Cross Site Scripting (XSS)

Troy Hunt
troyhunt.com
@troyhunt

pluralsight
hardcore developer training

Outline

- Understanding untrusted data and sanitisation
- Establishing input sanitisation practices
- Understanding XSS and output encoding
- Identifying the use of output encoding
- Delivering a payload via reflected XSS
- Testing for the risk of persistent XSS
- The X-XSS-Protection header

What constitutes untrusted data?

- **The integrity is not verifiable**
- **The intent may be malicious**
- **The data may include payloads such as:**
  - SQL injection
  - Cross site scripting
  - Binaries containing malware

Common sources of untrusted data

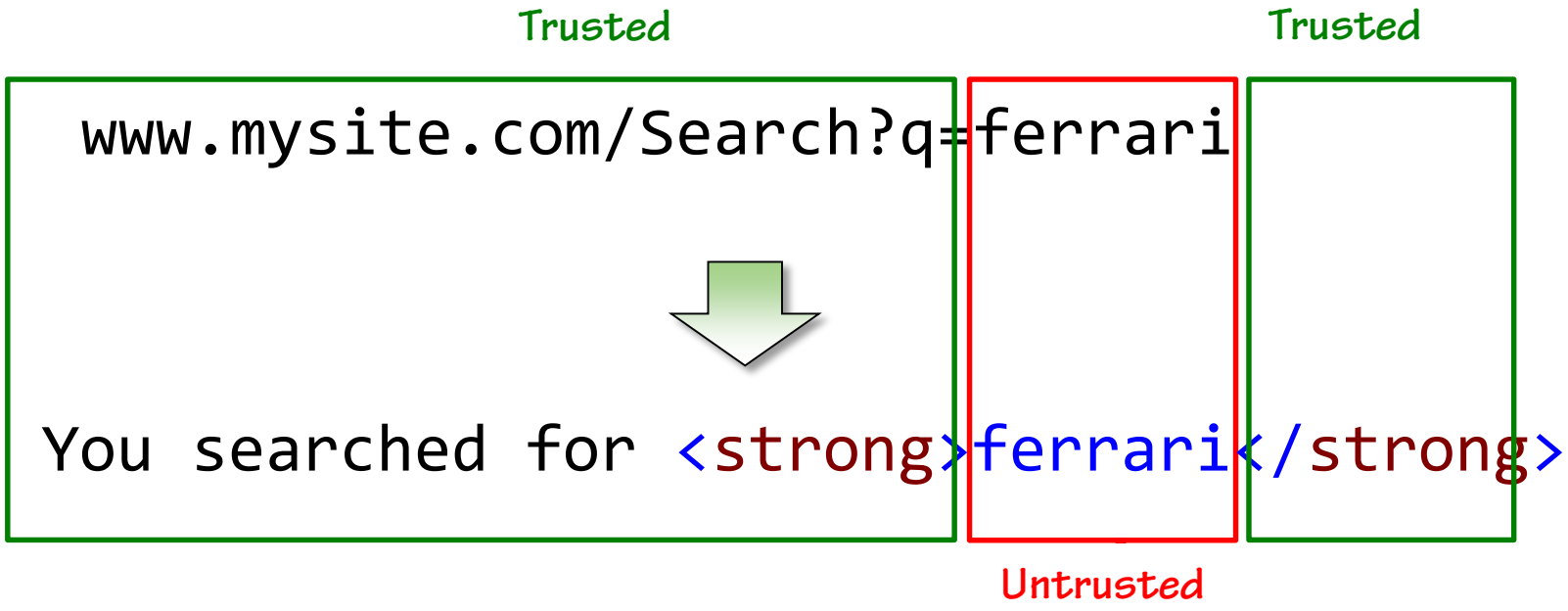
- **From the user**
  - In the URL via a query string or route
  - Posted via a form
- **From the browser**
  - In cookies
  - In the request headers
- **From any number of other locations**
  - External services
  - *Your own database!*

The use of input sanitisation

- **Untrusted data that might constitute a risk is frequently sanitised including characters such as:**
  - <>'/"\;
- **Explicitly rejecting specific characters is called a "blacklist" approach**
  - It's very implicit: "This is what could be bad so everything else *must* be ok!"
  - It's also easy to be incomplete
- **Explicitly accepting only approved characters is called a "whitelist" approach**
  - It's very explicit: "This is what we know is good so we're only going to allow these"
  - It's a more comprehensive, lower risk approach

Understanding reflected untrusted data

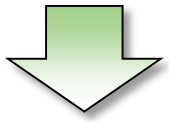
Identifying an XSS risk



Exploiting an XSS risk

Markup

www.mysite.com/Search?q=ferrarienzo



You searched for ferrarienzo

Markup

Output encoding concepts

- **The search term was never intended to be *markup*, only ever *data***
- **XSS attacks are possible because the app allows an XSS payload to break out of the data context and change the markup context**
- **To mitigate the risk of XSS, we want to make sure the search term appears on the screen *exactly* as it was entered**
- **So how do we write markup to display "enzo" on the screen?**
  - <i>enzo</i>

Output encoding contexts for “enzo”

- HTML <i>enzo</i>
- CSS \<i\>enzo\</i\>
- JavaScript '\x3ci\x3eenzo\x3c\x2fi\x3e'
- URL \00003Ci\00003Eenzo\00003C\00002Fi\00003E
- LDAP distinguished name \<i\>enzo\</i\>
- HTML attribute, HTML form URL, LDAP filter, URL path, XML, XML attribute

Summary

- **Sanitisation is the first defence against untrusted data**
  - Apply whitelist validation wherever possible
  - Missing sanitisation rules are easily discoverable by passing character ranges the website
- **Output encoding is absolutely critical for mitigating the risk of XSS**
  - Remember to encode for the correct context
  - Missing or incomplete encoding is easily discovered by inspecting the response body
- **Don't trust your own data – persistent XSS is an often-overlooked threat**
- **Native browser defences are great additional protection**
  - Don't disable them in IE through the use of the X-XSS-Protection header – fix the reason it's required instead!