Sleeping Easy

Secure development in the real world

Mark Young
mark@developer.geek.nz
WARNING: I'm not a pen tester (I'm not even a member of anonymous... or LulzSec)
Deliver Projects...

- On time
- Under budget
- Functionally complete
- With a happy client...
- ...and a sane team
- That perform well
- Are maintainable
- Look good...
- ...and are secure
Low friction security...

How do you build secure web applications without it costing you a fortune in money or sleep?

1. Architecting a secure culture in your business

2. Architecting secure applications
Why?
Security costs!

Key benchmark sample statistics on the annualized cyber crime cost:

- Maximum value: $51,925,510
- Fourth quartile average: $15,567,136
- Grand mean: $6,459,362
- Mean minus highest value: $5,426,040
- Mean minus two highest values: $4,699,304
- Third quartile average: $4,611,172
- Median: $3,788,468
- Second quartile average: $3,180,182
- First quartile average: $1,650,976
- Minimum value: $1,037,277

Ponemon Institute: First Annual Cost of Cyber Crime Study
Security costs

- Insecure applications are in the wild – lots!
- People ready to exploit your applications are in the wild
Why doesn’t everyone work securely?

• “Close enough’s good enough, don’t worry about that stuff”
• “they’re not testing that”
• “we’re not being paid to do that”
• “it won’t happen to us”
• “Just get it to production, we don’t have time to fix any of that now”

• “What’s what?”
Architecting your business for security
1. Increase the awareness of security

- Become a prophet of doom – “repent or be hacked”
- Scare people again
- Advocate best practice
- Demonstrate vulnerabilities using real well-known applications
- Include management

http://johngushue.typepad.com/
2. Make security a first class citizen in projects

- Ensure security is part of non-functional requirements
- Document specific risks in risk registers
  - Customer information disclosure
  - Negative news media
  - Loss of IP
  - Business disruption
  - Revenue loss
- Include security checklist in gating processes
- Schedule reviews in project plans
3. Empower your developers

- Demonstrate the fun side of application security
- Train
  - Make sure they at least know the top 10 and how vulnerabilities can be exploited
- Challenge
  - Turn your developers into testers
  - OWASP WebGoat (http://code.google.com/p/webgoat/)
  - Web Security Dojo (http://sourceforge.net/projects/websecuritydojo/)
4. Review

- Be humble
- Suspect everything
- Always keep a security eye patch on
Architect your code for security
Design for security

- When designing solutions and applications, include security
- Document how you’ll meet the OWASP Top 10 up front at the beginning of the project
- Assume developers will follow the path of least resistance – don’t rely on them
- Learn from your mistakes – if at all possible incorporate into a framework.
Security Design Principles

1. Secure by default
2. Defence in depth
3. Reduce your attack surface
4. Understand your frameworks
   • Authentication
   • Resource inclusion
   • Rendering
   • Validation
5. Make it easy
Also remember...

• Internal sites are still susceptible
  • How many companies have a sharepoint server called “intranet”, “moss” or “sharepoint”?  

• Make sure monitoring plans are in place for production systems 

• Application security is just one piece of the puzzle 

• Look to limit social exploits as well
Most common flaws

- A4: Insecure Direct Object References
- A2: Cross-Site Scripting (XSS)
- A5: Cross-Site Request Forgery (CSRF)
- Weak uses of encryption / custom rolled authentication
XSS

<h2>
  Hi <%= Html.Encode(Model.Name) %>!
</h2>

<%= Model.GoogleMapHtml() %>

<h2>
  Hi <%= Model.Name %>!
</h2>

<%= Model.GoogleMapHtml().AsHtml() %>

<h2>
  Hi <%= Model.Name %>!
</h2>

<%= Model.GoogleMapHtml() %>
<form method="POST" action="/cart/purchase">
  %= Html.AntiForgeryToken() %>
  <input name="bookid" type="hidden"
  <input type="submit" value="Buy Now!" />
</form>

<% using (Html.BeginFormWithAntiForgery("purchase", "cart")) { %>
  <input name="bookid" type="hidden"
  <input type="submit" value="Buy Now!" />
  <input type="submit" value="Buy Now!" />
</form>

<form method="POST" action="/cart/purchase/?key=12345">
  <input name="bookid" type="hidden" />
  <input type="submit" value="Buy Now!" />
</form>
Insecure Direct Object References

GET /user/account?id=12

[Authorize(Roles="Admin")]
[HttpGet]
public ActionResult Account([MapReference("UserId")] string id)
{
    var user = _users.FindById(Session.UserIdMap.GetId(id));
    if (user == null)
        return HttpNotFound();
    return View(user);
}
Broken Encryption

• Don’t do it! Unless you know what you’re doing
• Get it reviewed, and reviewed again…
• Padding oracles, known plaintext, chosen cipher-text attacks
• Use MACs

RijndaelManaged symmetricKey = new RijndaelManaged() { Mode = CipherMode.CBC };

// Reuse shared secret as IV
ICryptoTransform decryptor = symmetricKey.CreateDecryptor(
    symmetricKeyBytes, symmetricKeyBytes);
So, How do we build secure apps in a low-friction manner?

- Start off by changing mindsets in your business
- If necessary scare people
- If they still won’t listen, scare them some more
- Continue by empowering your team
- Finish by designing applications so that the “path of least resistance” follows secure development practices