu/news/2005/march9/hacker-030405.html
- July 2005: "Once I came across a vulnerability in an online auction application. When you put in a bad password, it locks you out of the system. I put in a bad password for a competing bidder, and it locked him out of the bidding," Grossman explains. (White Hat Technologies)
  http://www.devsource.com/article2/0,1895,1837351,00.asp
- Unconfirmed: A major record label recently released a "sneak preview"-the first song of an upcoming album with a URL ending in /track1.mp3. Obviously it wasn't long before the young audience realized that if they simply typed in "track2.mp3" or "track3.mp3" they could access the entire album. http://www.f5.com/communication/articles/2005/article030905.html
- May 2006: Regular visitors to www.carriagehouseglass.com could never see the hidden material, specialists said. Only visitors who knew the address of the pages inside could access the cache of downloadable Arabic writings...
  http://www.boston.com/business/articles/2006/05/07/
  ...crafts_website_hacked_by_terrorists/?rss_id=Boston_Globe_--_Business_News

Definition

- The OWASP Top Ten lists "Broken Access Control" as #2:
  "A2.5 How to Protect Yourself
  Forced Browsing Past Access Control Checks - many sites require users to pass certain checks before being granted access to certain URLs that are typically 'deeper' down in the site. These checks must not be bypassable by a user that simply skips over the page with the security check."
- Forceful browsing is an attempt to access a Resource that was not intended to be.
- Forceful Browsing is editing the url in a browser to gain access to files that the owner of the site didn't intend to be publicly accessible.
- Forceful browsing is the act of directly accessing pages (URL's) without consideration for its context within an application.
- Simply put, FB is an attempt to access a resource that was not intended or should not be permitted.

Demonstration

- www.google.com
- telnet www.google.com 80

- httpwatch
  http://www.simtec.ltd.uk/images/ebay.png

Why is FB dangerous?

- Hackers could gain access to sensitive files.
- Users may attempt to bypass application flow.
  Eg. consider "customers/action.php?a=d&customer=1"
  SQL = DELETE FROM customers WHERE customer='1';
  Guess = employees/action.php?a=d&employee=1
- Users may attempt to bypass application flow.
- SQL Injection is a form of FB.
  customers/action.php?a=d&customer=1
  YIELDS:
  DELETE FROM customers WHERE customer='1';
  vs
  customers/action.php?a=d&customer=x'; DROP TABLE customers; --
  YIELDS:
  DELETE FROM customers WHERE customer='x'; DROP TABLE customers; --';
- Session cookies are vulnerable as well.
- register_globals

Prevention

- Lock down your network:
  access control lists
  firewall
  Understand and justify every open service.
- Lock down your server:
  Understand and justify every open port.
  Scan (nmap)
  Use software firewall (eg iptables).
  doormand
  Beware/reconsider VPN.
  Explore secure alternatives to vulnerable services.
- Keep all components up-to-date:
  router IOS
  firewall
  OS
  Apache
- Judicious use of https:
  Understand the benefit.
  We insist on https exclusively.
- Use filename extensions to control access:
  eg Apache: do NOT deliver *.sql
- Short lifespan for inactivity AND finite lifespan for session cookies.
- Obscurity.
  Def: "the quality of being unclear or abstruse and hard to understand"
  What gain is there to you/anyone to advertise what O.S., webserver version, php/asp/etc?

```
# telnet www.ebay.com 80
...
Server: Microsoft-IIS/5.0
...

# telnet www.ask.com 80
...
Server: Apache/2.0.53 (Unix)
...
```

Our Application

- Allow Dealers to view their current discounts:
  mydiscounts.php?dealer=1
- In this particular case, the solution is:
  - Given their session, we already know who they are.
  - Knowing who they are, we don't need "?dealer=1".
  - Rewrite the page to pull their discounts given the session ID.
- Lessons learned:
  1. Authentication and Authorization must take place at every page.
  2. Don't assume that authorized users, once logged in, are going to behave!
  3. Repairing security is expensive. It must be built-in from the beginning.
  4. Safety/security records were made to be broken. You are only as good as your record. Thus, a very real danger of an insecure application is the cost to repair your damaged reputation.

Sanity Break

- Heard of BOFH?
  http://bofh.ntk.net/Bastard.html
- "Quiz ONE...
  5. The Adage "Separate the Problem from the Cause" in computing terms means:
  A. Diagnose the actual problem as opposed to it's effects
  B. Prevent Faulty Software from further compromising data
  C. Lock the problem down to a specific set of events
  D. Lock the User's Keyboard in the Tape Safe.
  E. Lock the User in the Tape Safe

Snack

- "What about all the other pages in our application? I don't want them there if I didn't give them a link to get there."
- HTTP_REFERER does not work. It is optional for browsers to send this. It is also spoofable.
- The solution is to supply a unique code for each link (a "snack"), then check that code when they get to the linked page.
- Consider the earlier URLs:
  customers/action.php?a=d&customer=1
  employees/action.php?a=d&employee=1
- With a snack, these become:
  customers/action.php?a=d&customer=1&snack=ee0afaea858c83832ddff334d3ed16ab
  employees/action.php?a=d&employee=1&snack=d9b788182729da6f4d2fcb781fd4a2ba

So, What is a Snack?

- First, let's talk about sessions:
  Here's how we do it:
  1. When you arrive at the login page, you receive a session cookie.
  2. The session cookie has a lifetime of 0, which means:
  (a) It's not written to disk - it stays in memory only. Remember users will access from Kinko's, their hotel room, their buddy's PC, etc.
  (b) It's restricted to this window (and its children) only.
  (c) When the user logs out, the cookie is deleted.
- Second, let's talk about md5:
  An md5 is essentially a one-way encryption of a string to a 32-character string.
  Eg. md5("hello world") = "5eb63bbbe01eeed093cb22bb8f5acdc3"
  md5("Hello world") = "f0ef7081e1539ac00ef5b761b4fb01b3" HOWEVER, Per wikipedia (http://en.wikipedia.org/wiki/Md5):
  ...On 18 March 2006, Vlastimil Klima published an algorithm that can find a collision within one minute on a single notebook computer, using a method he calls tunneling.
- So much for this class ;)
- The snack is an md5 of the user's session PLUS a keyword such as "customer", "employee", or EVEN "customer=1", "employee=1".
  Given that a user gets a different cookie each time they access the application, if we base the snack on the cookie, then every time they login, the snack for a particular page is different.
- Function to generate the snack value:

```
function snacker($value)
{
   /*
   By including the COOKIE in the snacker,
   AND by setting the COOKIE's lifespan to the life of the client browser window,
   the snacker changes every day!
   :)
   */
   $tmp = "nope";
   if (isset($_COOKIE["Session"])) $tmp = $_COOKIE["Session"];
   return md5($value.$tmp);
}
```

  Then printing the link becomes:

```
$url = "customers/action.php?a=d&customer=1&snack=".snacker("customer=1");
```

- Function to check the snack value:

```
function snackShouldBe($s)
{
   if (!isset($_GET["snack"]) || $_GET["snack"]!=snacker($s))
      {
      if (!isset($_GET["snack"])) $_GET["snack"]="";
      $m="Server: ".$_SERVER["SERVER_NAME"]."\n".
         "Remote IP: ".remoteIP()."\n".
         "Session: ".$_COOKIE["Session"]."\n".
         "User #: ...\n".
         "User Name: ...\n".
```

```
        "Requested URL: ".$_SERVER["REQUEST_URI"]."\n".
        "Snack Attack:  expected [$s]->[".snacker($s)."], got [".$_GET["snack"]."]";
    // send email(?)
    error_log($m);
    die();
    }
}
```

Now checking the link becomes:

```
snackShouldBe("customer=".$_GET["customer"]);
```

- What happens if the user (or application!) obtains the wrong answer at "snackShouldBe"?
  A ***"snack attack"*** of course!
  What do you do if you get a snack attack?
  - Email engineering(?) It could after all be a bug.
  - Deliver NOTHING to the browser. (exit;)
  - How you react is up to your implementation of snackShouldBe.
- Basically, it's a password for every page -- dependent on param's
  - conclusion: just another layer/technique
  - _GET vs _POST: we always use _GET.
  - does/can it help prevent SQL Injection?

Preventative Measures

- FIRST, authenticate & authorize on EVERY page.
- The Snack should be after authentication, before authorization(?)
  That is, if they don't have a valid session, die().
  If they have a valid session, check the snack.
  If the snack is good, are they authorized to be here.
- Can help prevent parameter tampering, can it help prevent sql injection?
- You must have a good webserver config
  we use strictly https - so snacks on "HTTP GET" command are not visible
  we use rsync scripts to publish to production server
  we restrict admin access to only from .localdomain
- pagechecks() in EVERY PAGE.
  This is critical. If you do nothing else, go back to your office and do this today.
  Authentication: Who is this? Invalid session -> die.
  Expected "snack" value. Snack invalid -> die.
  Authorization: Are they allowed in here? No -> die.
  Expected/optional $_GET, $_POST, $_COOKIES delineated.
  Check for extra variables (sign of tampering).
  HTTP Method(s) Allowed (GET vs POST). Invalid method -> redirect to another appropriate page.
  Send email to engineering if you're interested.
- Don't use "robots.txt" for exclusion:
  For example:
  User-agent: *
  Disallow: customers

  http://www.robotstxt.org/wc/exclusion.html:

"these methods rely on cooperation from the Robot"
That is, you've "unobscured" for the world.

- * Beware of ".bak" or ".old" or "test/" files.
  Also "sample" web site.

  You are responsible for the files that are on your server.
  Don't take the "Wah!" approach like the admin that found Al-Qaeda files on his server. Be proactive.
  Use "rsync" to post from your development server to your production server.
  The result is only files that you put are there.
  (Demonstrate)

  In some cases, this can cause a file to be interpreted differently.
  For example, Apache (by default) sends ".php" files to the PHP interpreter.
  If your developers copy "action.php" to "action.php.old", the source code will be delivered to the browser!
  (Demonstrate)

Side Effects

- Snack can help identify AUTH2 failures: eg you presented a link w/o checking or vice-versa.
- Snacks can wreak havoc on a webapp scanner (eg Acunetix). However, because it's a single function, just change the function on your scanned site.

Conclusion

- http://www.webappsec.org/projects/whid/list_year_2006.shtml
  Shows 34 incidents for 2006. Big-name websites, eg eBay (last month).
- There is no question Web applications are the number one vector for attackers. "The degree to which insecure Web applications can undermine perimeter defenses cannot be over-emphasized. The most robust security measures will not protect back-end systems if holes exist in the applications."
  ...the cost to damaged reputation, network security response, and application repair.
- "Security through obscurity is putting your money under your mattress.
  Security WITH obscurity is putting your money in a safe concealed behind a painting."
  Me 9/27/2004