DATACOM

Sleeping Easy

Secure development in the real world

Mark Young
mark@developer.geek.nz

tester

ber of anonymous...

pics.myspew.com

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?

HELL

SONY
make.believe

Labour

RSA

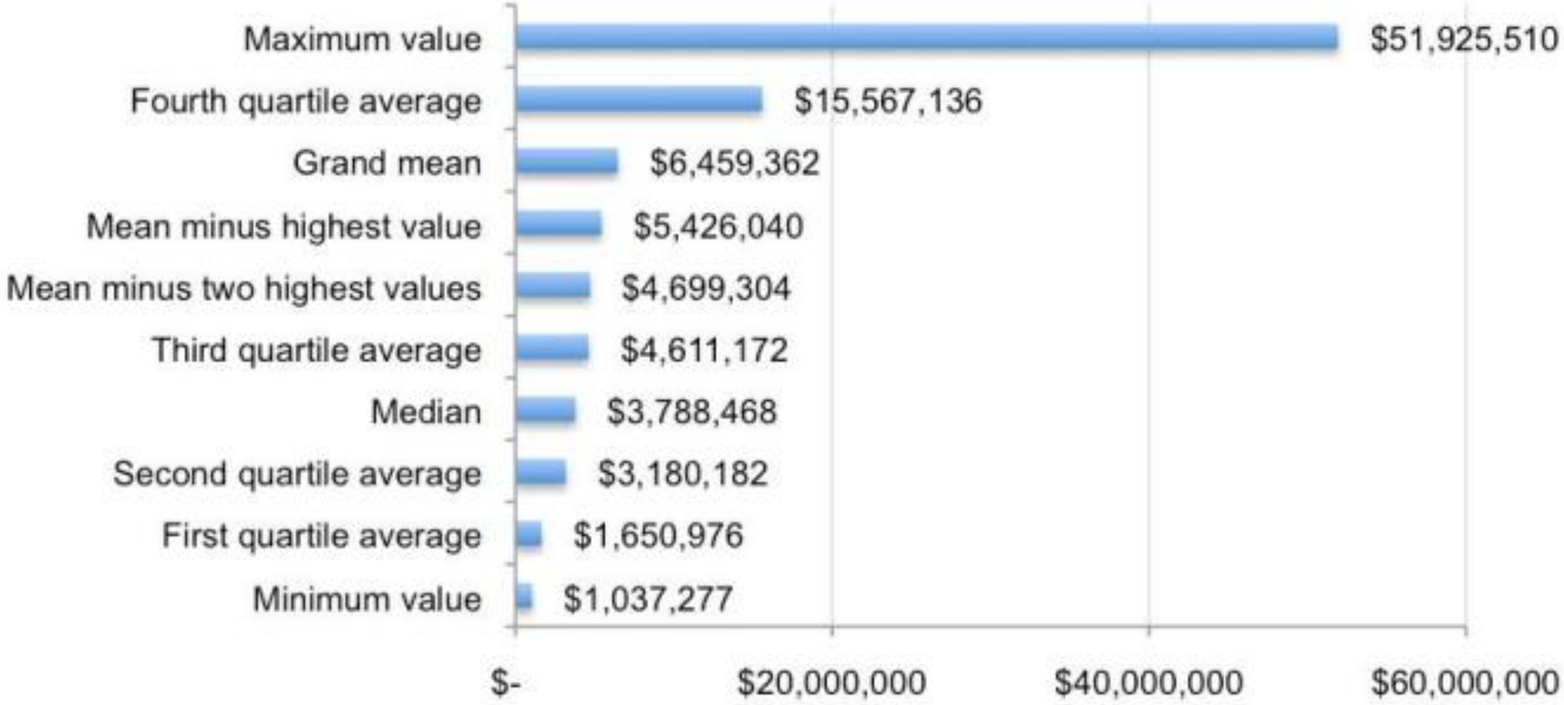
distribute.IT

FOX

HBGary
Detecting Tomorrow's Malware Today

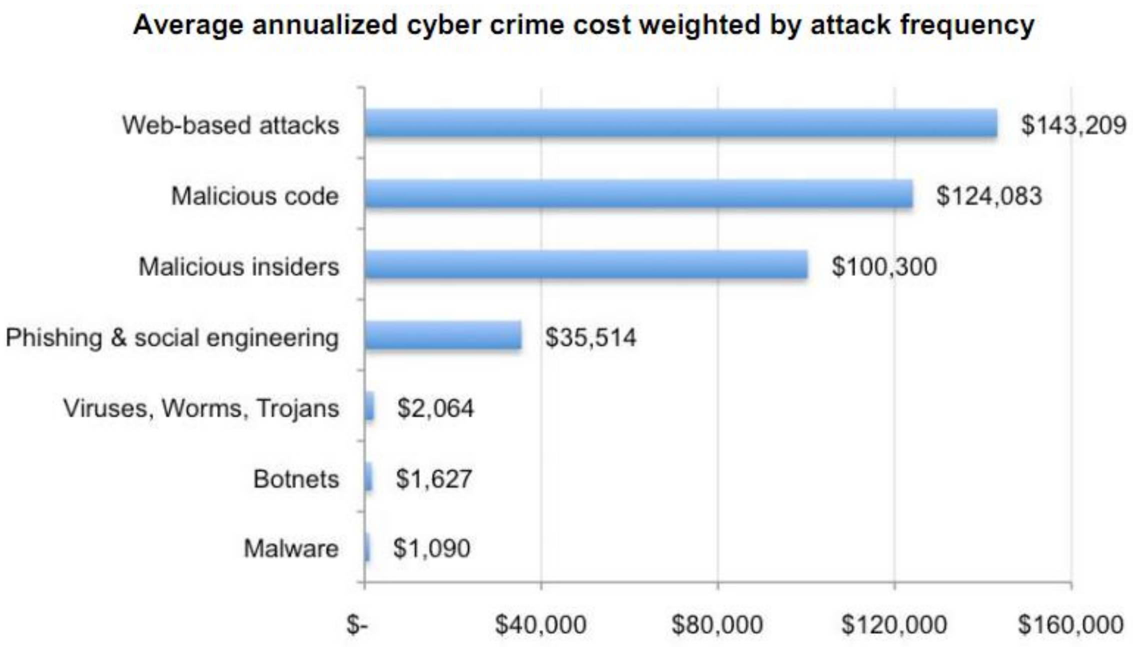
Security costs!

Key benchmark sample statistics on the annualized cyber crime cost



Ponemon Institute: First Annual Cost of Cyber Crime Study

Security costs



Ponemon Institute: First Annual Cost of Cyber Crime Study

- Insecure applications are in the wild – lots!
- People ready to exploit your applications are in the wild

Why doesn’t everyone work securely?

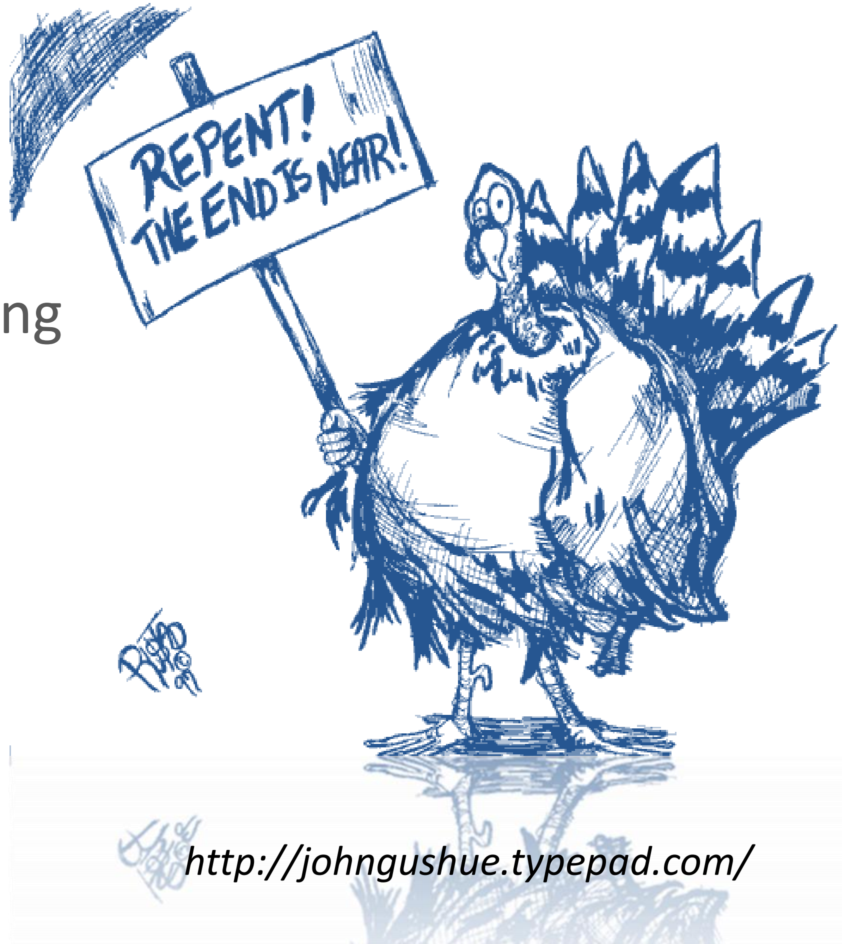
- “Close enough’s good enough, don’t worry about that stuff”
- “they’re not testing that”
- “we’re not being paid to do that”
- “it won’t happen to us”
- “Just get it to production, we don’t have time to fix any of that now”

- “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/)
  - OWASP LiveCD (https://www.owasp.org/index.php/Category:OWASP_Live_CD_Project)
  - 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>
  Hi <%: Model.Name %>!
</h2>
<%= Model.GoogleMapHtml() %>
```

CSRF

```
<form method="POST" action="/cart/purchase">
    <%= Html.AntiForgeryToken() %>
    <input name="bookid" type="hidden"
```

```
<% using (Ht                                              )
   { %>
  <input nam
  <input typ
<% } %>
```

```
<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 chipher-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