# Parameter Tampering

Troy Hunt
troyhunt.com
@troyhunt

pluralsight
hardcore developer training

# Outline

- Identifying untrusted data in HTTP request parameters
- Capturing requests and manipulating parameters
- Manipulating application logic via parameters
- Testing for missing server side validation
- Understanding model binding
- Executing a mass assignment attack
- HTTP verb tampering
- Fuzz testing

# 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

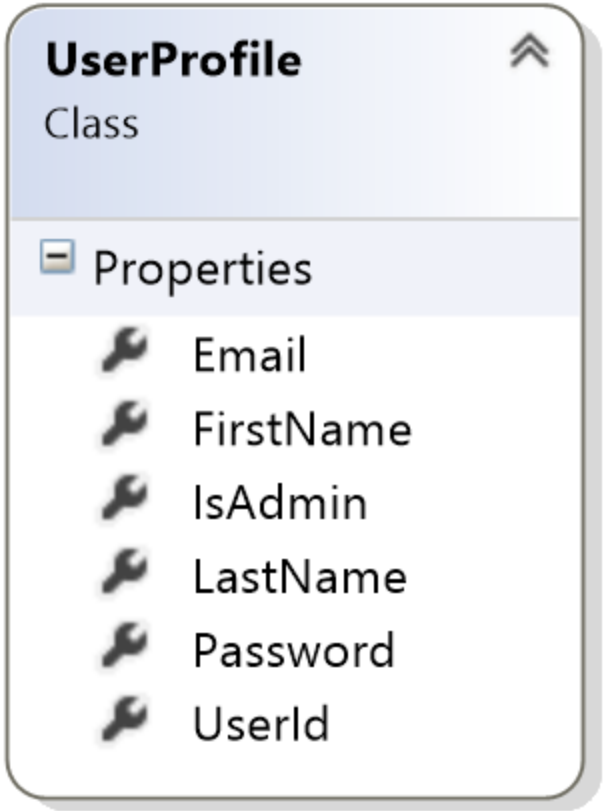
- **From the user**
  - In the URL via a query string or route
  - Posted via a form
- **From the browser**
  - In cookies
  - In the request headers
- **From any number of other locations**
  - External services
  - *Your own database!*

# What can be tampered within an HTTP request?

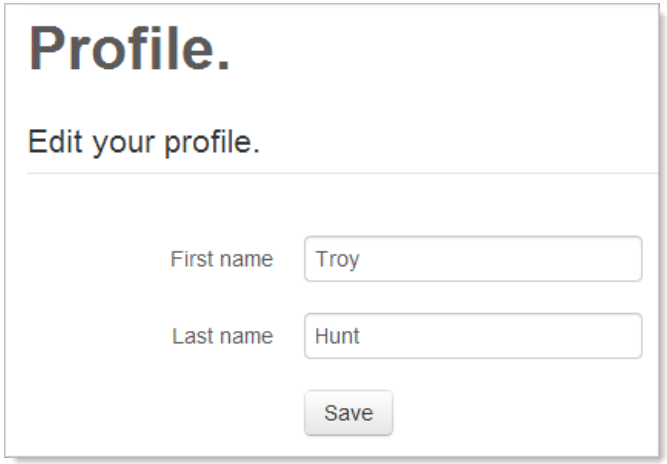
```
POST https://hackyourselffirst.troyhunt.com/Account/Login HTTP/1.1
Host: hackyourselffirst.troyhunt.com
Connection: keep-alive
Content-Length: 79
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Origin: https://hackyourselffirst.troyhunt.com
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36
    (KHTML, like Gecko) Chrome/28.0.1500.72 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: https://hackyourselffirst.troyhunt.com/Account/Login
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Cookie: ASP.NET_SessionId=qah03aymu2ion4mwj523feod;
    VisitStart=18/07/2013 12:01:10 PM

Email=troyhunt%40hotmail.com&Password=passw0rd&RememberMe=true
```

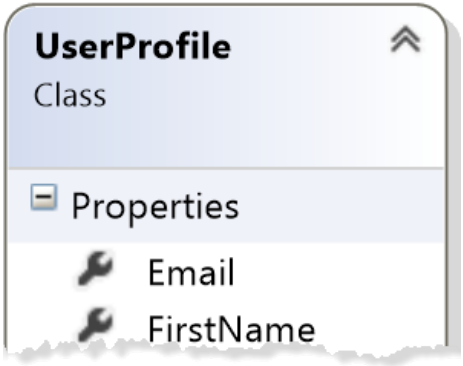
# Understanding models



# Automatically binding a form to a model




```
POST http://hackyourselffirst.troyhunt.com/Account/UserProfile HTTP/1.1
…
FirstName=Troy&LastName=Hunt
```



# Typical request payloads an attacker may test

- **Cross site scripting**
  - Patterns to manipulate page source
- **SQL injection**
  - Patterns to manipulate query execution
- **Directory traversal**
  - Patterns to access system files

# Why fuzz testing?

- **Vulnerability testing via untrusted data regularly adheres to common patterns**
- **Manual testing can be laborious:**
  - Lots of different attack payloads (malicious requests)
  - Lots of different attack vectors (untrusted data fields)
- **Fuzz testing can automate the process of bombarding an application with random data**
- **Fuzz testing tools generally identify responses which could indicate a vulnerability is present**

# Summary

- **Always assume that all aspects of an HTTP request can be – *and will be* – manipulated by attackers**
  - Verb, path, protocol, accept headers, user agent, referrer, accept language, cookies, request body
- **Don't rely on controls that depend on the browser to implement them**
  - Validation, for example, must still occur on the server
- **Be conscious of where risks might be present in automated processes**
  - Watch out for model binding and mass assignment risks
- **Consider which verbs should be allowed on each resource**
- **Automate your testing**
  - Fuzzing is a very simple way to get started with automation