KCDC

The Hacker Psyche Exposed

HACKED!

Attackers Advantage & Defenders Dilemma

Mike Benkovich

mike@benko.com

@mbenko

Benkotips
Technology Services

About me...

- Mike Benkovich – mike@benko.com
- First computer was Commodore PET
- Avid blogger on www.benkotips.com
- Last job was MSDN Evangelist for Microsoft
- Entrepreneur – Founder of Imagine Technologies, Inc.
- Follow me on twitter @mbenko
- Founder of TechMasters (Toastmasters for Geeks)

  www.techmasters-tc.com - #TechMasters
- Links from today – http://bit.ly/hacktrx

Agenda

- Growing importance of security
- Principles of Security
- Threat Modeling
- Know your threats

Is security important?

- What is the conversation about?
- Security vs. Identity
- What are we protecting?
- Where is the threat?
- Who is the enemy?

the perfect scenario

- Eliminate all data and allow no users

the perfect scenario

- Assume all users are evil and all input is corrupt

Benkotips Technology Services

the perfect scenario

- Disconnected from all other machines

the perfect scenario

- Get some sharks with laser beams

the real world

- - 69% chance of falling victim to cybercrime in your lifetime
- - 1 out of 3 hacks originated in the USA
- - 57 million Americans receive scam emails per year
- - In 2011 77 million accounts on Sony Playstantion were hacked at once
- - In one year about $1 trillion in intellectual property worldwide is hacked
- - 110 million Target credit card identities stolen 2013 holiday season

http://holykaw.alltop.com/wp-content/uploads/2013/04/hacker-target-victim-statistics-infogrphic-e1367241780834.jpg

Benkotips Technology Services

DEMO TIME

The Golden Rule of Security

All users are Evil.

All input is corrupt…

…until proven otherwise!

Attackers advantage & defenders dilemma

1. The defender must defend all points
   The attacker can choose the weakest point
2. The defend can only defend known exploits
   The attacker can probe for new ones
3. The defender must be constantly diligent
   The attacker can attack at will
4. The defend must play by the rules
   The attacker can play dirty

Build Secure Apps

- Follow secure coding techniques
- Engage Threat Modeling
- Design with security in mind
- Apply proven security principles.
- Know security threats

Defense in Depth...

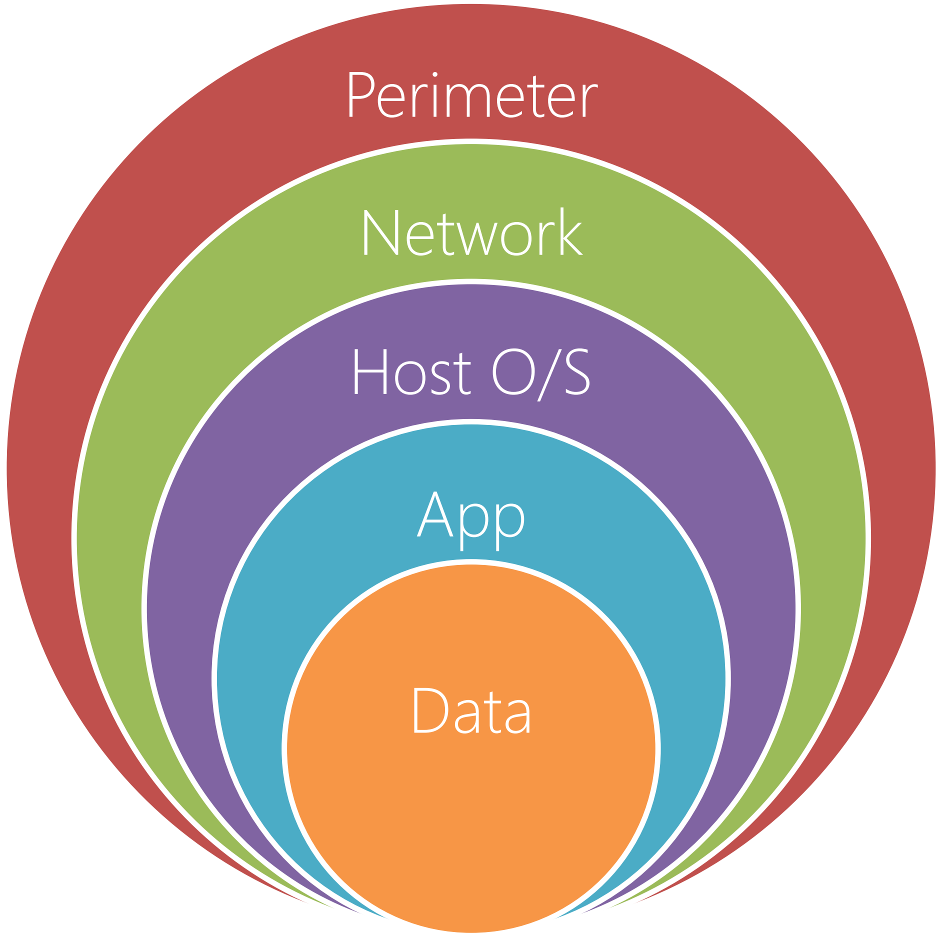
> Perimeter – Data center, theft, security of devices and machines

> Network – Firewall, viruses and worms

> Host/OS – Patched machines, buffer overflows

> App – XSS, Insecure direct obj references, session mgmt, injection

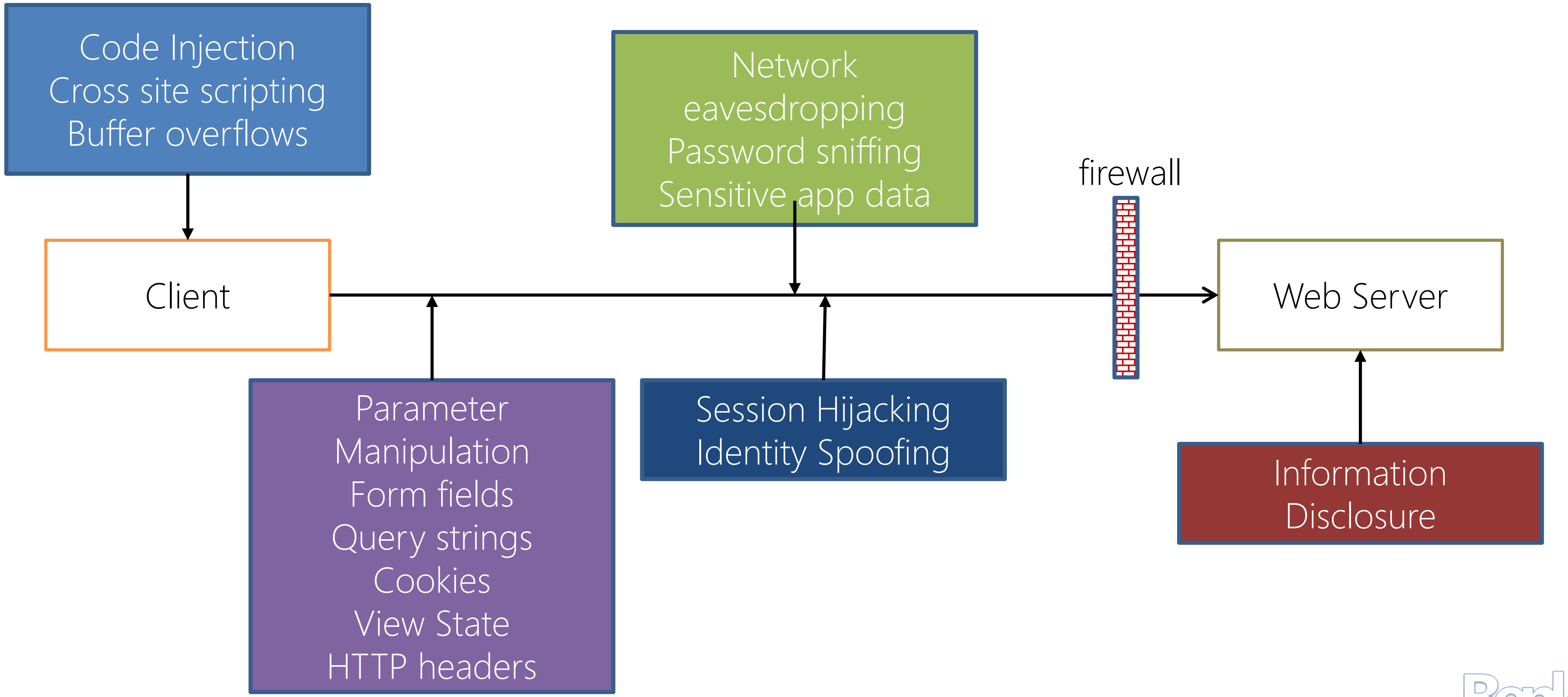
> Data – Encrypted at rest, security ACL's, identity



Threat Modeling

- Identify Assets
- Decompose the Application
- Identify the Threats
- Document the Threats
- Rate the Threats

Common threats



OWASP - http://owasp.org

- Open Web Application Security Project
- Mission: To make software security visible
- Tracks common exploits and provide documentation
- List of go to resources
- Top 10 exploits

Current Top 10 Exploits (updated 2013)

1. Injection
2. Broken Authentication and Session Management
3. Cross-Site scripting (XSS)
4. Insecure Direct Object References
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Missing Function Level Access Control
8. Cross-Site Request Forgery
9. Using Components with Known Vulnerabilities
10. Unvalidated Redirects and Forwards

OWASP Top Security Risks

Cross Site Scripting (XSS)

Injection attack

Insecure Direct Object Reference

Integer Overflow

OWASP Top Security Risks

Cross Site Scripting (XSS)

Injection attack

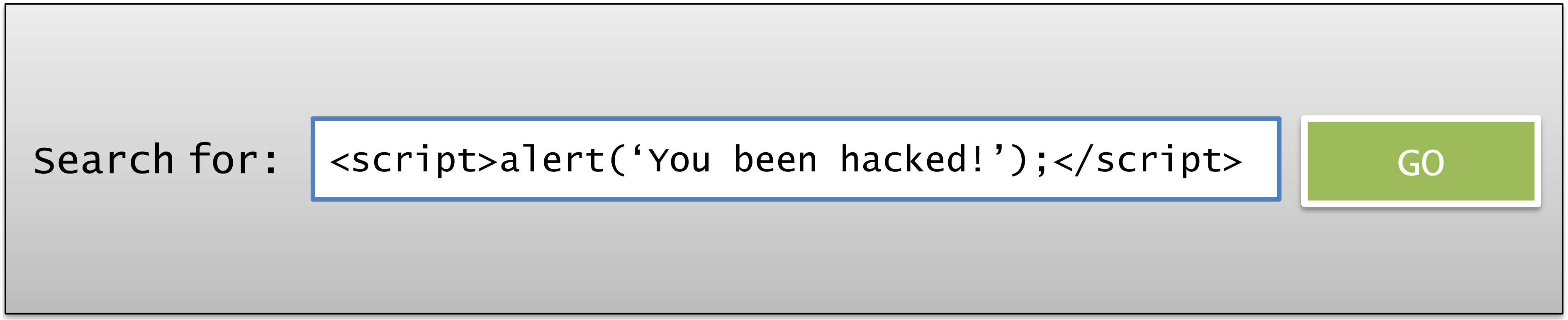
Insecure Direct Object Reference

Integer Overflow

Cross Site Scripting

What is it Cross Site Scripting?

- Allows hackers to run malicious script in a client's Web browser
- Any Web page that renders dynamic HTML based on content that users submit is vulnerable



```
Search for: <script>alert('You been hacked!');</script>
```

Cross Site Scripting

DEMO

Benkotips Technology Services

Cross Site Scripting

- Potential Risks
- Hackers can embed <script>, <object>, <applet>, and <embed> tags
- Hackers can steal Web session information, modify the user's screen

Cross Site Scripting

- How To Mitigate
- Validate and constrain input
- Properly encode output
- Microsoft Anti-Cross Site Scripting Library
- What about Server.HTMLEncode?
  - Uses blacklist for exclusion
  - Less secure

Cross Site Scripting

- Real World Example
- Attackers redirected PayPal visitors to a page warning users their accounts had been compromised.
- Victims were then redirected to a phishing site and prompted to enter sensitive financial data.

Source: http://www.acunetix.com/news/paypal.htm

OWASP Top Security Risks

Cross Site Scripting (XSS)

Injection attack

Insecure Direct Object Reference

Integer Overflow

OWASP Top Security Risks

Cross Site Scripting (XSS)

Injection attack

Insecure Direct Object Reference

Integer Overflow

SQL Injection

- What SQL Injection?
- Affects dynamic SQL queries which utilize user input as part of the query
- Attacker submits data containing a command that SQL server executes
- Attack Vectors

  - Query strings   - Forms   - Web Services

For example...authentication

- Using an unexpected value in a dynamic SQL statement

```
SELECT CustomerID FROM CMRC_Customers
WHERE EmailAddress = '' AND Password =''


SELECT CustomerID FROM CMRC_Customers
WHERE EmailAddress = 'bob' or 1=1;--' AND Password =''
```

For example...

what happens if you add an unexpected string to the user name?

SQL Injection

- Potential Risks
- Probe databases
- Bypass authorization
- Execute multiple SQL statements
- Call built-in stored procedures (e.g. xp_cmdshell)

SQL Injection

DEMO

SQL Injection

- How to Mitigate
- Constrain and sanitize input data.
- Use type-safe SQL parameters
- Restrict permissions for account used to access database
- Do not disclose error information
- Use LINQ to SQL to access and interact with data

SQL Injection

- Real World Example
- The official government website for the state of Rhode Island (www.ri.gov) was the victim of a SQL Injection attack in January of last year.
- Hackers allegedly stole credit card data from individuals who have done business online with state agencies.
- The hackers claimed to have stolen as many as 53,000 credit card numbers

Source: http://www.webappsec.org/projects/whid/list_id_2006-3.shtml

OWASP Top Security Risks

Cross Site Scripting (XSS)

Injection attack

Insecure Direct Object Reference

Integer Overflow

OWASP Top Security Risks

Cross Site Scripting (XSS)

Injection attack

Insecure Direct Object Reference

Integer Overflow

Insecure Direct Object Reference

- What is Insecure Direct Object Reference?
- Occurs when a direct reference to a file, directory, database record, etc. is exposed to users
- Typically exposed in the URL as a querystring or form parameter
- Hacker can manipulate reference to access other objects

Insecure Direct Object Reference

DEMO

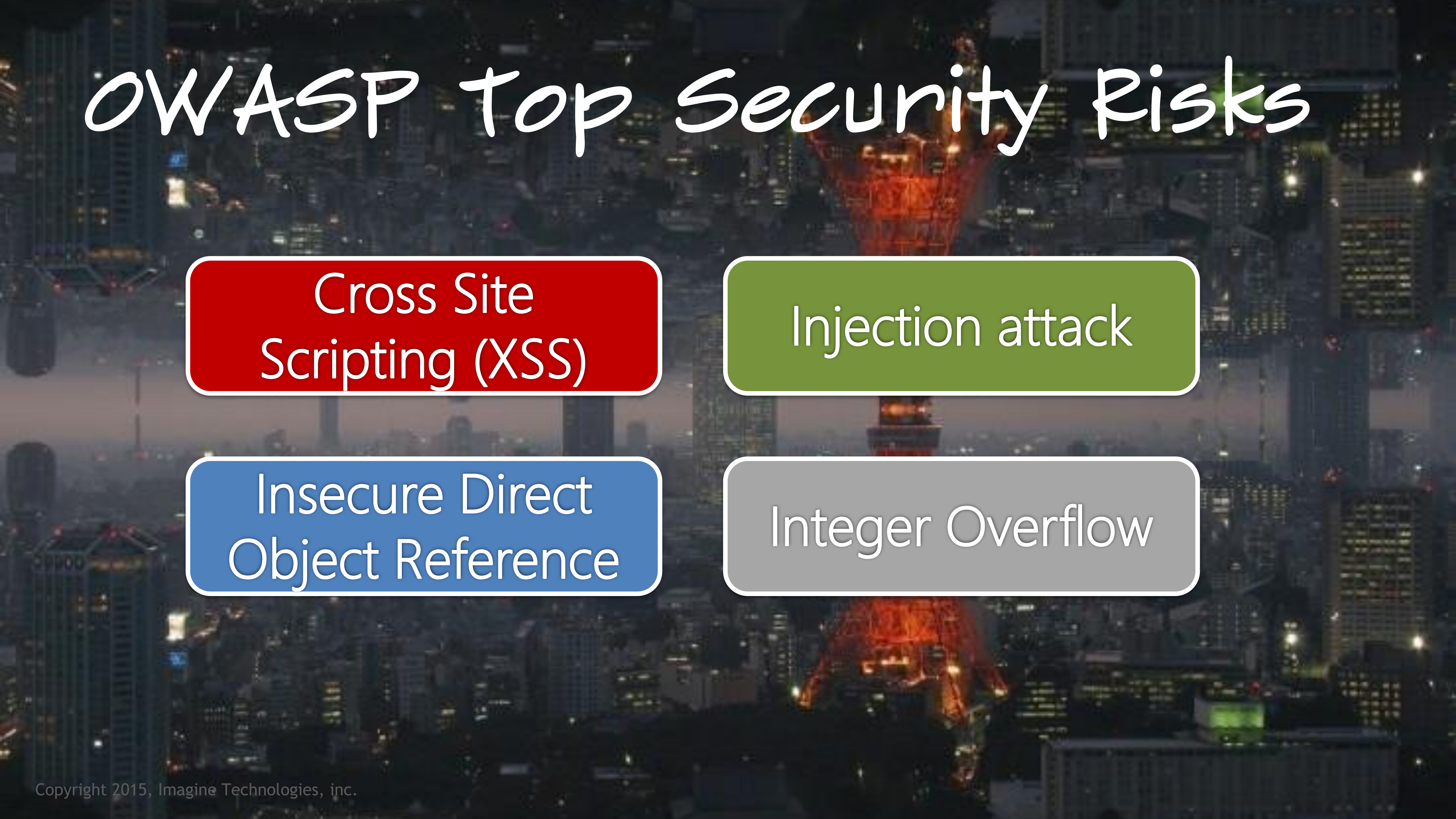
Insecure Direct Object Reference

- Potential Risks
- Attacker can access other files or resources on the server
  - Web.Config – contains database connection and user account info
  - SAM file – Holds the user names and password hashes for every account on the local machine
  - This data can be used to create additional attacks

Insecure Direct Object Reference

- Steps To Mitigate
- Avoid directly referencing objects wherever possible
- Use an index to assign a unique id, then reference the id
- If a direct reference must be used employ methods to ensure only authorized objects are shown
- Encrypt sensitive sections in web.config

OWASP Top Security Risks



- Cross Site Scripting (XSS)
- Injection attack
- Insecure Direct Object Reference
- Integer Overflow

OWASP Top Security Risks

Cross Site Scripting (XSS)

Injection attack

Insecure Direct Object Reference

Integer Overflow

Integer Overflow

- What is Integer Overflow?
- Occurs when an calculation causes an integer to exceed the max or min value allowed by its data type

Integer Overflow

- Potential Risks
- Data corruption
- Application crashes, instability
- Execution of arbitrary code

Preventing Integer Overflow

- How To Mitigate
- Validate user input
  - Check for min and max values
- Use the correct data type
- Execute your code in a checked context

Integer Overflow

- Real World Example
- Apple’s OS X operating system contained a vulnerability which could be exploited remotely by an attacker to compromise a user's system.
- The ffs_mountfs() method was vulnerable to an integer overflow which could potentially allow abritrary code to be executed.

Source:
[illegible]

Integer Overflow

DEMO

Defense in Depth

- - Multi-layered defense
  - Physical – Network – O/S – Services – Applications – Requests
- - Secure by design, by default and in deployment
- - Minimize attack surface
- - Secure defaults
- - Principle of Least Privilege

Secure Coding

Web Security

Coding Practices

- Data validation
- Data type checking
- Proper encoding
- Anti-tampering measures

Data Access Strategies

- Use of roles to ensure weakest possible account
- Parameterized commands

Effective Administration

- Control access to resources with proper authentication
- Rigorous password policies

Session Summary

- Validate Input / Encode Output (Anti-XSS library)
- Parameterize SQL Queries
- Least privilege Account
- Execute in a checked context
- ViewStateUserKey = Session.ID
- Reference objects Indirectly
- Encrypt Web.Config

Code techniques we covered

- OWASP Top 10 Exploit List www.owasp.org
- AntiXSS Encoding
- SQL Parameterization & LINQ
- Indirect Reference Map
- Encrypting sensitive data in Web.config
- CHECK on calculations

Where can I get more info?

- Visit my site www.BenkoTIPS.com
  - Resources from today's talk
  - Blog from this event - www.BenkoTIPS.com
  - Webcasts
  - Downloads
  - More!

BONUS Content...CSRF

Cross Site Scripting (XSS)

Injection attack

Insecure Direct Object Reference

Integer Overflow

Cross Site Request Forgery

Cross Site Request Forgery

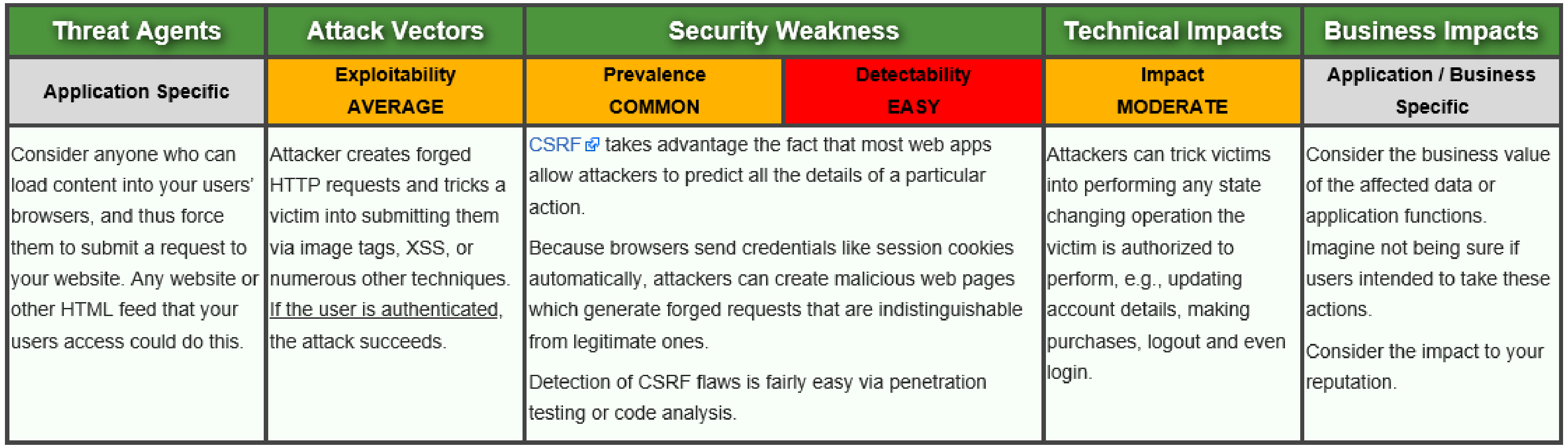
Overview

- Tricks a logged-on victim’s browser to send a request to a vulnerable web application
- Request is sent by the victim, not the attacker
- Can be difficult to detect
- Also known as “One-Click” vulnerability

OWASP Thread Assessment

A8 Cross-Site Request Forgery (CSRF)

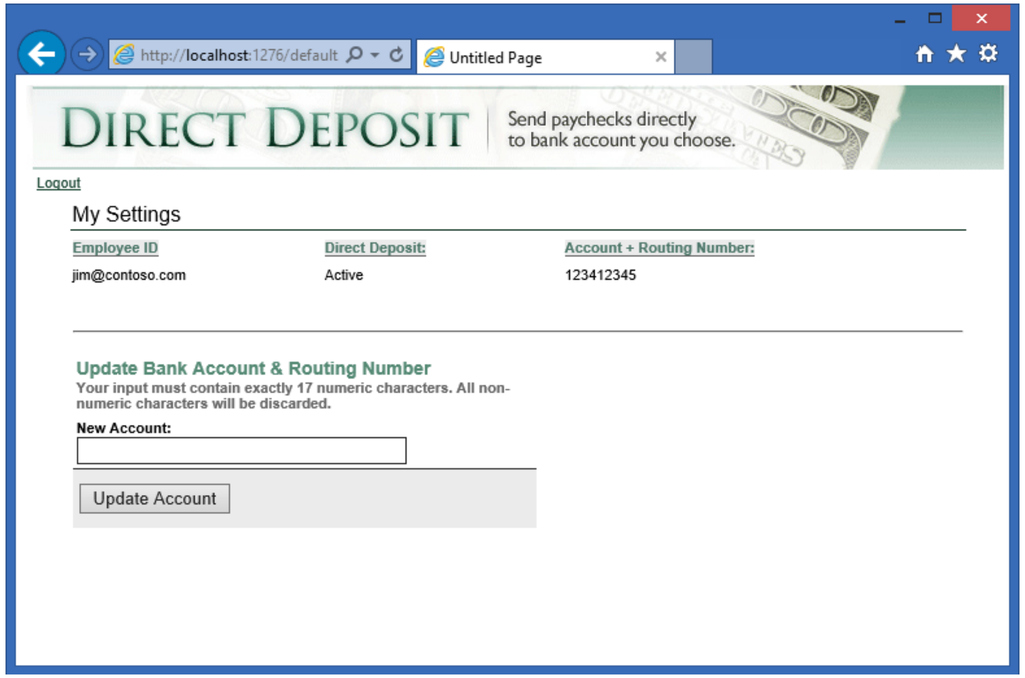
An attack where a page on our site (victim) is sent an HTTP request to complete submission of data or function from the exploiter's code. The user is tricked to load or click a site that sends the attack.



Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence COMMON	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anyone who can load content into your users' browsers, and thus force them to submit a request to your website. Any website or other HTML feed that your users access could do this.	Attacker creates forged HTTP requests and tricks a victim into submitting them via image tags, XSS, or numerous other techniques. If the user is authenticated, the attack succeeds.	CSRF takes advantage the fact that most web apps allow attackers to predict all the details of a particular action. Because browsers send credentials like session cookies automatically, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones. Detection of CSRF flaws is fairly easy via penetration testing or code analysis.		Attackers can trick victims into performing any state changing operation the victim is authorized to perform, e.g., updating account details, making purchases, logout and even login.	Consider the business value of the affected data or application functions. Imagine not being sure if users intended to take these actions. Consider the impact to your reputation.

Example

User logs into bank and remains authenticated


http://localhost:1276/default
Untitled Page
DIRECT DEPOSIT
Send paychecks directly to bank account you choose.
Logout

My Settings

Employee ID	Direct Deposit:	Account + Routing Number:
jim@contoso.com	Active	123412345

Update Bank Account & Routing Number

Your input must contain exactly 17 numeric characters. All non-numeric characters will be discarded.

New Account:

User identity is cached in the browser

The attacker depends on the authenticated session

Example

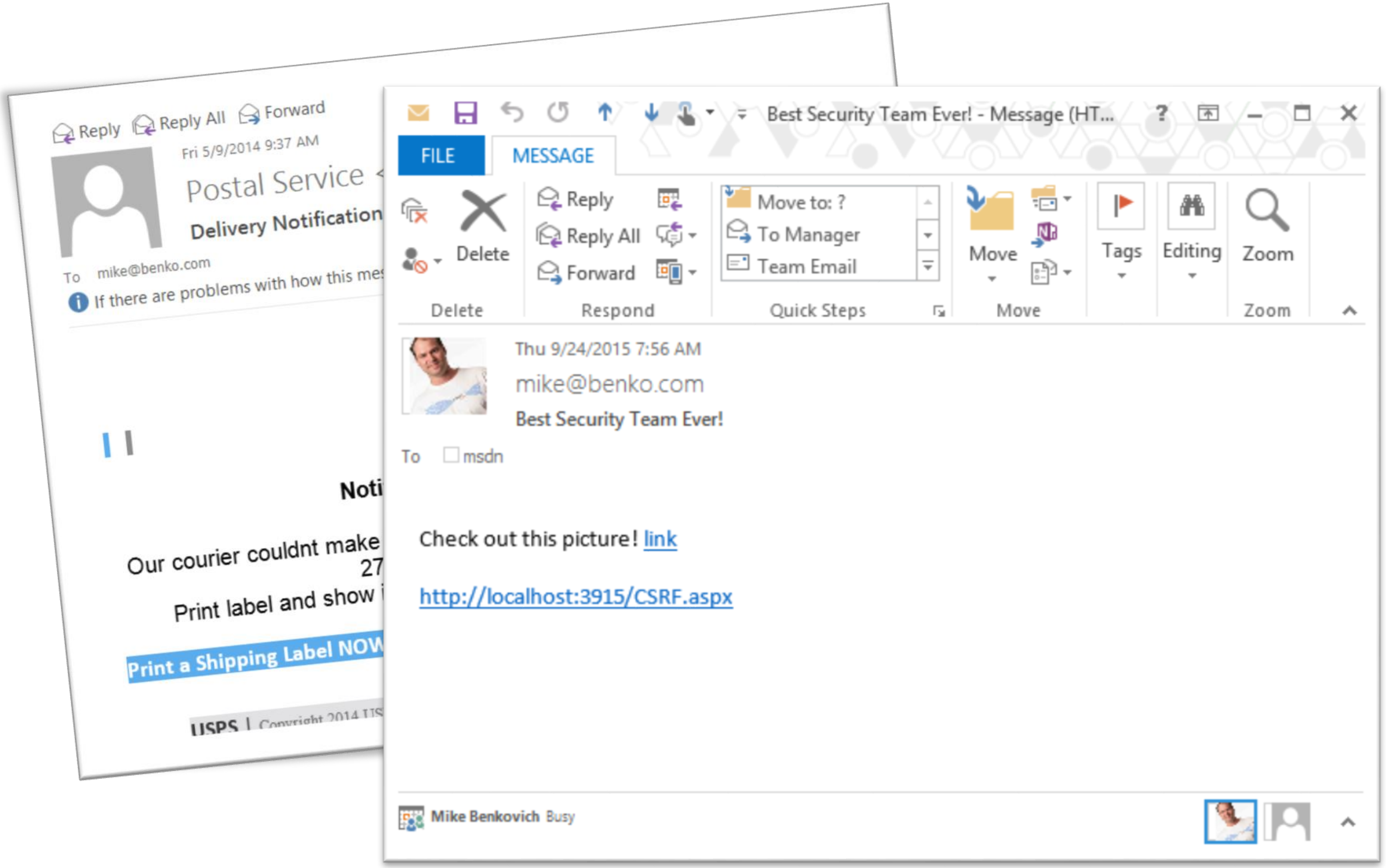
The hacker identifies the request to move funds

```
http://example.com/app/transferFunds?amount=1500
&destinationAccount=4673243243
```

Then creates an exploit page with image tag that embeds CSRF and sends it in a phishing attack

```
 <img src="http://example.com/app/transferFunds?
          amount=1500&destinationAcct=123-4567-89”
          width="0" height="0" />
```

Phishing email sent to user…



Reply Reply All Forward
Fri 5/9/2014 9:37 AM
Postal Service <
Delivery Notification
To mike@benko.com
If there are problems with how this mes

Noti

Our courier couldnt make
27
Print label and show i
Print a Shipping Label NOW

USPS | Copyright 2014 US

Best Security Team Ever! - Message (HT...

FILE MESSAGE

Delete
Reply
Reply All
Forward
Respond
Move to: ?
To Manager
Team Email
Quick Steps
Move
Tags
Editing
Zoom

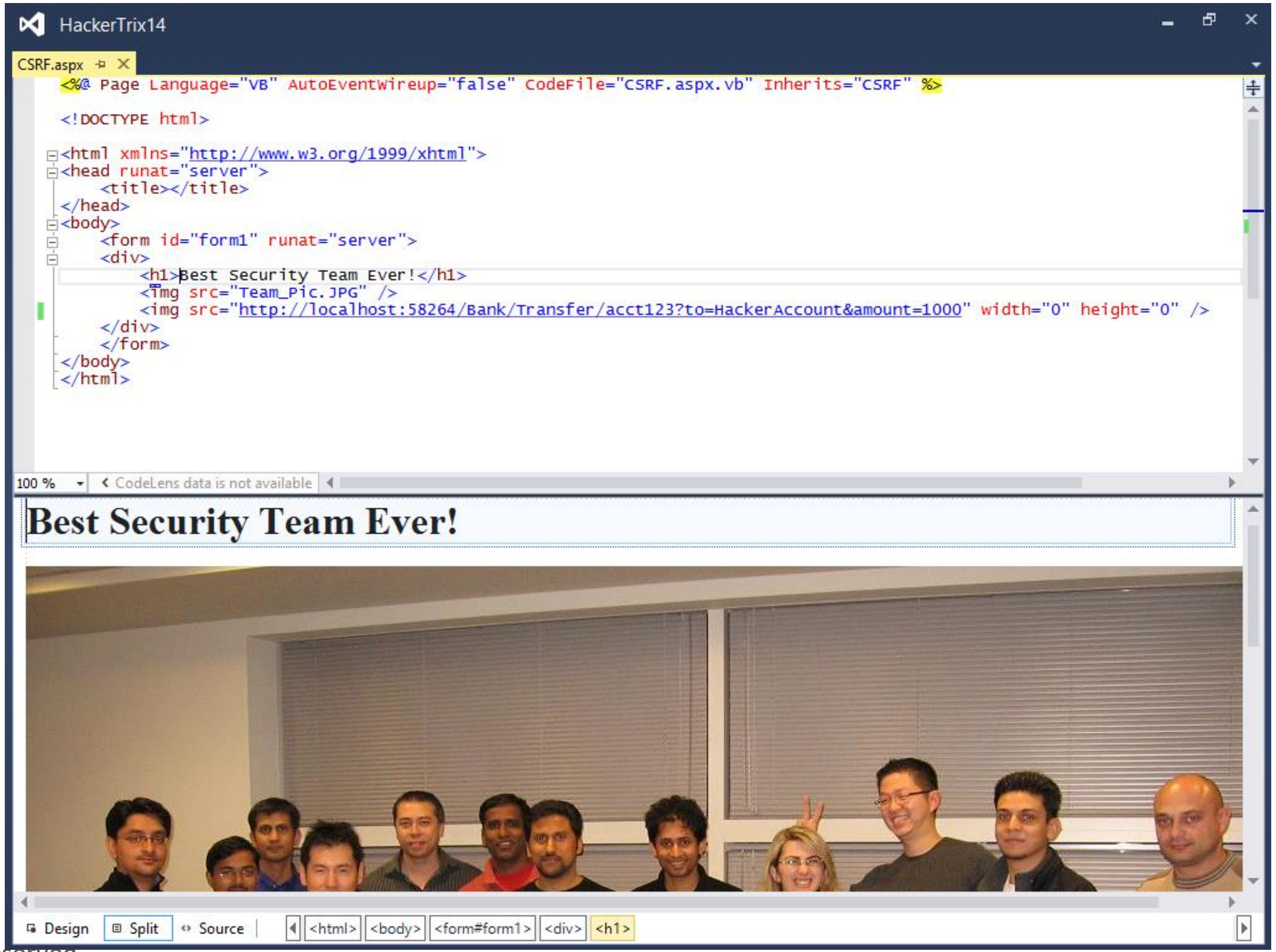
Thu 9/24/2015 7:56 AM
mike@benko.com
Best Security Team Ever!
To msdn

Check out this picture! link

http://localhost:3915/CSRF.aspx

Mike Benkovich Busy

Code in page includes CSRF hack



```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="CSRF.aspx.vb" Inherits="CSRF" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <h1>Best Security Team Ever!</h1>
        <img src="Team_Pic.JPG" />
        <img src="http://localhost:58264/Bank/Transfer/acct123?to=HackerAccount&amount=1000" width="0" height="0" />
    </div>
    </form>
</body>
</html>
```

Best Security Team Ever!

Cross Site Request Forgery (CSRF)

Depends on

- Authenticated user that has a valid state
- Site with malicious intent has code on it which sends request to our site to execute a function
  - Hidden or cloaked as iframe or img or other src
- Because we're already authenticated it executes

Cross Site Request Forgery

How to Mitigate

- Include unique token which the server validates when a request is received
  - WebForms: ViewStateUserKey
    - Must use unique value for each user
    - Recommended:
      ViewStateUserKey = Session.ID
  - MVC: AntiForgeryToken
    - Add in view @Html.AntiForgeryToken()
    - Annotation in controller [ValidateAntiForgeryToken]
- Require user confirmation with a shared secret

Cross Site Request Forgery

Potential Risks

- Exposes victims private information to attacker
- Attacker can alter data, make purchases, retrieve account info.
- Victim is usually unaware any changes have taken place

Cross Site Request Forgery

DEMO

Cross Site Request Forgery

Real World Example

- A security flaw at FTD.com made it possible to access customer data simply by copying a cookie from one computer to another.
- In addition, sequential values were used as identifiers, making it easier to guess the numbers of other valid cookies.

Source: http://www.news.com/2100-1017-984585.html