Cross Site Scripting (XSS)

Troy Hunt

troyhunt.com

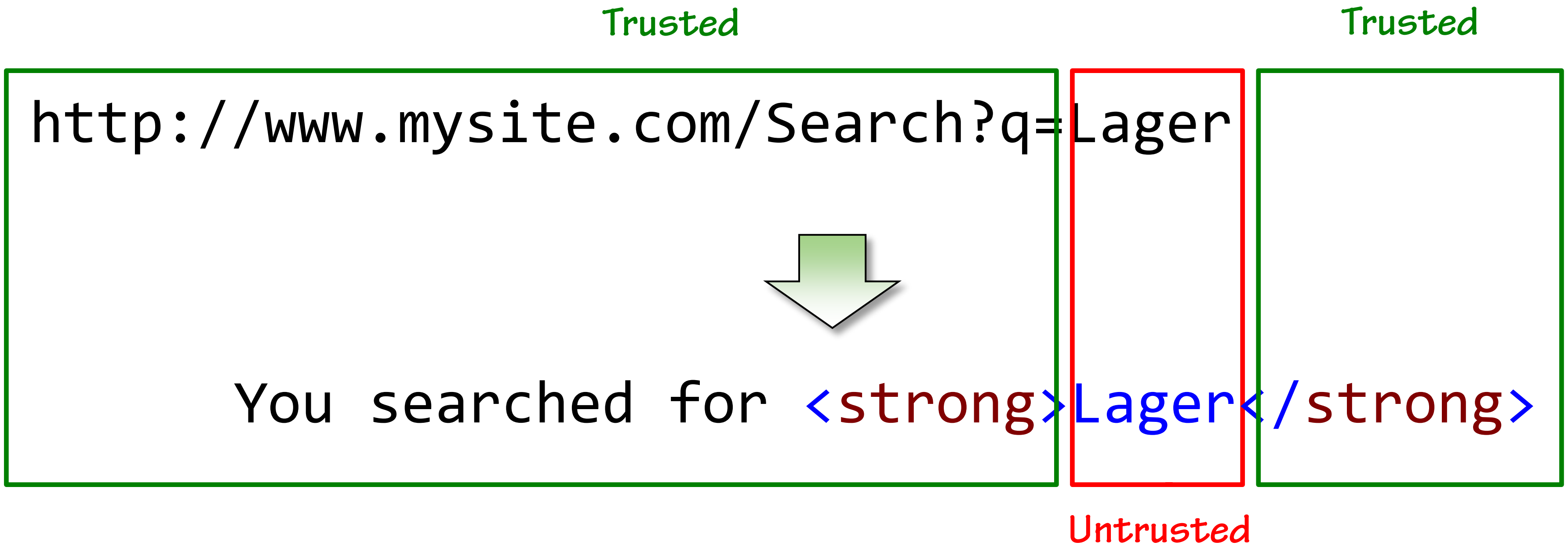
Outline

- How OWASP views the risk
- Performing an attack
- Understanding XSS
- Overview and implementation of output encoding
- Native output encoding implementations in web forms and MVC
- Whitelisting
- Using request validation as a native defence
- Reflective and persistent XSS
- Native browser defences
- Payload obfuscation

OWASP overview and risk rating

Threat Agents
—
Consider anyone who can send untrusted data to the system including external users, internal users, and administrators.

Understanding XSS



```
http://www.mysite.com/Search?q=Lager

        You searched for <strong>Lager</strong>
```

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 “<i>Lager</i>” on the screen?
  - &lt;i&gt;Lager&lt;/i&gt;

Output encoding contexts

- CSS
- HTML
- HTML attribute
- HTML form URL
- JavaScript
- LDAP distinguished name
- LDAP filter
- URL
- URL path
- XML
- XML attribute

How does this change our encoding?

- **Take our original string of “<i>Lager</i>” and encode for different contexts:**

Context	Encoded string
HTML	&lt;i&gt;Lager&lt;/i&gt;
JavaScript	\x3ci\x3eLager\x3c\x2fi\x3e
CSS	\00003Ci\00003ELager\00003C\00002Fi\00003E

- ***Always* use a well-proven library to implement output encoding**

Summary

- **Output encoding is the cornerstone of XSS protection**
  - Remember to correctly encode for different output contexts
- **Be wary of the encoding differences between web forms controls**
  - Many control properties don’t natively implement encoding
  - ASP.NET MVC offers excellent *implicit* encoding support
- **Whitelist are still very important**
  - Always validate all untrusted data against a whitelist of allowable values
- **Request validation is an important safety net**
  - It’s even better in .NET 4.5 but don’t rely on it alone
- **Code your app as though you have persistent XSS in the database**
- **XSS browser defences exist, but they’re inconsistent**
- **Expect attackers to use obfuscated URLs**