Principles of Secure Web Application Development

Alan Lai & Scott Kirkland

Common vulnerabilities in Web Applications

Alan Lai & Scott Kirkland

Alan Lai

- Application Developer
  - College of Agricultural & Environmental Sciences Dean's Office
- My blog: http://anlai.wordpress.com
- Written several web applications
  - https://github.com/anlai

Scott Kirkland

- Application Architect
  - College of Agricultural & Environmental Sciences Dean's Office
- My blog
  - http://weblogs.asp.net/srkirkland/
- Open source coder
  - https://github.com/srkirkland
  - https://github.com/ucdavis
- Co-founded the local .NET User Group

CC Image courtesy of Legozilla on Flickr
http://www.flickr.com/photos/legozilla/3553485856/in/photostream/

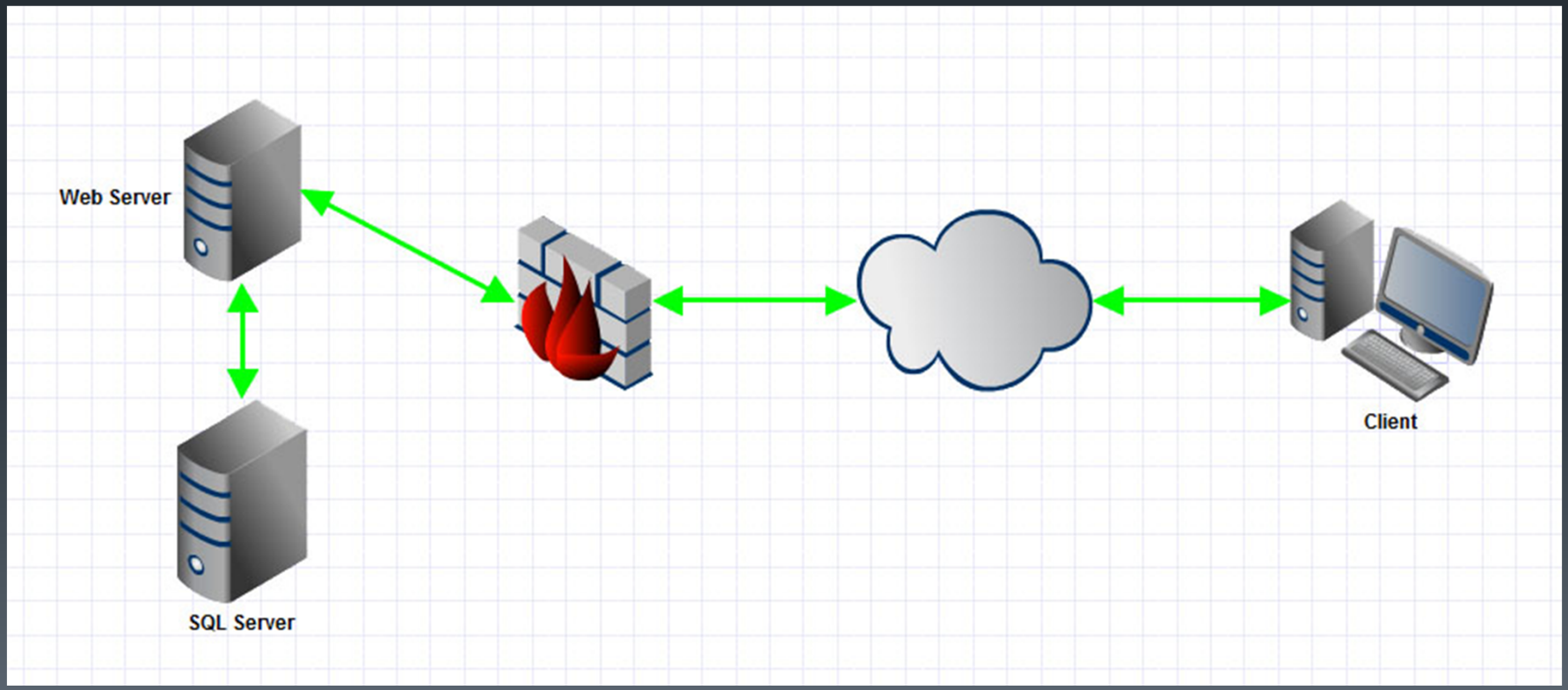
Contents

- SQL Encryption
- SSL
- Web Service Security
- Cross Site Scripting (XSS)
- Cross Site Request Forgery (CSRF)
- Insecure Direct Object Reference

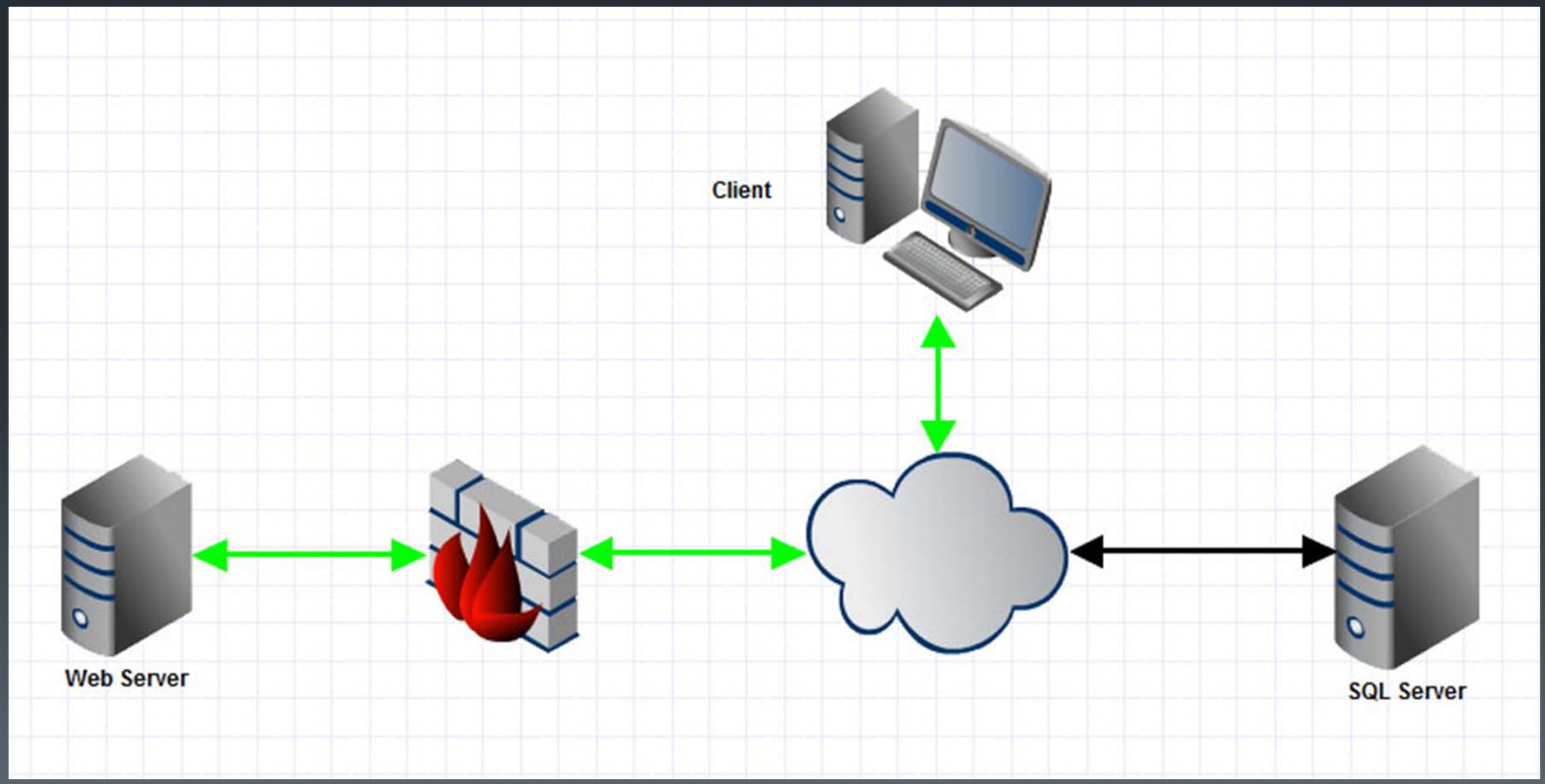
SQL Encryption

- Cell Level Encryption
- Transparent Data Encryption (TDE)
- Connection to SQL Server

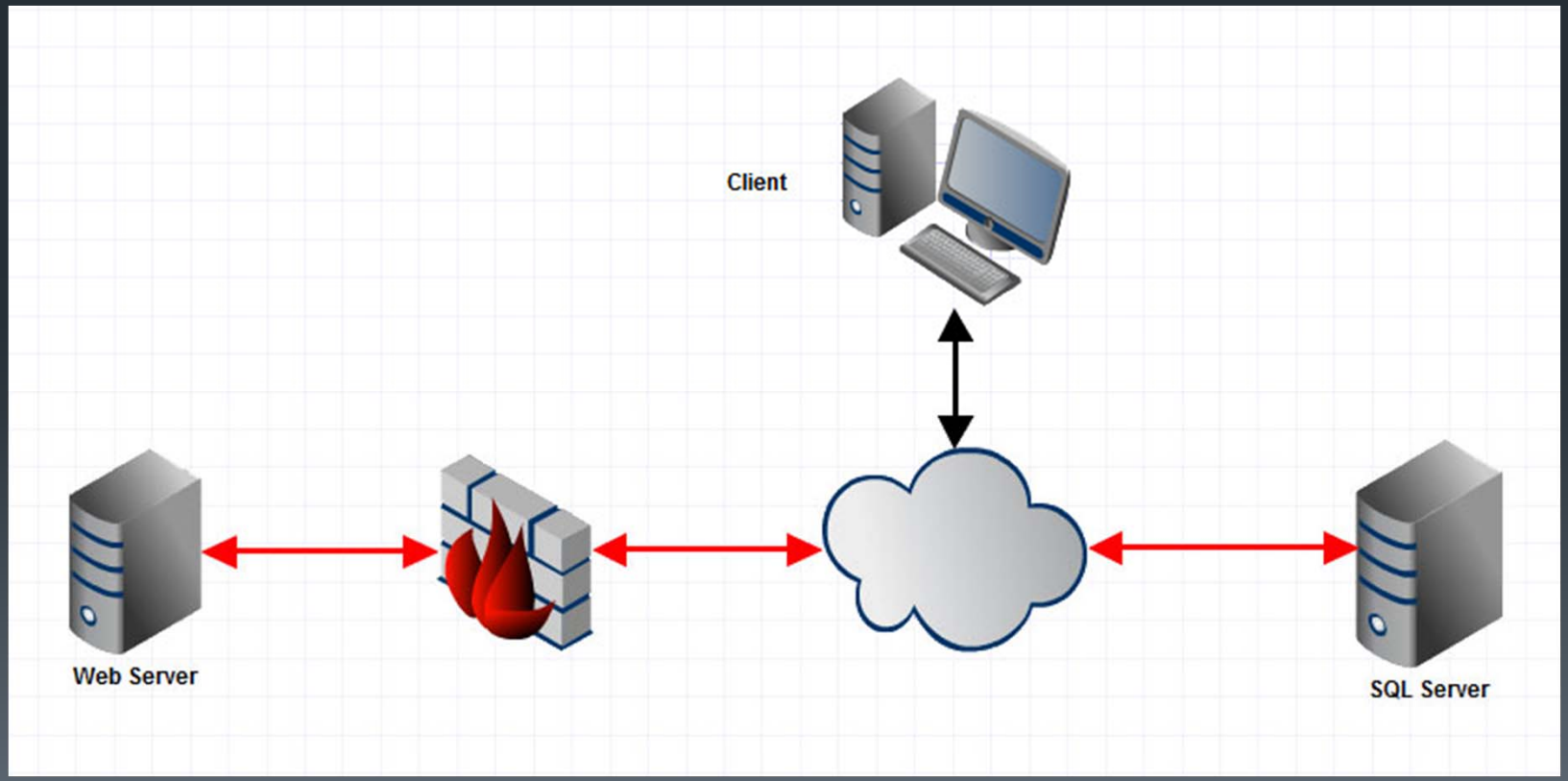
Traditional Architecture

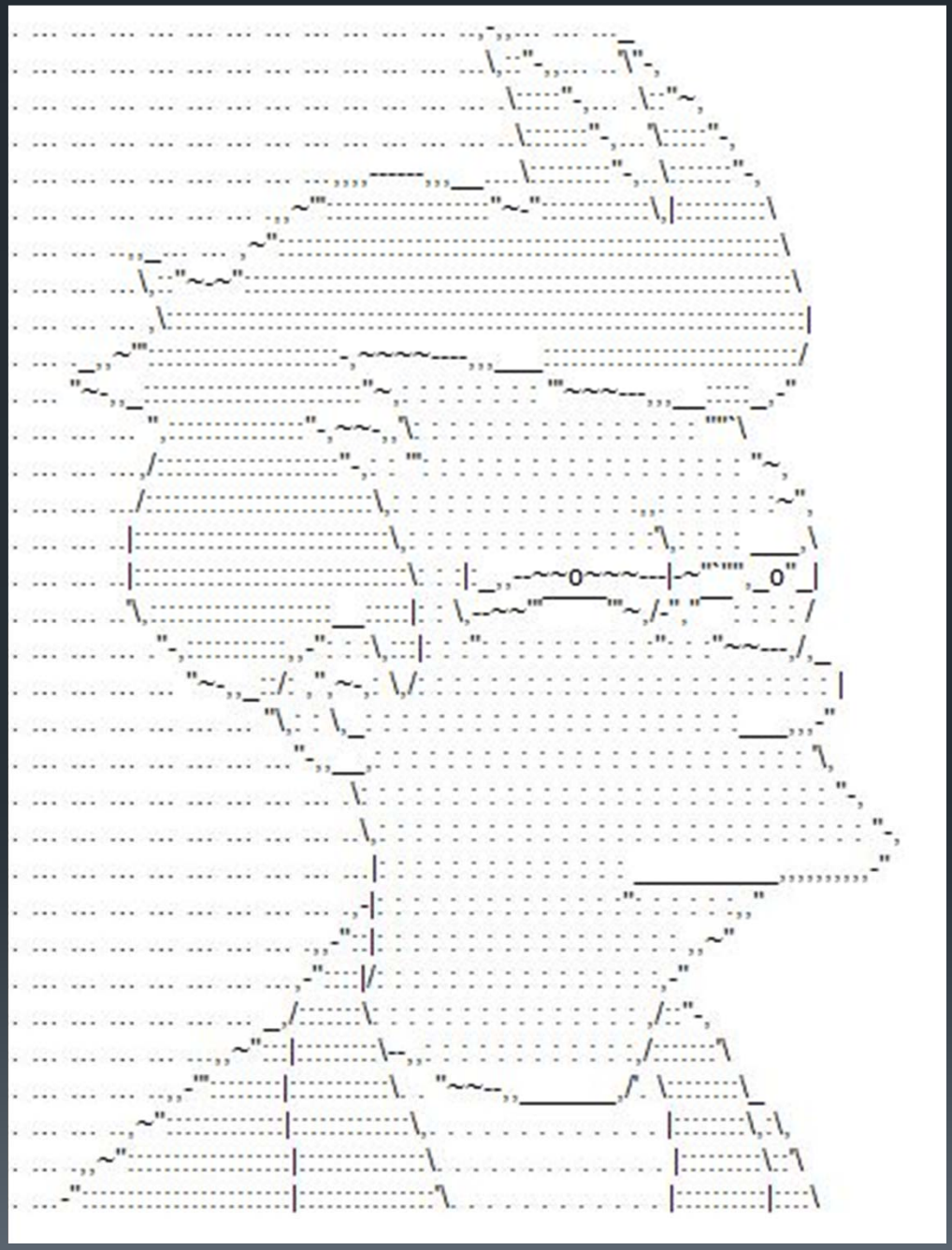


Cloud Based Solution



Cloud Based Problem

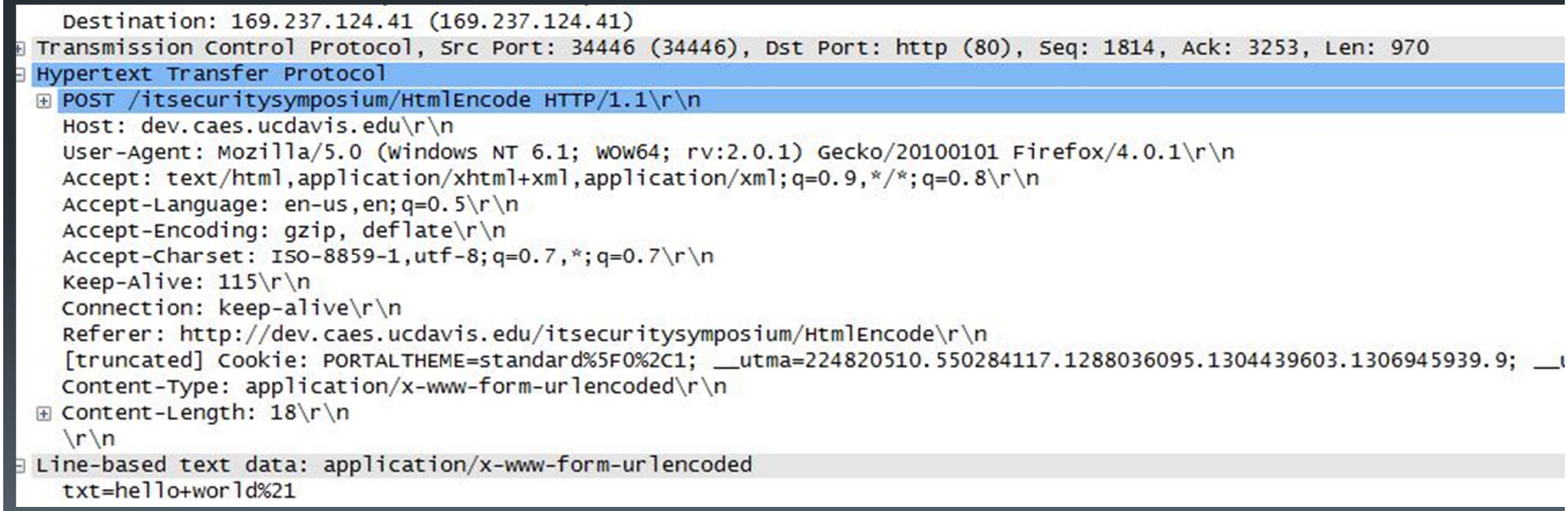




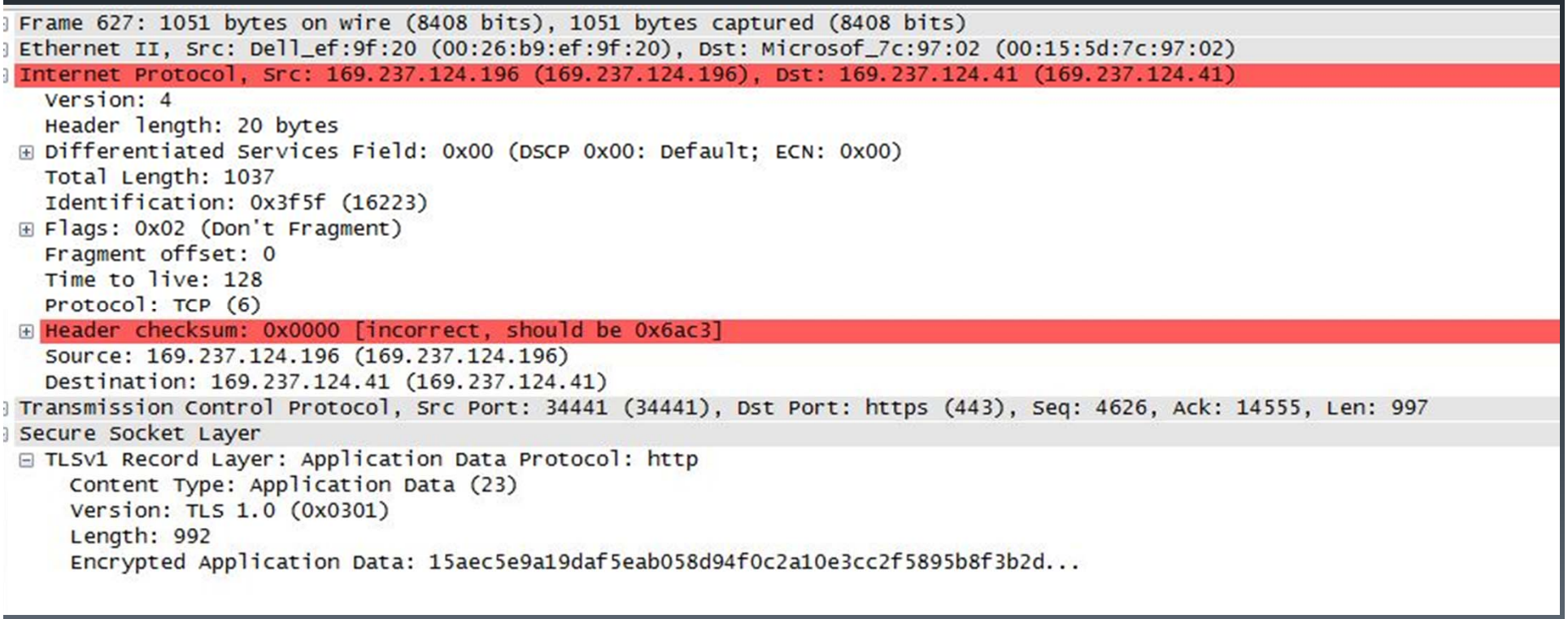
SSL

- SSL should be used with sensitive information
- Lots of sites only encrypt authentication
- Vulnerable to Session Hijacking
- When moving to SSL consider:
  - Page Assets
  - Performance
  - Caching

Non-SSL


Destination: 169.237.124.41 (169.237.124.41)
Transmission Control Protocol, Src Port: 34446 (34446), Dst Port: http (80), Seq: 1814, Ack: 3253, Len: 970
Hypertext Transfer Protocol
POST /itsecuritysymposium/HtmlEncode HTTP/1.1\r\n
Host: dev.caes.ucdavis.edu\r\n
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:2.0.1) Gecko/20100101 Firefox/4.0.1\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
Referer: http://dev.caes.ucdavis.edu/itsecuritysymposium/HtmlEncode\r\n
[truncated] Cookie: PORTALTHEME=standard%5F0%2C1; __utma=224820510.550284117.1288036095.1304439603.1306945939.9;
Content-Type: application/x-www-form-urlencoded\r\n
Content-Length: 18\r\n
\r\n
Line-based text data: application/x-www-form-urlencoded
txt=hello+world%21

SSL


Frame 627: 1051 bytes on wire (8408 bits), 1051 bytes captured (8408 bits)
Ethernet II, Src: Dell_ef:9f:20 (00:26:b9:ef:9f:20), Dst: Microsof_7c:97:02 (00:15:5d:7c:97:02)
Internet Protocol, Src: 169.237.124.196 (169.237.124.196), Dst: 169.237.124.41 (169.237.124.41)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
Total Length: 1037
Identification: 0x3f5f (16223)
Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 128
Protocol: TCP (6)
Header checksum: 0x0000 [incorrect, should be 0x6ac3]
Source: 169.237.124.196 (169.237.124.196)
Destination: 169.237.124.41 (169.237.124.41)
Transmission Control Protocol, Src Port: 34441 (34441), Dst Port: https (443), Seq: 4626, Ack: 14555, Len: 997
Secure Socket Layer
TLSv1 Record Layer: Application Data Protocol: http
Content Type: Application Data (23)
Version: TLS 1.0 (0x0301)
Length: 992
Encrypted Application Data: 15aec5e9a19daf5eab058d94f0c2a10e3cc2f5895b8f3b2d...

Session Hijacking

- Session cookie is passed for every request
- Cookie is stolen when not using SSL
- Attacker uses stolen cookie to impersonate user

- Firesheep is a firefox extension that can do this easily.

- Big sites like Facebook, Twitter and Github have moved to using SSL to prevent this attack.

Assets and SSL

- Images, CSS and script files
- Need to be transmitted using SSL
- Causes problems with Content Delivery Networks (CDN)

SSL Performance

- SSL Handshakes take more time
- Overhead for encryption/decryption

SSL Caching

- Sites can’t be cached between server and client
- Files can still be cached on the client and the server

Web Service Security

- Example Services
  - Netflix Odata (Read only) – no security
    - http://odata.netflix.com/Catalog/Titles?$filter=Instant/Available%20eq%20true
  - Flickr API (Read/Write) – authentication token
    - http://api.flickr.com/services/rest/?method=flickr.test.echo&name=value&api_key=blank
- Reasons to Secure with token
  - Only users with token can access a read/write service
  - Rate throttling
- Use SSL to protect token from replay attack

Cross Site Scripting (XSS)

- Malicious code can be injected onto a trusted web site
- Victim views site which executes malicious code

- Gateway/Attack vector for other attacks
  - Script Injection
  - Cookie stealing
  - Content rewriting
  - Local storage hijacking
  - CSRF

- Twitter / Facebook has these vulnerabilities

XSS Demo!

Cookie Policy

- Cookies can be used to store any piece of information.
- Cookies can be accessed through Javascript.
  - Fairly trivial to steal information stored in cookie
- Can be solved by setting httponly on cookie to true.

Html Encode

- Need to safely display content entered by users
- Encoding examples:
  - becomes <div>
  - <script type="javascript"> becomes <script type="…..
- Consult your server side language.

st Forgery

InternetNews.com >> Security

Facebook Hit With New CSRF Worm

Latest exploit replicates itself through bogus wall postings; security researchers wonder what else is out there.

November 23, 2009
By Kenneth Corbin: *More stories by this author:*

Security researc
worm spreading
people out to ac
automatically re
profile pages.

Like so much ma
itself as a produ

An image of a ne
a user's wall alo
activities of that
in the image is a link, bracketed by the text:
"Want 2 C Something Hot?" and "Click da'
button, baby."

Myspace CSRF and XSS Worm (Samy)

THU, 13 OCT 2005 AT 17:13:33 GMT 13 COMMENTS TWEETS

In the comments to my article on CSRF, someone questioned whether CSRF is really anything worth worrying about. Rather than give a hypothetical example, I can point to a real one that is getting some attention today:

- Myspace Hack Overview
- Myspace Hack Technical Details

Zero Day
Ryan Naraine and Dancho Danchev

Mobile
RSS
Email Alerts

1 Comment | Share | Print | Facebook | Twitter | Recommend 4 Votes

Home / News & Blogs / Zero Day

CSRF vulnerability allows Twitter 'follow' abuse

Cross Site Request Forgery

- Demo
  - /Csrf/ controller
  - CSRF of a banking site with both GET and POST examples
  - To counter CSRF, should use POST and must use shared anti-forgery token.

CSRF: Prevention

- Require a secret, user-specific token in all form submissions
- Always require POST for non-idempotent (state-changing) actions
- Non-sufficient solutions
  - Checking “Referrer” header
  - Verifying X-Requested-With (RoR & Django)

Insecure Direct Object Reference

- Ex. http://all.hail.hypnotoad.com?message=1
- Malicious user can phish just by changing the id
- Anyone can generate http request/post

- Authorize users when accessing objects
- Even if you have already authorized them on the last page

Thank you!

- Demo Project at https://github.com/srkirkland/ITSecuritySymposium
- Alan: anlai@ucdavis.edu
- Scott: srkirkland@ucdavis.edu

- Questions?