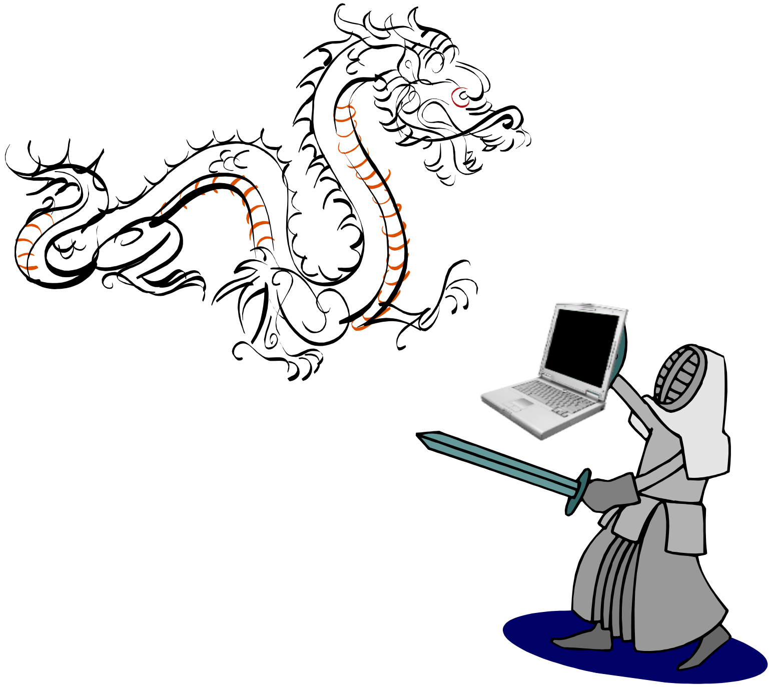
HTTP Response Splitting



HTTP Response Splitting
The Attack

- HTTP Response Splitting is a protocol manipulation attack, similar to Parameter Tampering
- The attack is valid only for applications that use HTTP to exchange data
- Works just as well with HTTPS because the entry point is in the user visible data
- There are a number of variations on the attack

HTTP Response Splitting
The Attack

- An HTTP message response includes two parts :
  - Message Headers – metadata that describes a request or response
    - Each terminated by a carriage return (\r) and a linefeed (\n)

```
GET http://www.google.com/ HTTP/1.1\r\n
Host: www.google.com\r\n
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
rv:1.9.0.1; Google-TR-5.7.806.10245-en) Gecko/2008070208
Firefox/3.0.1 Paros/3.2.13\r\n
Accept: text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
Keep-Alive: 300\r\n
Proxy-Connection: keep-alive\r\n
```

HTTP Response Splitting
The Attack

- Then the Message Body which is the raw data of the response

```
<HTML>\r\n
<HEAD>\r\n
<TITLE>Your Title Here</TITLE>\r\n
</HEAD>\r\n
<BODY>\r\n
</BODY>\r\n
…
</HTML>\r\n
```

HTTP Response Splitting
The Attack

- The Message Headers are also separated from the message body a carriage return/linefeed pair

```
GET http://www.google.com/ HTTP/1.1 \r\n
Host: www.google.com \r\n
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.1; Google-TR-
        5.7.806.10245-en) Gecko/2008070208 Firefox/3.0.1 Paros/3.2.13 \r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 \r\n
Accept-Language: en-us,en;q=0.5 \r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 \r\n
Keep-Alive: 300 \r\n
Proxy-Connection: keep-alive \r\n    ←
 \r\n ←
<HTML>
<HEAD>
<TITLE>Your Title Here</TITLE>
```

HTTP Response Splitting
The Attack

- Those two consecutive carriage-return-linefeed pairs are the source of HTTP response splitting vulnerabilities
- The HTTP response splitting vulnerability is not the attack, it is simply the path that makes it possible
- The key to the attack is ability for an attacker to modify the message headers
- HTML is stateless, so neither the web server nor the browser has any problem with this seemingly odd behavior

Why didn't the creators of HTTP think about this?

HTTP Response Splitting
The Attack – Example

- Let’s understand how a normal page redirection works in HTTP
  - Example: A page containing a redirect script:

```
protected void processRequest(HttpServletRequest aRequest, HttpServletResponse
aResponse) throws ServletException, IOException {
                                        redirect(“http://www.new-url.com”, aResponse);
}
```

  - A request like:
  - would redirect to:

http://www.bank.com/offer.jsp?page=http://www.bank.com/freechecking

  - How do the headers work behind the scenes?

http://www.bank.com/freechecking

HTTP Response Splitting
The Attack – Example

- Under the hood, the request is

```
GET /latestoffer.jsp?page=http://www.bank.com/freechecking HTTP/1.1\r\n
Host: www.bank.com \r\n
      …
\r\n
```

- The server responds with an HTTP 302 (redirect)

```
HTTP/1.1 302 Found \r\n
…
Location: http://www.bank.com/freechecking \r\n
      NOTE: THIS COULD BE THE USER INPUT IN HEADER
…
\r\n
```

HTTP Response Splitting
The Attack – Example

- The browser then fetches the new page

```
GET / HTTP/1.1 \r\n
Host: http://www.bank.com/freechecking  \r\n
…
\r\n
```

- The server responds with HTTP 200 (found) and the page

```
HTTP/1.1 200 OK \r\n
…
\r\n
```

HTTP Response Splitting
The Attack – Example

- But the user can input something that terminates the response and initiates an attack

```
/latestoffer.jsp?page=foobar%0d%0aContent-
Length:%200%0d%0d%0a%0aHTTP/1.1%20200%20OK%0d
%0aContent-Type:%20text/html%0d%0a
Content-Length:%2019%0d%0a%0d%0a<html>Attack</html>
```

%0d%0a is the URL encoding of the \r\n

Remember that you need two \r\n sequences between the headers and the body

HTTP Response Splitting
The Attack – Example

- Which results in

```
HTTP/1.1 302 Moved Temporarily
        Location: http://www.mybank.com/latestoffer.jsp?page=foobar
        Content Length: 0
        HTTP/1.1 200 OK
        Content-Type: text/html
        Content-Length: 19
        <<Anything you want>>
        Server:  gws
        Content-Type: text/html
        Content-Length:%2019

        <html>Attack</html>
        …
```

First HTTP response

Second (inserted) HTTP response

Superfluous data

HTTP Response Splitting
The Attack – Example

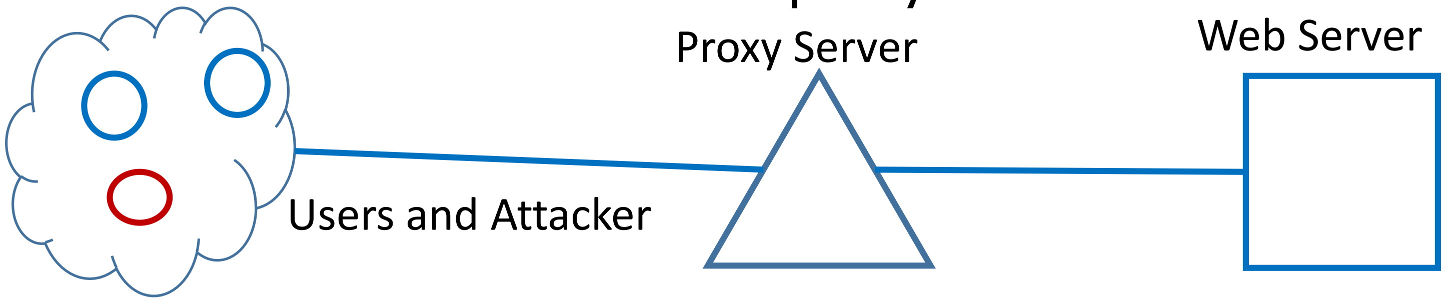
- The dangerous part of this, is <<Anything you want>>
- A script that can take over the user's browser or steal cookie information
  - A redirection to a different host and web page
  - A page that mimics another site and collect credentials
  - It can poison the web cache leading to site defacement
- However, the exploit is not complete
- There are now two responses, but only one request
- The web server will simply hold the second response

HTTP Response Splitting
The Attack - Example

- The attacker has to issue another request
- In the simplest case, simply send http://www.bank.com
- How the attacker does this is dependent on the situation and the attackers goals
- See the following example of cache poisoning

HTTP Response Splitting
The Attack – Cache Poisoning

- One goal of the attacker might be cache poisoning
  - A site has a proxy server for web pages
  - The attacker and victims are behind the proxy server



  - When a response is received by the proxy server, it saves it to answer future requests
  - So the proxy server saves both responses from the attack
  - If the second response defaces a real page, or creates a page with a malicious JavaScript embedded, everyone on the network will get it

HTTP Response Splitting
The Attack – Browser Cache Poisoning

- The attacker creates an HTTP Response Splitting attack based on a URL

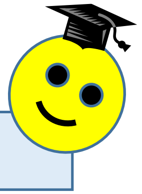
```
http://somesite.com/start.php?first=xxx<script> …
</script>&lang=fr%0d%0aContent-Length:0%0d%0a
HTTP/1.1%20200%20Found%0d%0aContent-
Length:550%0d%0a …
```

- and seduces a victim into clicking on it
- The web servers first response contains a Cross-site Scripting attack
- The script issues an Ajax request that sends the second request
- And the victim’s web cache (and any proxy server) is poisoned

HTTP Response Splitting Consequences

- HTTP Response Splitting can lead to:
  - Cross-site Scripting (XSS) attacks
  - Cross User Defacement
  - Web Cache Poisoning
  - Page Hijacking
  - Browser Cache Poisoning
  - Browser Hijacking

????



HTTP Response Splitting Discovery

- Check for any data outside of the Trust Boundary that is used in any HTTP header
  - Try inserting a carriage return/linefeed pair to see it is allowed to pass through
  - If so, you have a vulnerability
  - Be suspicious of redirects in code – they often use information stored in the client
- Be aware that Post data can also be used in an attack
  - It may be advantageous, because URL's have limited length
  - It requires that the attack be perpetrated via a script so it is more difficult to implement

HTTP Response Splitting Remediation

- If there are values outside the Trust Boundary that are used in HTTP messages,
  - Validate the values by whitelisting
    - They are only allowed to be certain values, nothing else
    - For example, all language designators must be two alphabetic characters, exactly
- In the event that a subject parameter might be allowed to contain a CR/LF pair, URL encode all data in HTTP headers with the HTML entity reference
  - \r => &#13;
  - \n => &#10;
  - This prevents them from being accepted as the control sequence \r\n

HTTP Response Splitting Avoidance

- Design Phase
  - Identify all application inputs that could be used in HTTP headers
  - Specify secure coding guidelines for handling the data
  - Reduce the number of cases as much as possible to reduce the attack surface size
  - Establish a test plan for validating that all cases are correctly remediated
  - If client-side data is used to redirect or modify the HTTP headers, remap the data to an ordinal set on the server side
    - If there are 10 pages you can redirect to, change them to 'A' .. 'J' externally
    - This is essentially a look up table and prevents the attacker's content from being used in an attack

HTTP Response Splitting Avoidance

- Implementation Phase
  - All inputs must be validated
  - Be aware of any use of input data in HTTP headers and code accordingly
- Testing Phase
  - Use dynamic analyzers to validate the application (they are good at finding this vulnerability)