Insufficient Transport Layer Protection

Troy Hunt
troyhunt.com

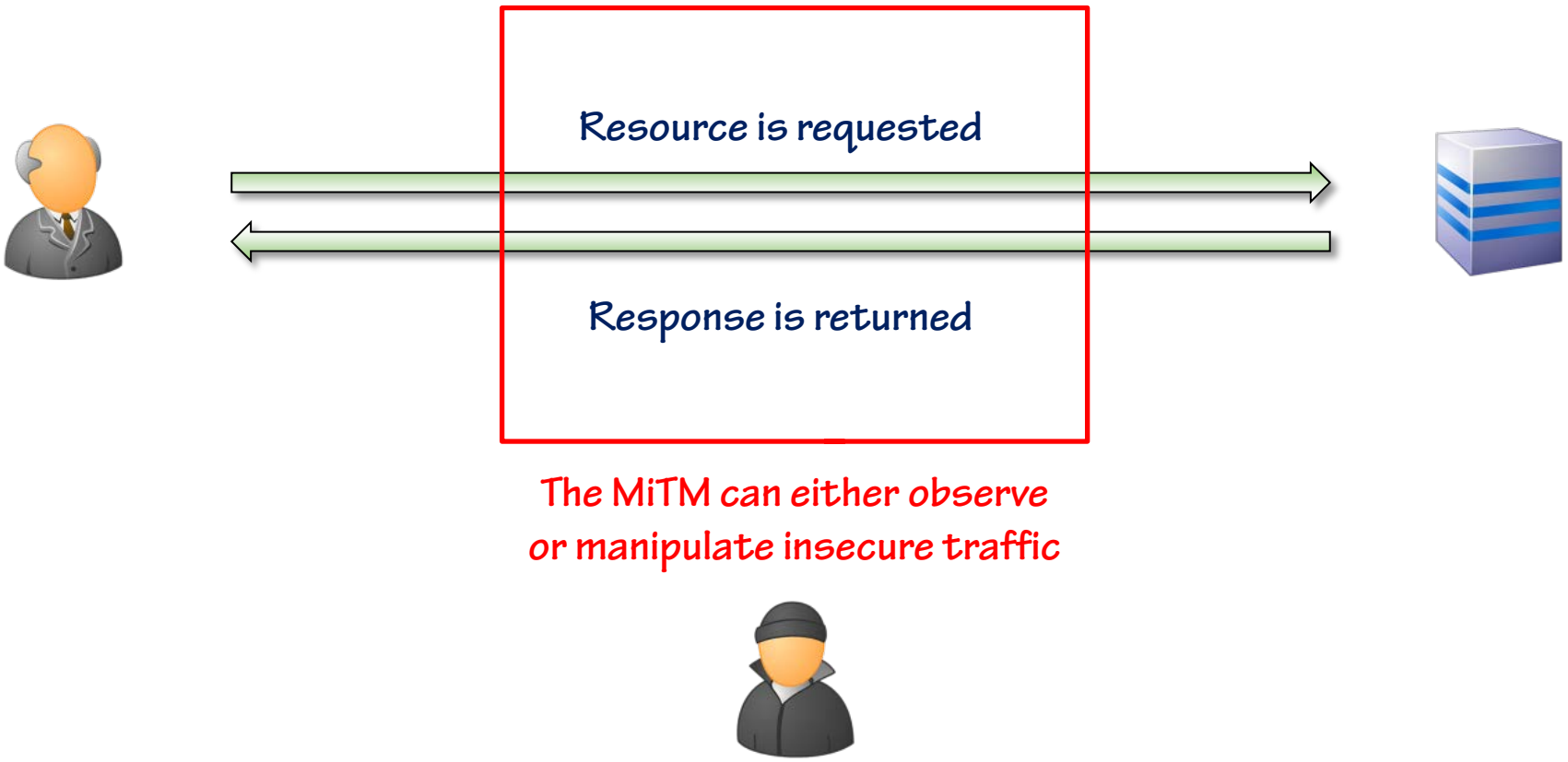
Outline

- How OWASP views the risk
- Performing an attack against a vulnerable application
- Understanding secure cookies and how to implement them in .NET
- Forcing web forms and MVC to use a secure connection
- The risk of mixed mode content
- Using HTTP strict transport security to force secure requests
- Patterns of insufficient HTTPS and other considerations

OWASP overview and risk rating

Threat Agents
—
Consider anyone who can monitor the network traffic of your users. If the application is on the internet, who knows how your users access it. Don't forget back end connections.

Understanding a man in the middle (MiTM) attack



Getting an MiTM between victim and the server

- **There are many, many ways of intercepting traffic:**
  - Physically tap an Ethernet cable
  - Intercept traffic at the ISP level
  - Monitor unprotected traffic at a Wifi hotspot
  - Create a rogue wireless access point
- **Rogue wireless access points are extremely easy to setup**
  - We'll use a Wifi Pineapple for this example

Attack scenario

Victim

Attacker

Using HTTP strict transport security

- A way to disallow the browser from making *any* HTTP requests to a site is to implement an HSTS header
- This can be done very simply in the BeginRequest event:

```
HttpContext.Current.Response
.AddHeader("Strict-Transport-Security", "max-age=31536000");
```

- For the next 31,536,000 seconds (12 months), the browser may not make an HTTP request to the site

But HSTS has some restrictions

- **The header will only be observed if sent with an HTTPS response**
- **The certificate must be trusted**
- **The user usually still needs to be able to issue an HTTP request first that doesn't get hijacked**
- **Browser support is patchy:**
  - Works in Chrome and Firefox
  - Ignored in Internet Explorer and Safari
- **Go ahead and use HSTS – it doesn't hurt – but it's also far from comprehensieve**

Other insufficient TLS patterns

- **Loading login forms over HTTP, even if they post to HTTPS**
  - The credentials may be protected in transit, but they can also be harvested if an MiTM has injected script into the page
- **Loading HTTPS login forms inside an iframe on an HTTP page**
  - The parent page is vulnerable so it may have been manipulated to load a different login form into the iframe
- **Allowing a page to load over HTTP when there is no use case where it ever should**
  - For example, a login page should never be allowed to load over HTTP
- **Passing sensitive data such as credentials in HTTPS addresses**
  - There are still multiple points where this may be logged

SSL comes with a performance cost (just)

- **Encrypting and decrypting traffic has to have *some* overhead on the server infrastructure**
- **It does, but it's very little according to Google after they moved GMail to HTTPS only:**

  "In order to do this we had to deploy no additional machines and no special hardware. On our production front-end machines, SSL/TLS accounts for less than 1% of the CPU load, less than 10KB of memory per connection and less than 2% of network overhead. Many people believe that SSL takes a lot of CPU time and we hope the above numbers (public for the first time) will help to dispel that."

  - Google, 2010

Even HTTPS everywhere still has risks

- **Take a common scenario:**
  - User types americanexpress.com into their address bar and presses enter
  - The browser defaults the scheme to HTTP and makes the request
  - The server responds with an HTTP 301 “Moved Permanently” with a new location of https://www.americanexpress.com
  - The browser requests the Amex website over a secure scheme
- **The second step is still vulnerable as it’s an HTTP request**
- **MiTM tools such as sslstrip can proxy traffic to a secure site backwards and forwards between HTTP and HTTPS**

Summary

- **There are many ways of implementing HTTPS that are *insufficient***
  - It's not a case of having TLS or not having it, it has to be done right
- **Remember that cookies can be sensitive due to what they allow an attacker to do with them**
  - Flag them as secure to minimise the risk of an MiTM attack
- **Make sure that resources which require secure connections cannot be loaded over HTTP**
- **Ensure that secure pages do not embed insecure resources**
- **Use HSTS for secure sites, but remember the limited support**
- **Avoid the security anti-patterns: login forms over HTTP, insecure pages embedding secure login forms and sensitive data in the URL**
- **SSL has some overhead, just not much and it's also not fool proof**