TDT4237 Software Security

OWASP Testing Guide - part one

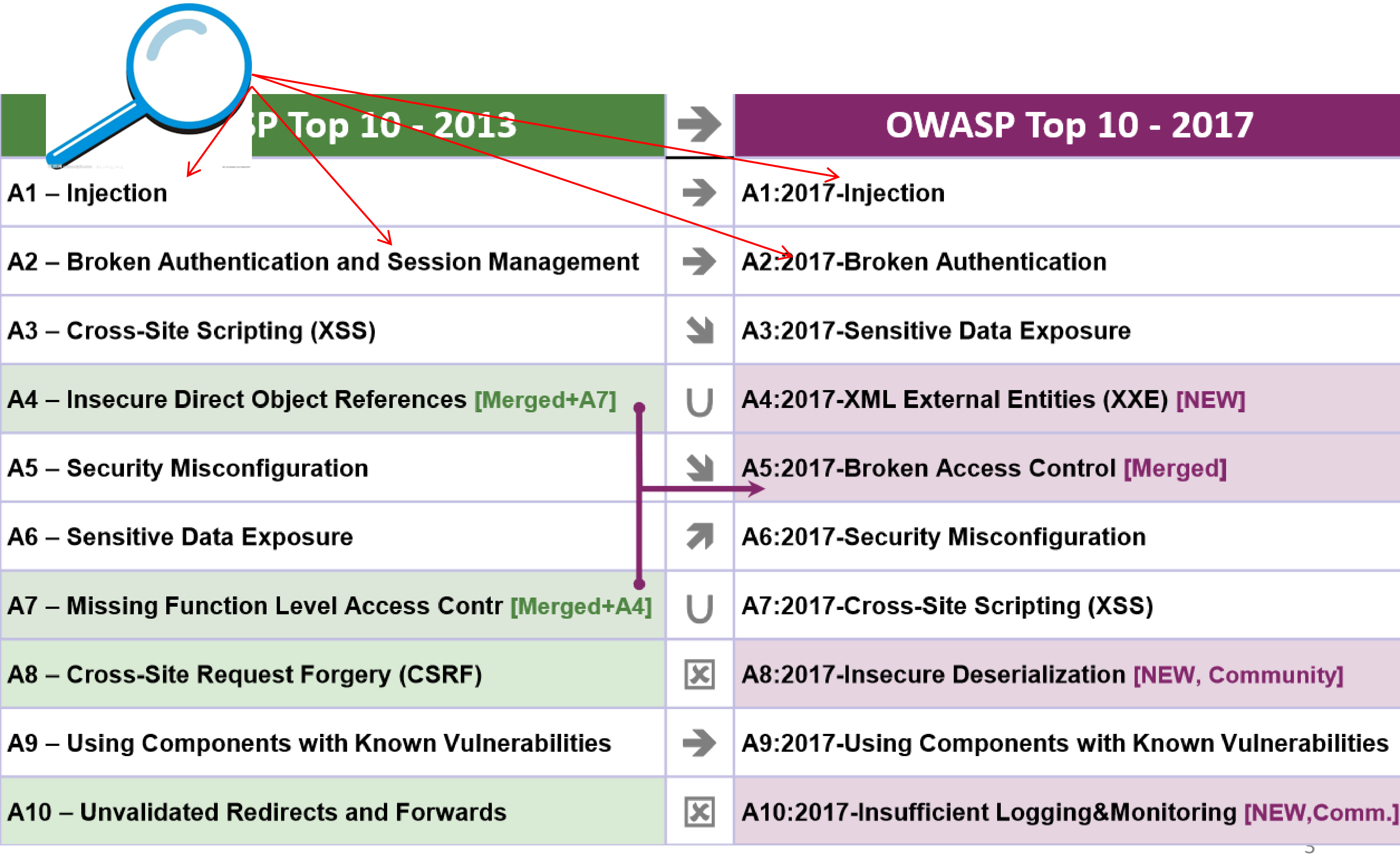
- Information gathering
- Injection attacks
- Session management attacks

Practical issues

- We need a reference group

If you are interested in , send email to jingyue.li@ntnu.no

10 Most Critical Web Application Security Risks



P Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Information gathering

- Why information gathering?
  - Attacker
    - A map to attack
    - Look for low hanging fruit
    - Improve efficiency
  - Developer/internal tester
    - Decide test scope, coverage, prioritization
    - Improve test efficiency

The more you know about the application's structure, the better you can plan your tests!

What information to gather

- Application structure, e.g., page map
- Data flow within the application, e.g.,
  - Parameters and value
  - Get and post

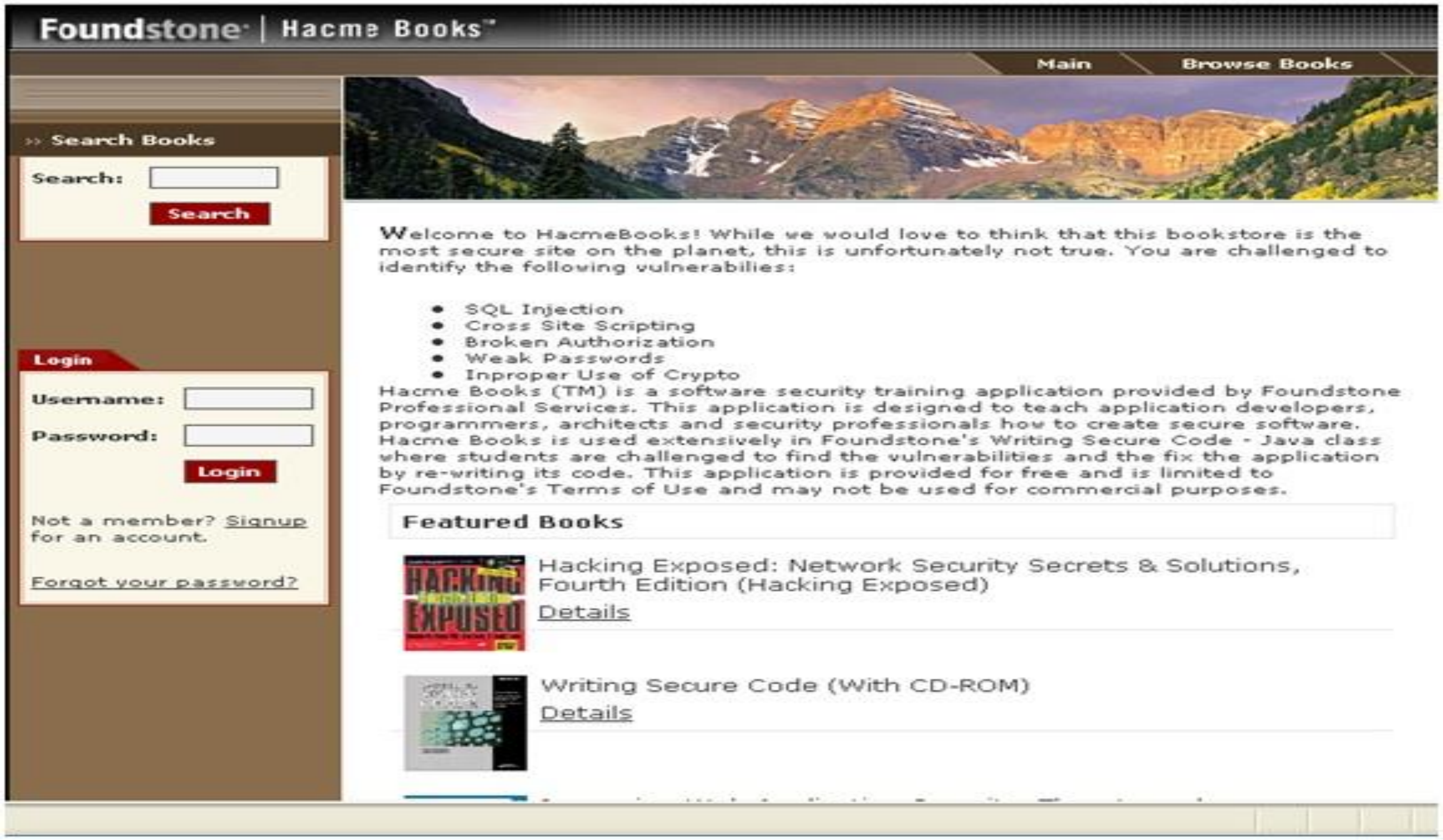
- Always start manually
- Use tools to complete

A good page map includes

- All pages you have found in the application
  - Including subdomains
- Any external links
- Trust zones
  - Needs authentication vs. open
- Any parameters passed

Page map example - Hacmebooks

Hacme Books is representative of real-world J2EE scenarios and demonstrates the security problems that can potentially arise in these applications.



Not a member? Signup for an account.

Forgot your password?

Welcome to HacmeBooks! While we would love to think that this bookstore is the most secure site on the planet, this is unfortunately not true. You are challenged to identify the following vulnerabilies:

- SQL Injection
- Cross Site Scripting
- Broken Authorization
- Weak Passwords
- Inproper Use of Crypto

Hacme Books (TM) is a software security training application provided by Foundstone Professional Services. This application is designed to teach application developers, programmers, architects and security professionals how to create secure software. Hacme Books is used extensively in Foundstone's Writing Secure Code - Java class where students are challenged to find the vulnerabilities and the fix the application by re-writing its code. This application is provided for free and is limited to Foundstone's Terms of Use and may not be used for commercial purposes.

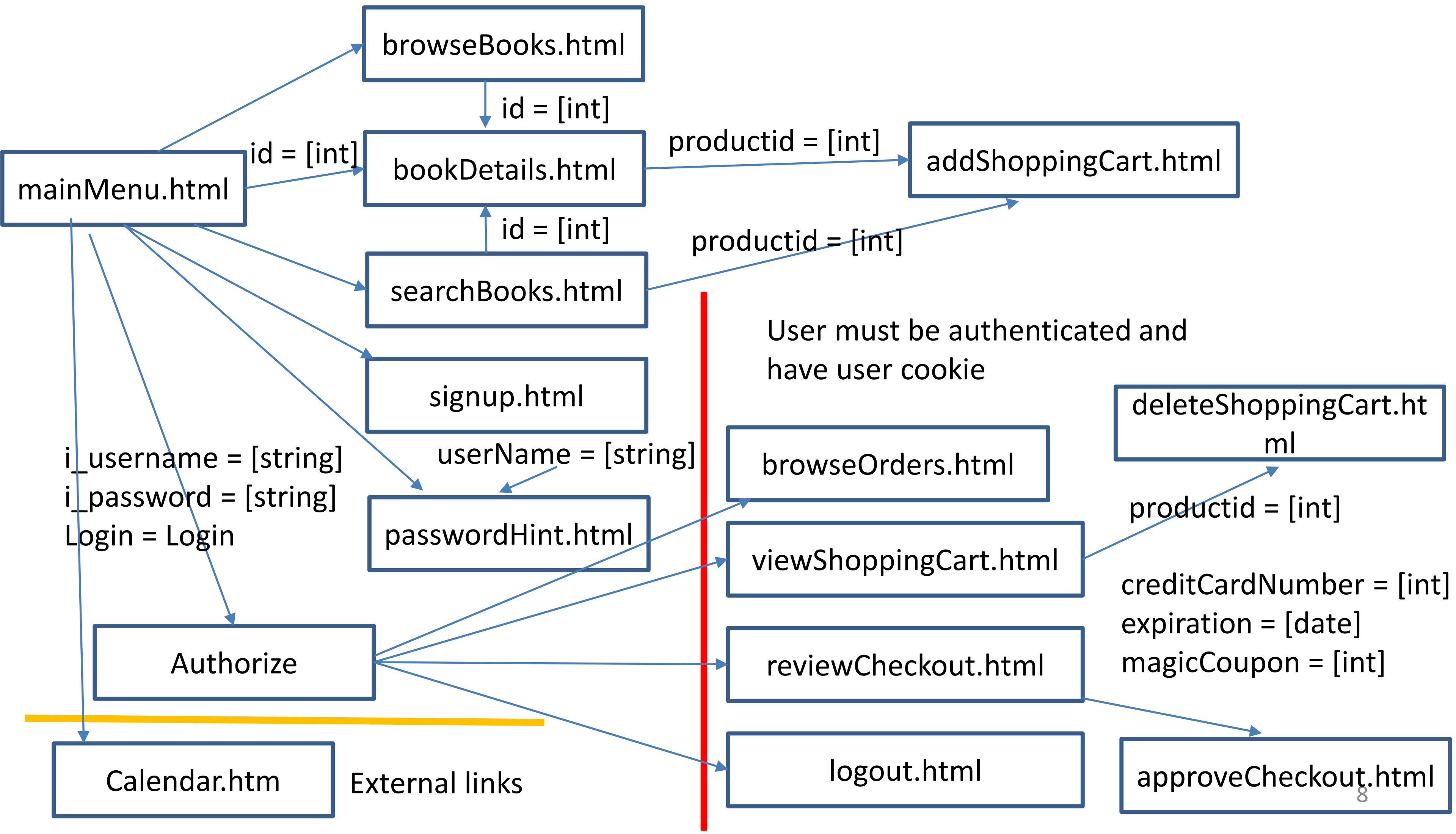
Featured Books

Hacking Exposed: Network Security Secrets & Solutions, Fourth Edition (Hacking Exposed)
Details

Writing Secure Code (With CD-ROM)
Details

https://webapppentest.wordpress.com/2012/11/26/hacme-books-week-1/

Parts of Hacmebooks page map



Tools for making page map

- Why use a web proxy?
  - To capture and examine requests
  - To manipulate requests
    - To learn more about the application
  - Can also be used for attacks

KALI LINUX

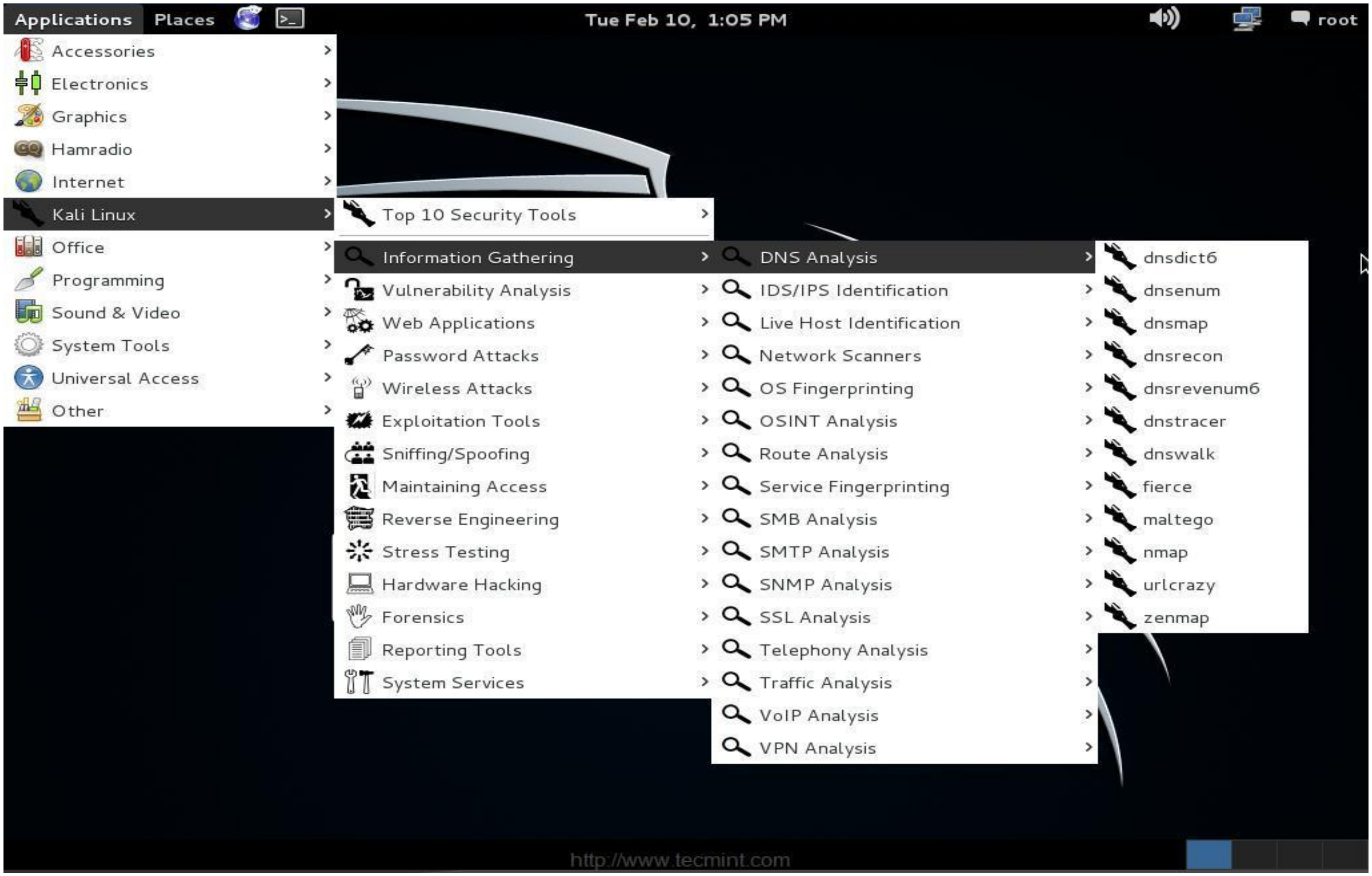
Tool set

Web debugging proxy

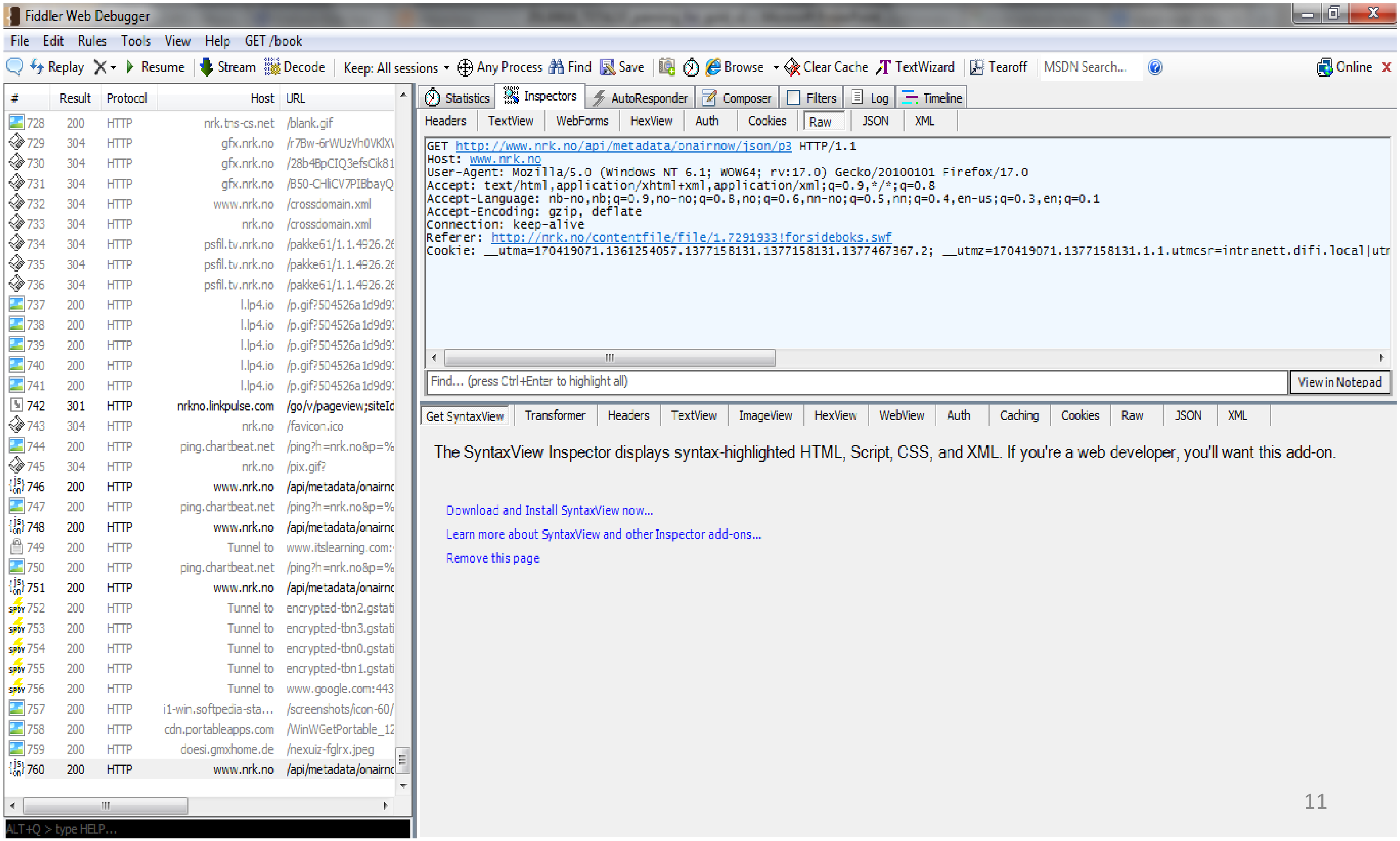
HTTrack

Web mirroring

Kali Linux


Applications
Places
Tue Feb 10, 1:05 PM
root
Accessories
Electronics
Graphics
Hamradio
Internet
Kali Linux
Office
Programming
Sound & Video
System Tools
Universal Access
Other
Top 10 Security Tools
Information Gathering
Vulnerability Analysis
Web Applications
Password Attacks
Wireless Attacks
Exploitation Tools
Sniffing/Spoofing
Maintaining Access
Reverse Engineering
Stress Testing
Hardware Hacking
Forensics
Reporting Tools
System Services
DNS Analysis
IDS/IPS Identification
Live Host Identification
Network Scanners
OS Fingerprinting
OSINT Analysis
Route Analysis
Service Fingerprinting
SMB Analysis
SMTP Analysis
SNMP Analysis
SSL Analysis
Telephony Analysis
Traffic Analysis
VoIP Analysis
VPN Analysis
dnsdict6
dnsenum
dnsmap
dnsrecon
dnsrevenum6
dnstracer
dnswalk
fierce
maltego
nmap
urlcrazy
zenmap
http://www.tecmint.com

Fiddler tool


Fiddler Web Debugger
File Edit Rules Tools View Help GET /book
Replay Resume Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search... Online
Statistics Inspectors AutoResponder Composer Filters Log Timeline
Headers TextView WebForms HexView Auth Cookies Raw JSON XML
Find... (press Ctrl+Enter to highlight all)
View in Notepad
Get SyntaxView Transformer Headers TextView ImageView HexView WebView Auth Caching Cookies Raw JSON XML
ALT+Q > type HELP...

#	Result	Protocol	Host	URL
728	200	HTTP	nrk.tns-cs.net	/blank.gif
729	304	HTTP	gfx.nrk.no	/r7Bw-6rWUzVh0VKlX\
730	304	HTTP	gfx.nrk.no	/28b4BpCIQ3efsCik81
731	304	HTTP	gfx.nrk.no	/B50-CHliCV7PIBbayQ
732	304	HTTP	www.nrk.no	/crossdomain.xml
733	304	HTTP	nrk.no	/crossdomain.xml
734	304	HTTP	psfil.tv.nrk.no	/pakke61/1.1.4926.26
735	304	HTTP	psfil.tv.nrk.no	/pakke61/1.1.4926.26
736	304	HTTP	psfil.tv.nrk.no	/pakke61/1.1.4926.26
737	200	HTTP	l.lp4.io	/p.gif?504526a1d9d9
738	200	HTTP	l.lp4.io	/p.gif?504526a1d9d9
739	200	HTTP	l.lp4.io	/p.gif?504526a1d9d9
740	200	HTTP	l.lp4.io	/p.gif?504526a1d9d9
741	200	HTTP	l.lp4.io	/p.gif?504526a1d9d9
742	301	HTTP	nrkno.linkpulse.com	/go/v/pageview;siteId
743	304	HTTP	nrk.no	/favicon.ico
744	200	HTTP	ping.chartbeat.net	/ping?h=nrk.no&p=%
745	304	HTTP	nrk.no	/pix.gif?
746	200	HTTP	www.nrk.no	/api/metadata/onairnc
747	200	HTTP	ping.chartbeat.net	/ping?h=nrk.no&p=%
748	200	HTTP	www.nrk.no	/api/metadata/onairnc
749	200	HTTP	Tunnel to	www.itslearning.com:
750	200	HTTP	ping.chartbeat.net	/ping?h=nrk.no&p=%
751	200	HTTP	www.nrk.no	/api/metadata/onairnc
752	200	HTTP	Tunnel to	encrypted-tbn2.gstati
753	200	HTTP	Tunnel to	encrypted-tbn3.gstati
754	200	HTTP	Tunnel to	encrypted-tbn0.gstati
755	200	HTTP	Tunnel to	encrypted-tbn1.gstati
756	200	HTTP	Tunnel to	www.google.com:443
757	200	HTTP	i1-win.softpedia-sta...	/screenshots/icon-60/
758	200	HTTP	cdn.portableapps.com	/WinWGetPortable_12
759	200	HTTP	doesi.gmxhome.de	/nexuiz-fglrx.jpeg
760	200	HTTP	www.nrk.no	/api/metadata/onairnc

```
GET http://www.nrk.no/api/metadata/onairnow/json/p3 HTTP/1.1
Host: www.nrk.no
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:17.0) Gecko/20100101 Firefox/17.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: nb-no,nb;q=0.9,no-no;q=0.8,no;q=0.6,nn-no;q=0.5,nn;q=0.4,en-us;q=0.3,en;q=0.1
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://nrk.no/contentfile/file/1.7291933!forsideboks.swf
Cookie: __utma=170419071.1361254057.1377158131.1377158131.1377467367.2; __utmz=170419071.1377158131.1.1.utmcsr=intranett.difi.local|utr
```

The SyntaxView Inspector displays syntax-highlighted HTML, Script, CSS, and XML. If you're a web developer, you'll want this add-on.

Download and Install SyntaxView now...

Learn more about SyntaxView and other Inspector add-ons...

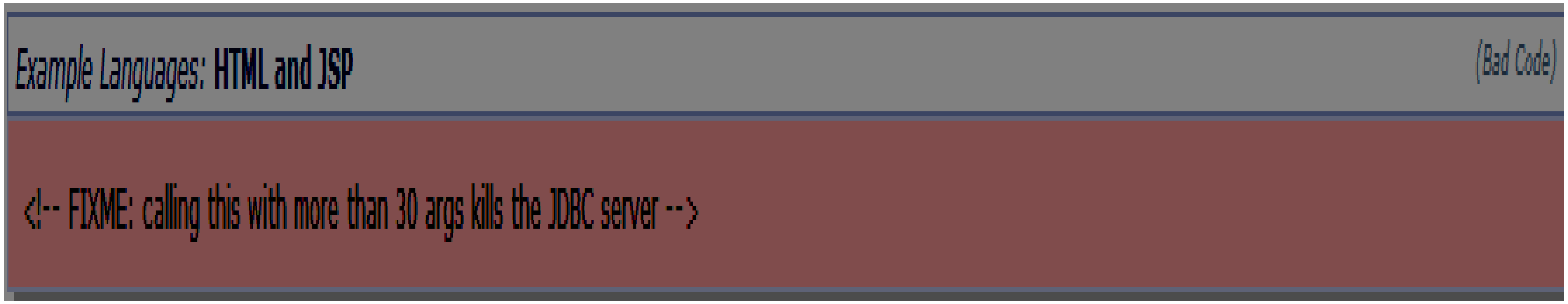
Remove this page

HTML Information leakage

Common Weakness Enumeration

A Community-Developed Dictionary of Software Weakness Types

CWE-615: Information Exposure Through Comments



Example Languages: **HTML and JSP** *(Bad Code)*

```
<!-- FIXME: calling this with more than 30 args kills the JDBC server -->
```

What information to gather (cont’)

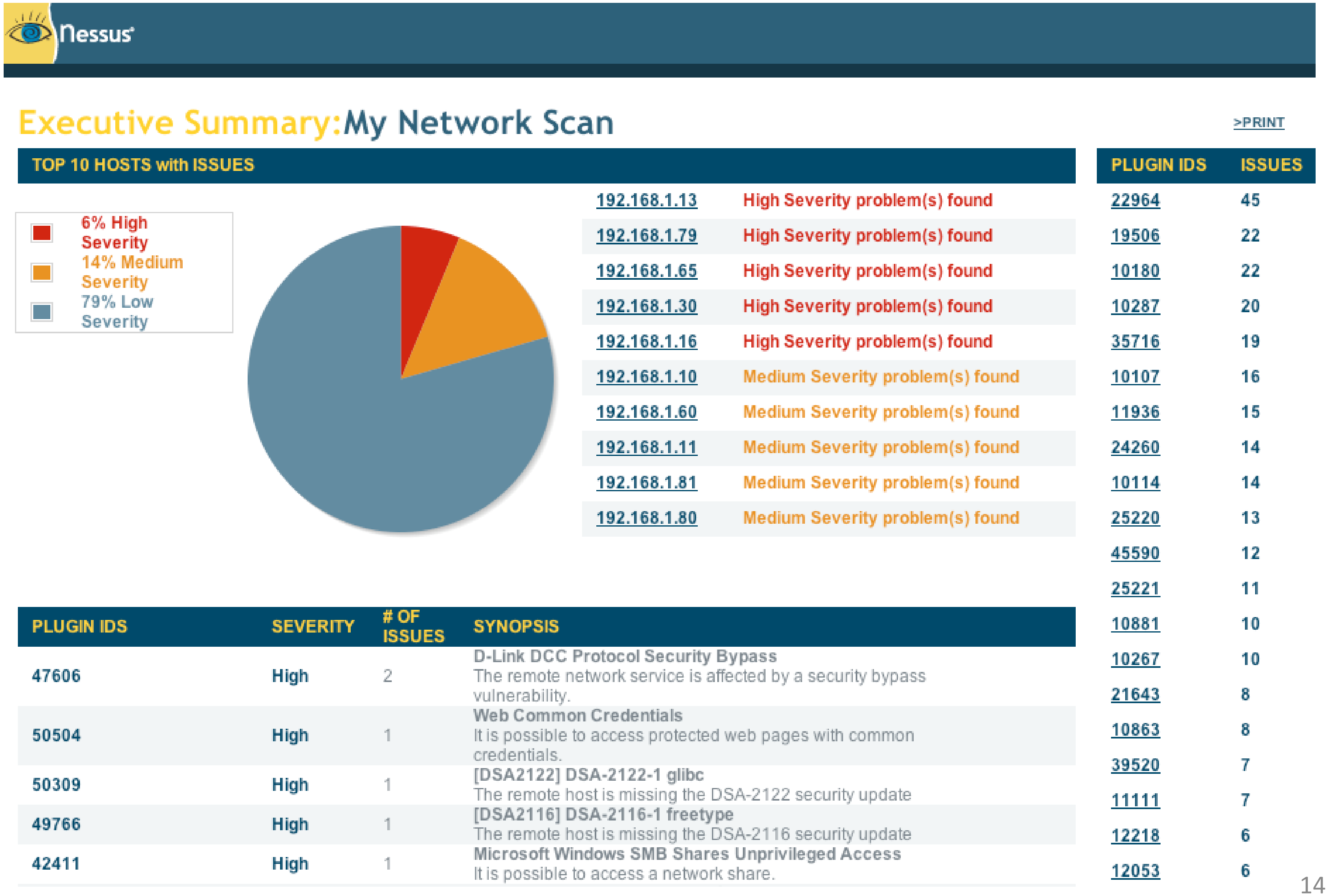
- Infrastructure or platform, e.g.,
  - Web server (OTG-INFO-002)
  - Applications on the webserver (OTG-INFO-004)
  - Web application framework (OTG-INFO-008)
  - Network/infrastructure configuration (OTG-CONFIG-001)

- Vulnerability scanners

nessus®

OpenVAS
Open Vulnerability Assessment System

Nessus report

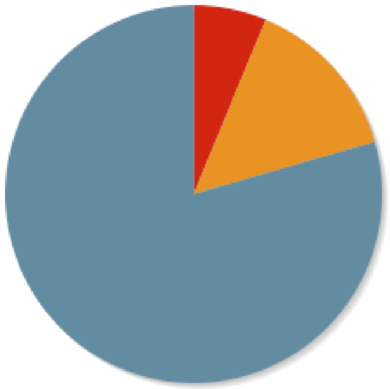


Nessus

Executive Summary:My Network Scan

>PRINT

TOP 10 HOSTS with ISSUES



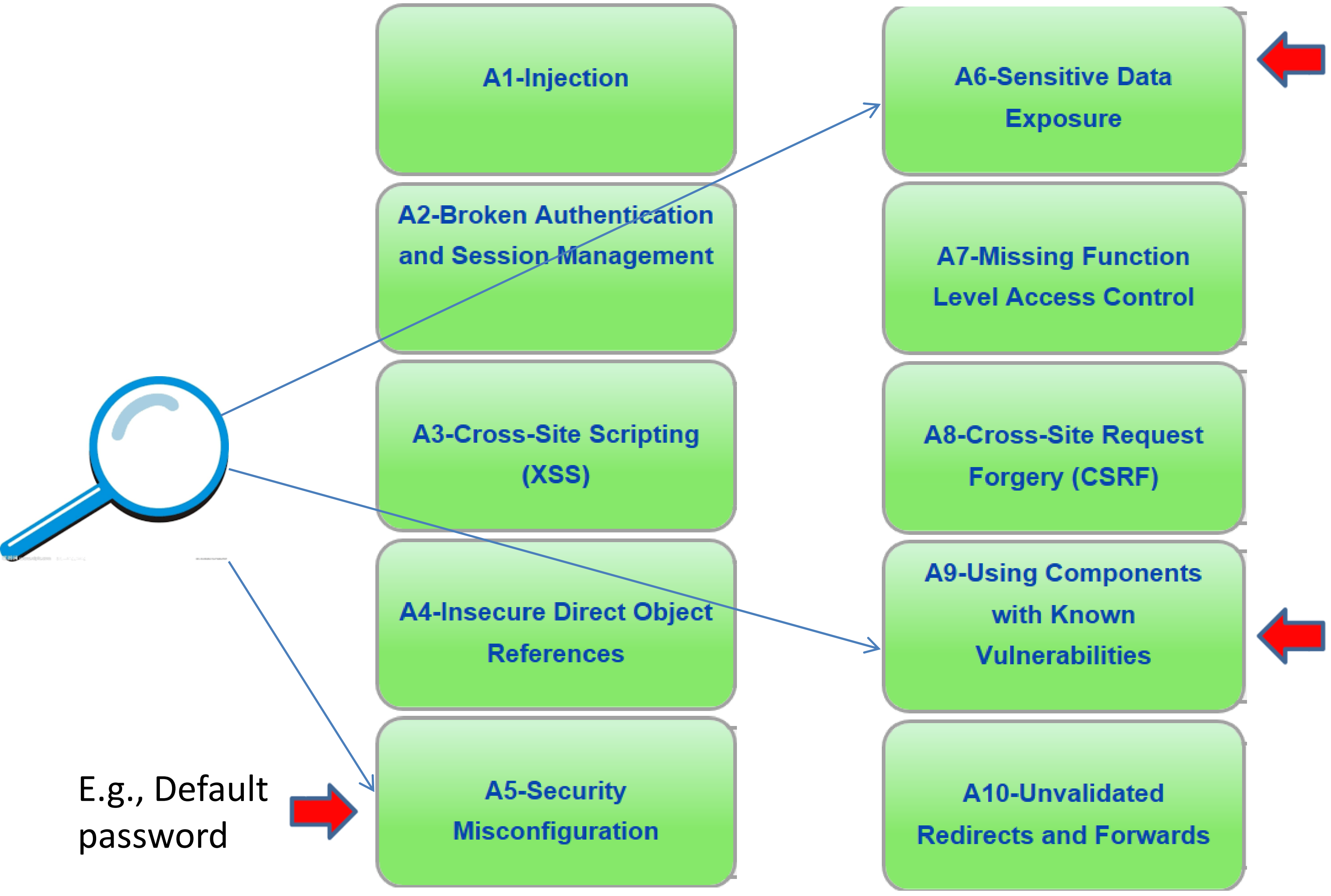
🟥	6% High Severity
🟧	14% Medium Severity
▨	79% Low Severity

192.168.1.13	High Severity problem(s) found
192.168.1.79	High Severity problem(s) found
192.168.1.65	High Severity problem(s) found
192.168.1.30	High Severity problem(s) found
192.168.1.16	High Severity problem(s) found
192.168.1.10	Medium Severity problem(s) found
192.168.1.60	Medium Severity problem(s) found
192.168.1.11	Medium Severity problem(s) found
192.168.1.81	Medium Severity problem(s) found
192.168.1.80	Medium Severity problem(s) found

PLUGIN IDS	ISSUES
22964	45
19506	22
10180	22
10287	20
35716	19
10107	16
11936	15
24260	14
10114	14
25220	13
45590	12
25221	11
10881	10
10267	10
21643	8
10863	8
39520	7
11111	7
12218	6
12053	6

PLUGIN IDS	SEVERITY	# OF ISSUES	SYNOPSIS
47606	High	2	**D-Link DCC Protocol Security Bypass** The remote network service is affected by a security bypass vulnerability.
50504	High	1	**Web Common Credentials** It is possible to access protected web pages with common credentials.
50309	High	1	**[DSA2122] DSA-2122-1 glibc** The remote host is missing the DSA-2122 security update
49766	High	1	**[DSA2116] DSA-2116-1 freetype** The remote host is missing the DSA-2116 security update
42411	High	1	**Microsoft Windows SMB Shares Unprivileged Access** It is possible to access a network share.

Identify vulnerabilities by info. gathering


A1-Injection
A2-Broken Authentication and Session Management
A3-Cross-Site Scripting (XSS)
A4-Insecure Direct Object References
A5-Security Misconfiguration
A6-Sensitive Data Exposure
A7-Missing Function Level Access Control
A8-Cross-Site Request Forgery (CSRF)
A9-Using Components with Known Vulnerabilities
A10-Unvalidated Redirects and Forwards

E.g., Comments in code or error message

E.g., Using Nessus Vulnerability Scanner

E.g., Default password

Injection Attacks

<< All input is evil. >> Michael Howard

2013 OWASP top 10 list

- A1-Injection
- A2-Broken Authentication and Session Management
- A3-Cross-Site Scripting (XSS)
- A4-Insecure Direct Object References
- A5-Security Misconfiguration
- A6-Sensitive Data Exposure
- A7-Missing Function Level Access Control
- A8-Cross-Site Request Forgery (CSRF)
- A9-Using Components with Known Vulnerabilities
- A10-Unvalidated Redirects and Forwards

Injection attacks

- SQL injection
- Blind SQL injection
- Xpath injection
- …

Injection attack

- Malicious inputs inserted into
  - Query/Data
  - Command
- Attack string alters intended semantics
  - Query/Data
  - Command

SQL injection – normal input

Username:
Password:
Log In

“Server side login code (E.g., PHP)”

$ result = mysql_query (“ select * from Users where (name = ‘$ user’ and password = ‘$pass’); ”);

Application constructs SQL query from parameter to DB, e.g.,

Select * from
Users where name = user1 and password = TDT4237

SQL injection – Attack scenario (1)

- Attacker types in this in the ***username*** field

  user1 ’ OR 1=1); --

- At the serverside, the code to be executed

  $ result = mysql_query (“ select * from Users where (name = ‘user1 ’ OR 1=1); -- and password = ‘whocares’); ”);

- SQL query constructed is

Select * from Users

Where name = user1 OR 1= 1

1=1 is always true. All user data compromised

SQL injection – Attack scenario (2)

- If attacker types this in the ***username*** field

  user1 ’ OR 1=1); Drop TABLE Users; --

- SQL query constructed is

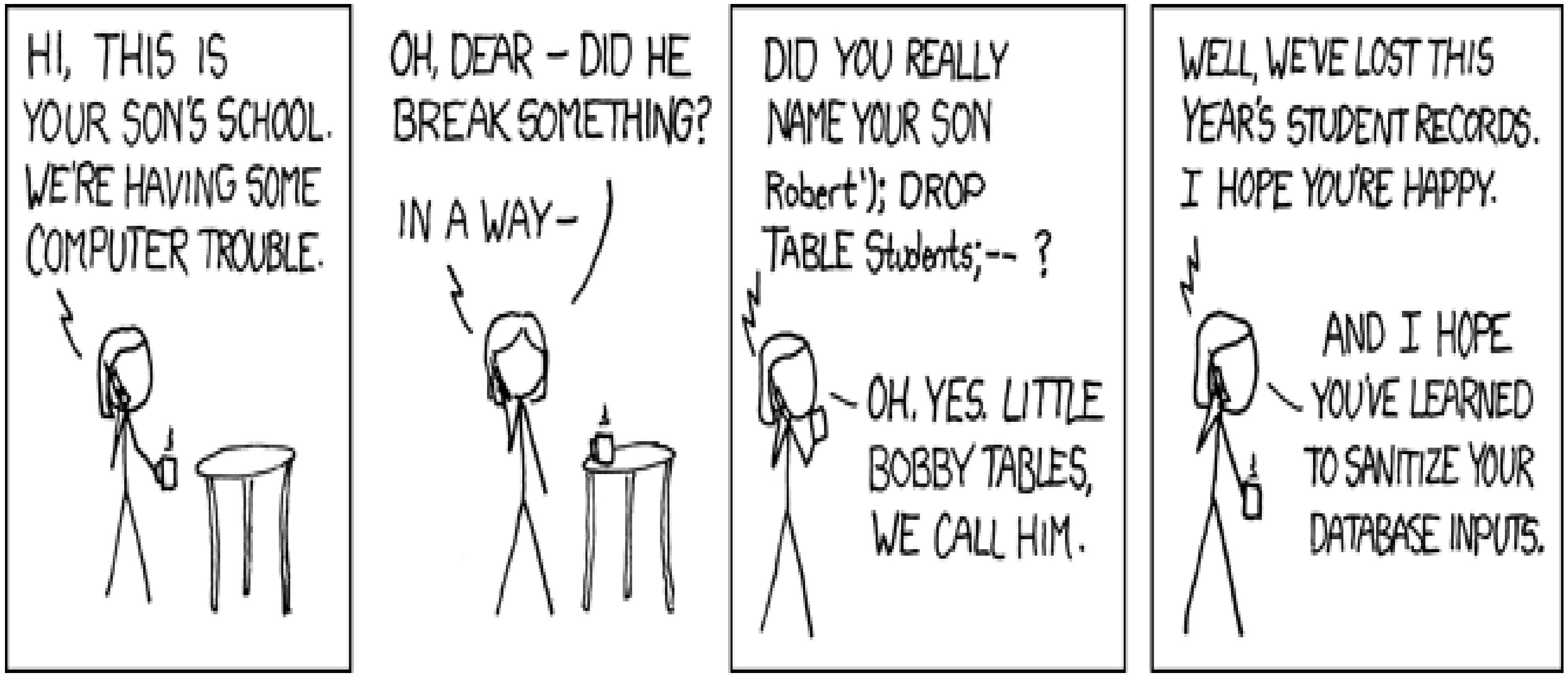
  Select * from Users

  Where name = user1 OR 1= 1;

  Drop TABLE Users;

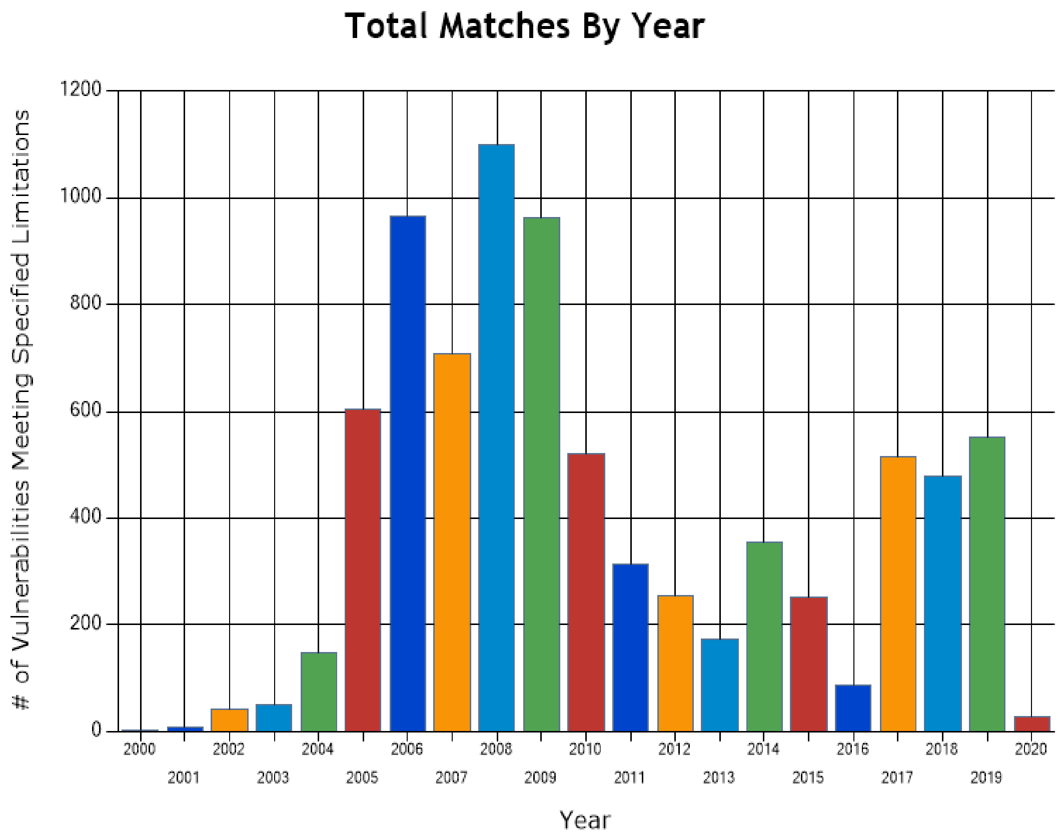
Delete the Table Users

SQL injection humor


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Is SQL injection just a humor?

- SQL injection attack towards CardSystems (a credit card payment processing company) in June 2005
- 263,000 credit card #s (unencrypted) stolen from its DB



By searching key word *SQL injection* in https://nvd.nist.gov/vuln/search/statistics?form_type=Basic&results_type=statistics&query=sql+injection&search_type=all

Why so common?

What can you achieve?

- Bypass authentication
- Privilege escalation
- Stealing information
- Destruction

Blind SQL injection

- Systematically reverse engineering DB schema
- First, insert legitimate info. (e.g., a userID) in DB
- Then check the site is vulnerable to SQL injection?
  - First register as legal user using "attackerUserID"
  - Then, run SQL inject attack and see results
    - SELECT Id FROM Users WHERE userID= attackerUserID AND 1=1; --

      Id shows, vulnerable to SQL injection

      TRUE
  - Manipulate condition after AND to guess something
    - If the guess is correct, Id will show
    - If the guess is wrong, Id will not show

Blind SQL injection (cont')

- Guess DB schema through a binary search

Q: What is the first letter of a Table in DB?

SELECT Id from Users WHERE userID= attackerUserID AND ascii(low (substring ((SELECT Top 1 name FROM sysobjects WHERE xtype = 'U'), 1, 1))) > 109

True or false?

- First letter after m (ascii of m is 109), "*Id*" will show
- First letter before m, "*Id*" will not show

Xpath injection

User/password/account DB in XML (users.xml)

```
<?xml version=”1.0” encoding=”ISO-8859-1”?>
  <users>
    <user>
      <username>gandalf</username>
      <password>Abcd3</password>
      <account>admin</account>
    </user>
    <user>
      <username>Stefan0</username>
     <password>w1s3c</password>
     <account>guest</account>
    </user>
  </users>
```

Xpath injection (cont’)

- Normal query

  username= ‘gandalf’ and password = ‘Abcd3’

- Normal Xpath query

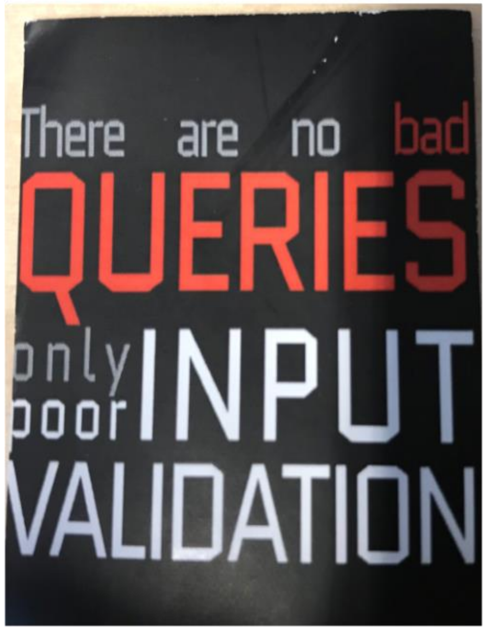
  string(//user[username/text()=’gandalf’ and password/text()=’Abcd3’]/account/text())

- Attack query

  string(//user[username/text()=’’ or ‘1’ = ‘1’ and password/text()=’’ or ‘1’ = ‘1’]/account/text())

SQL injection countermeasures

- Blacklisting
- Whitelisting
- Escaping
- Prepared statement & bind variables
- Mitigating impact


There are no bad
QUERIES
only poor INPUT
VALIDATION

Blacklisting

- Filter quotes, semicolons, whitespace, and …?
  - E.g. Kill_quotes (Java) removes single quotes

```
String kill_quotes(String str) {
  StringBuffer result = new    StringBuffer(str.length());
  for (int i = 0; i < str.length(); i++) {
    if (str.charAt(i) != '\'')
      result.append(str.charAt(i));
  }
  return result.toString();
}
```

user1 ’ OR 1=1); --

Pitfalls of Blacklisting

- Could always miss a dangerous character
- May conflict with functional requirements
  - E.g., A user with name O’Brien

Whitelisting

- Only allow well-defined safe inputs
- Using RegExp (regular expressions) match string
  - E.g., *month* parameter: non-negative integer
    - RegExp: ^[0-9]+$
    - ^ beginning of string, $ end of string
    - [0-9] + matches a digit, + specifies 1 or more
- Pitfalls: Hard to define RegExp for all safe values

Escaping

- Could escape quotes instead of blacklisting
  - E.g. Escape(O’Brien) = O’’Brien

INSERT INTO USERS(username, passwd) VALUES (‘O’’Brien’, ‘mypasswd’)

- Pitfalls: like blacklisting, could always miss a dangerous character

Prepared statements & Bind variables

- Root cause of SQL injection attack
  - Data interpreted as control, e.g., user1 ’ OR 1=1); --,
- Idea: decouple query statement and data input

Examples of Java prepared statement*

```
PreparedStatement stmt=con.prepareStatement("update emp set name=? where id=?");

stmt.setString(1,"Sonoo"); //1 specifies the first parameter in
    the query i.e., name

stmt.setInt(2,101);

int i=stmt.executeUpdate();

System.out.println(i+" records updated");
```

Bind variable;
Data Placeholder

*https://www.javatpoint.com/PreparedStatement-interface

Examples of PHP prepared statement

- Prepare the statement with placeholders
  - $ ps = $ db->prepare('SELECT * FROM Users WHERE name = ? and password = ?');

Bind variable;

Data Placeholder

- Specify data to be filled in for the placeholders
  - $ ps -> execute (array($current_username,
    $current_passwd));

- No explicit typing of parameters like java

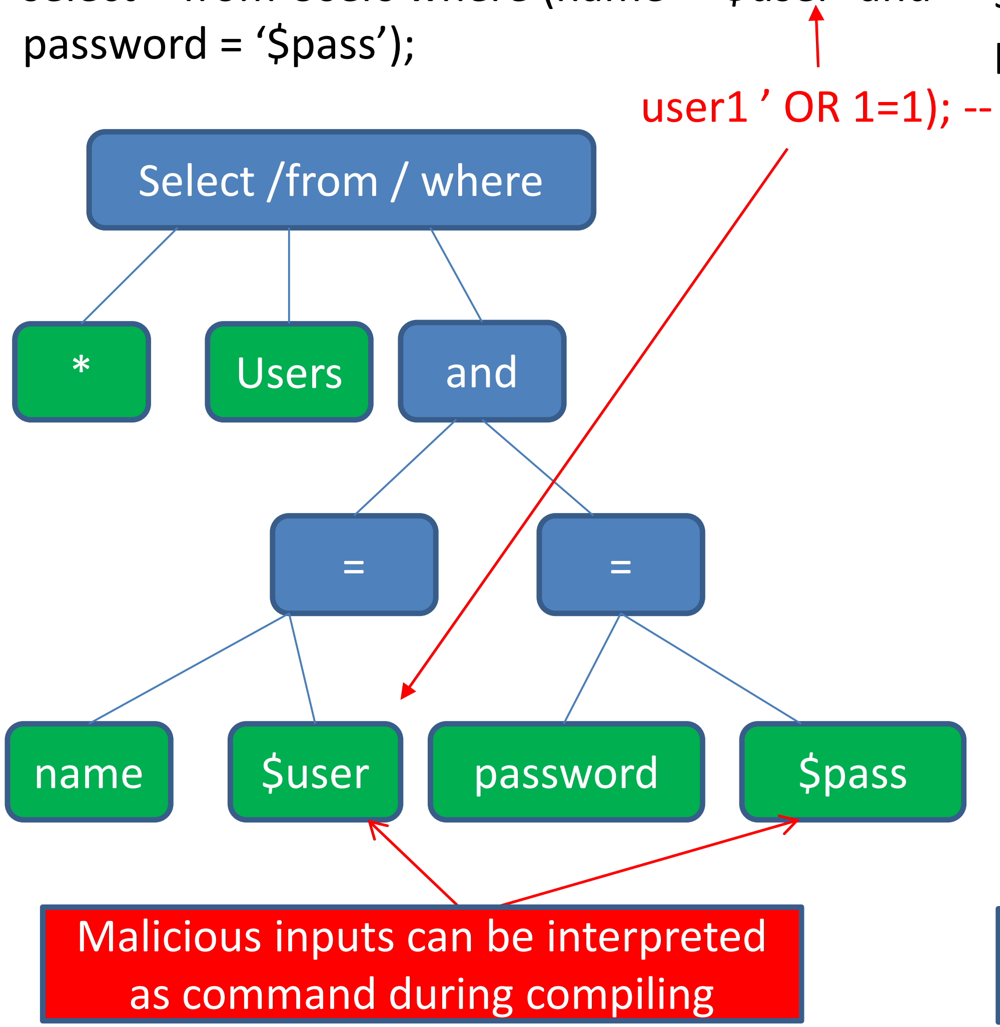
Why prepared statements & bind variables work?

- Decoupling lets us compile the prepared statement before binding the “query input data” !!!
  - Prepared statements
    - Preserve the structure of the intended query
    - “Query input data” is not involved in query parsing or compiling
  - Bind variables
    - ? Placeholders guaranteed to be data (not control)

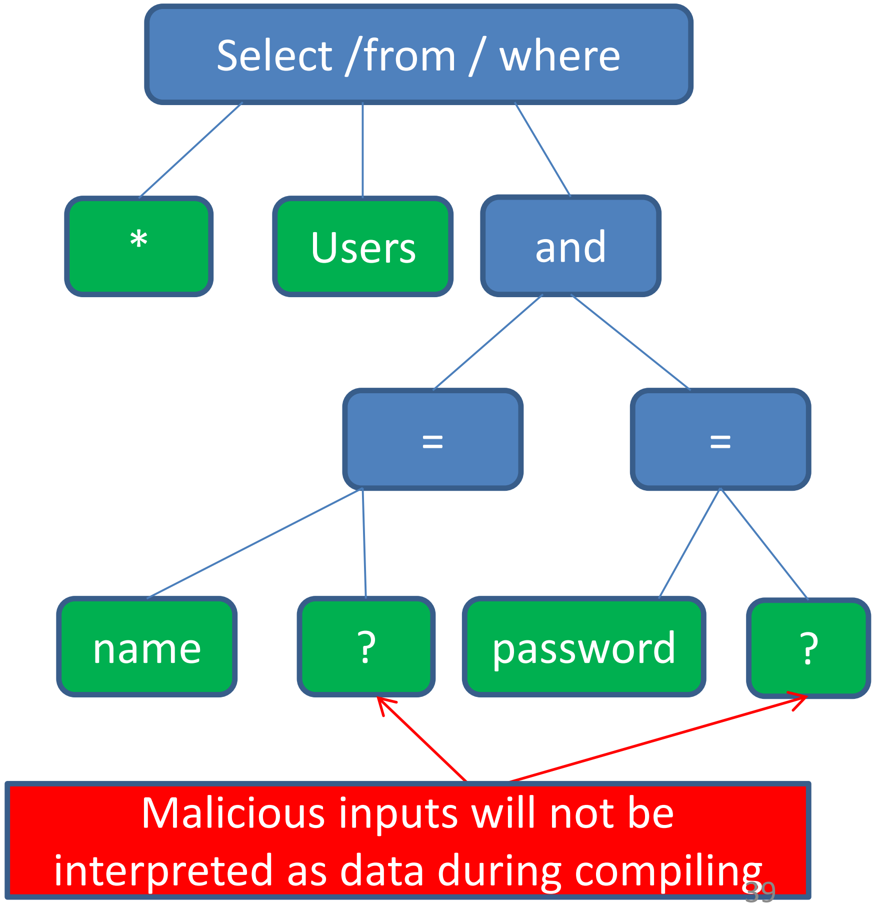
Why Prepared statements & Bind variables work (cont')?

select * from Users where (name = '$user' and password = '$pass');

select * from Users where (name = '?' and password = '?');



Malicious inputs can be interpreted as command during compiling



Malicious inputs will not be interpreted as data during compiling

Mitigating impact

- Prevent schema & information leakage
  - E.g., Not display a detailed error message to external users
  - E.g., Not display stack traces to external users
- Limiting privileges
  - No more privileges than typical user needs
  - E.g., Read access, tables/views the user can query
  - E.g., No drop table privilege for a typical user

Mitigate impact (cont')

- Encrypt sensitive data, e.g.,
  - Username, password, credit card number

- Key management precautions
  - Do not store the encryption key in DB

Question: Which principles have been applied to injection countermeasures?

- Secure the weakest link
- Practice defense in depth
- Fail securely
- Compartmentalize
- Be reluctant to trust
- Follow the principle of least privilege
- Keep it simple
- Promote privacy
- Remember that hiding secrets is hard
- Use your community resources

OWASP SQL injection test cases

- Testing for SQL Injection (OTG-INPVAL-005)
  - Oracle Testing
  - MySQL Testing
  - SQL Server Testing
  - Testing PostgreSQL
  - MS Access Testing
  - Testing for NoSQL injection

OWASP other injection test cases

- Testing for LDAP Injection (OTG-INPVAL-006)
- Testing for ORM Injection (OTG-INPVAL-007)
- Testing for XML Injection (OTG-INPVAL-008)
- Testing for SSI Injection (OTG-INPVAL-009)
- Testing for XPath Injection (OTG-INPVAL-010)
- IMAP/SMTP Injection (OTG-INPVAL-011)
- Testing for Code Injection (OTG-INPVAL-012)

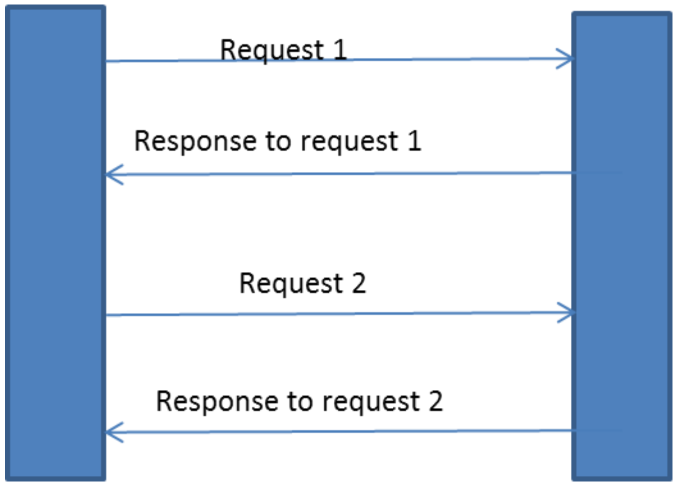
Session Management Attacks

2013 OWASP top 10 list

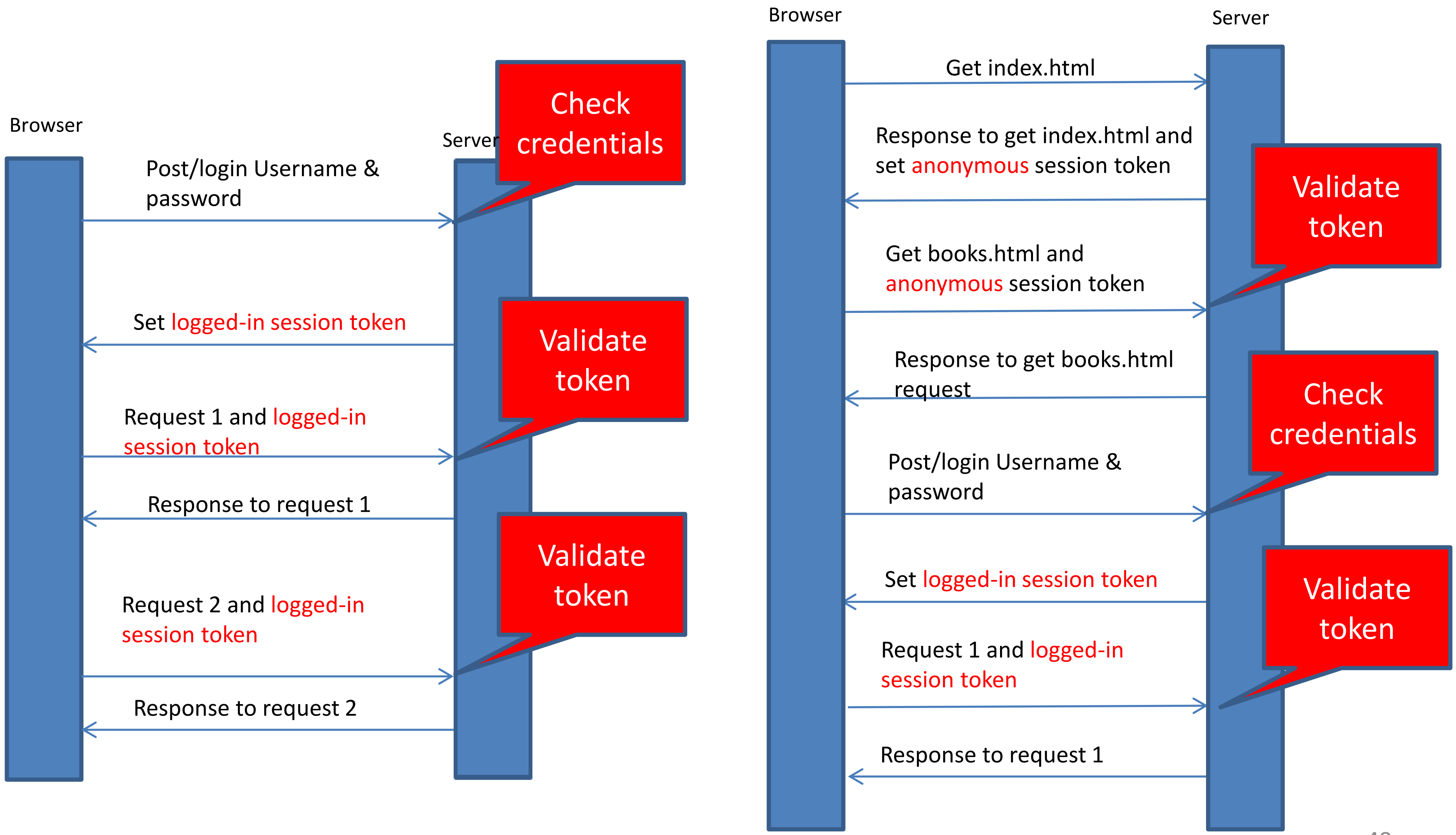
- A1-Injection
- A2-Broken Authentication and Session Management
- A3-Cross-Site Scripting (XSS)
- A4-Insecure Direct Object References
- A5-Security Misconfiguration
- A6-Sensitive Data Exposure
- A7-Missing Function Level Access Control
- A8-Cross-Site Request Forgery (CSRF)
- A9-Using Components with Known Vulnerabilities
- A10-Unvalidated Redirects and Forwards

Why session management?

- HTTP is stateless
- Impossible to know if Req1 and Req2 are from the same client
- Users would have to constantly re-authenticate
- Session management
  - Authenticate user once
  - All subsequent requests are tied to the user



Session tokens


Browser
Server
Check credentials
Post/login Username & password
Set logged-in session token
Validate token
Request 1 and logged-in session token
Response to request 1
Validate token
Request 2 and logged-in session token
Response to request 2
Browser
Server
Get index.html
Response to get index.html and set anonymous session token
Validate token
Get books.html and anonymous session token
Response to get books.html request
Check credentials
Post/login Username & password
Set logged-in session token
Validate token
Request 1 and logged-in session token
Response to request 1

Where to store session token

- Embed in all URL links

  https://site.com/checkout? sessionToken= 1234

- In hidden form field

  <input type= “hidden” name = “sessionToken” value = “1234”>

- Browser cookie

  setcookie: sessionToken = 1234

None is perfect. A combination of all of the above increases security

Issues of embedding token in URL links

- The HTTP Referer header
  - Get /wiki/ntnu HTTP/1.1
  - Host: en.wikipedia.org
  - Keep alive: 300
  - Connection: keep-alive
  - Referer: https://www.google.no/search?dcr=0&ei=m8VbWoDuIor36ATWtLa4CQ&q=ntnu+wiki&oq=ntnu+wiki …

  Referer leaks URL session token to 3rd parties

- Users may publish URL (with token info.) in blogs

Issues of embedding token in hidden form field

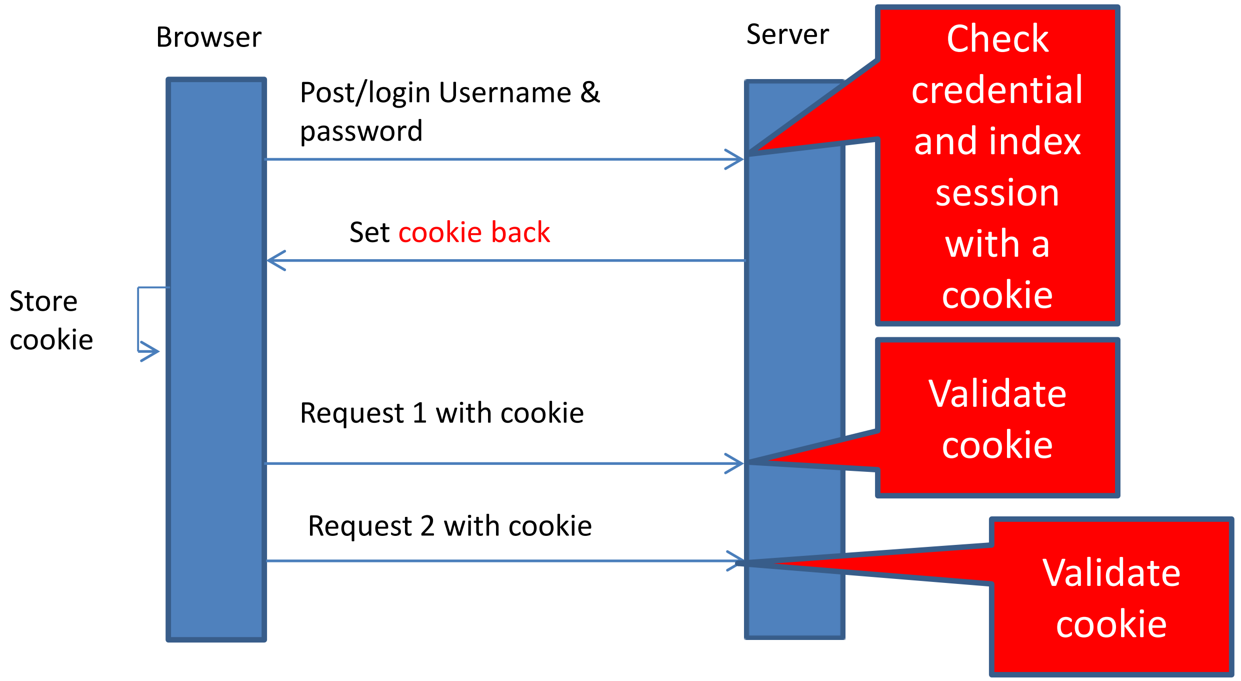
- Do not work for long-lived sessions
- Every protected web page must embed this hidden token

Issues of embedding token in cookies

- The browser sends cookies with every request, even when it should not (e.g., CSRF)

- Explained in detail in the following slides

Session management with cookie



How cookie works

- Setting and sending cookies
  - In header of HTTP response (Server to browser)

    set-Cookie: token=**1234**; expire=Wed, 3-Aug-2016 08:00:00; path=/; domain = idi.ntnu.no
  - In header of HTTP request (Browser to server, when visit the domain of the same scope)

    Cookie: token=**1234**

- Cookie protocol problem
  - Sever only sees Cookie: NAME = VALUE
  - Server does not see which domain sends the cookie

Session management attacks and countermeasures

- Session token theft
- Session token predication attack
- Session fixation attack

Session token theft – Sniff network

- User (e.g., Alice)
  - Alice logs in login.site.com (HTTPS)
  - Alice gets logged-in session token
  - Alice visits non-encrypted.site.com (HTTP)
- Attacker
  - Wait for Alice to log in
  - Steal the logged-in session token (in HTTP)

    E.g., FireSheep (2010) sniff WiFi in wireless cafe
  - Impersonate Alice to issue request

Session token theft – Logout problem

- What should happen during logout
  - 1. Delete session token from the client
  - 2. Mark session token as expired on the server
  - Many web sites do (1) but not (2)!!
- Attacker
  - If can impersonate once, can impersonate for a long time
  - E.g., Twitter sad story
    - Token does not become invalid when the user logs out

https://packetstormsecurity.com/files/119773/twitter-cookie.txt (2013)

Solutions to Session token theft

- Always send Session ID over an encrypted channel
- Remember to log out
- Time out session ID
- Delete expired session ID
- Binding session token to the client’s IP or computer

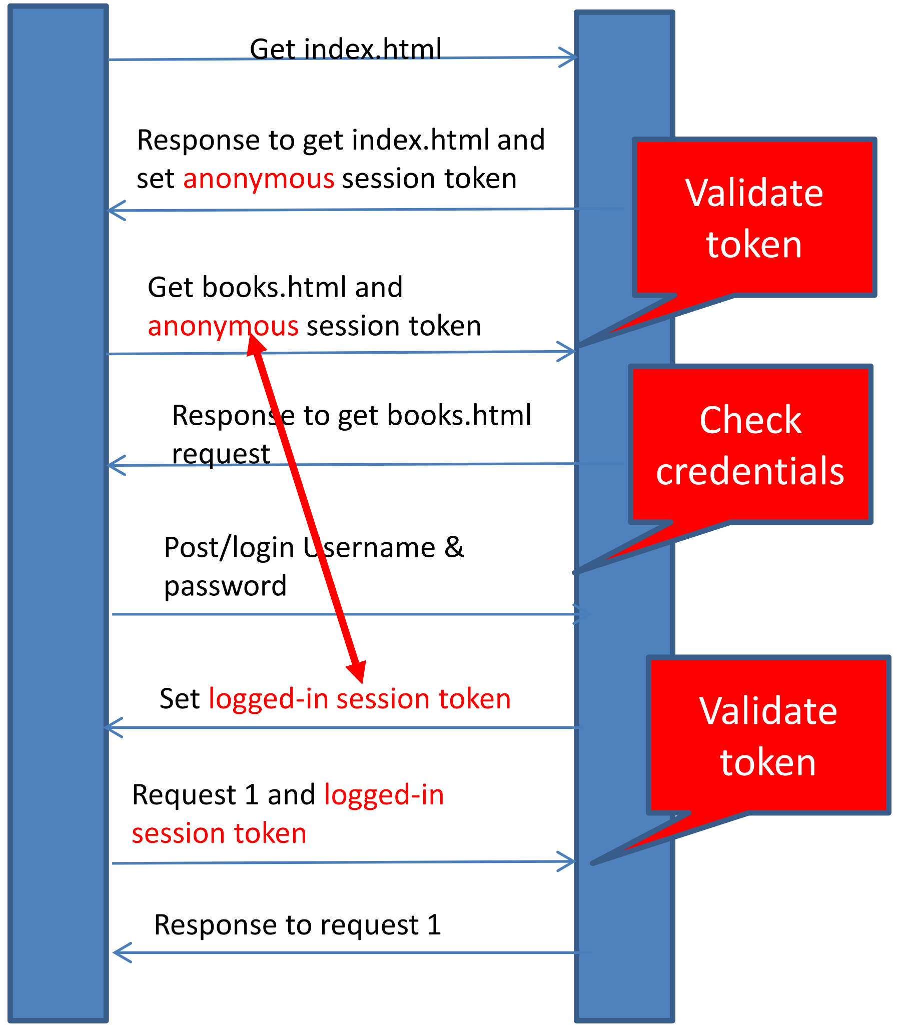
Binding session token to client's IP or Computer

- Idea:
  - Overcome cookie protocol problem
    - Sever only sees Cookie: NAME = VALUE
    - The server does not see which domain sends the cookie
- Combine IP
  - Possible issue: IP address changes (Wifi / 3G)
- Combine user agent: weak defense, but does not hurt

Session token predication attack

- Predicable tokens, e.g., counter
- Non-predicable token means
  - Seeing one or more token
  - Should not be able to predict other tokens
- Solution:
  - Do not invent own token generator algorithm
  - Use token generator from the known framework (e.g. ASP, Tomcat, Rails)

Session fixation attack


Get index.html
Response to get index.html and set anonymous session token
Validate token
Get books.html and anonymous session token
Response to get books.html request
Check credentials
Post/login Username & password
Set logged-in session token
Validate token
Request 1 and logged-in session token
Response to request 1

- User (e.g., Alice):
  - Visits site using an anonymous token
- Attacker
  - Overwrites user's anonymous token with own token
- User:
  - Logs in and gets anonymous token elevated to logged-in token
- Attacker:
  - Attacker's token gets elevated to logged-in token after user logs in
- Vulnerability: Sever elevates the anonymous token without changing the value

How to overwrite session token?

- Tampering through network
  - Alice visits non-encrypted.site.com (HTTP)
  - The attacker injects into the response to overwrite the secure cookie

    ```
    Set-cookie: SSID=maliciousToken;
    ```
- Cross-site scripting
  - How? Explain in lecture next week

Mitigate session fixation

- Always issue a new session token, when elevating from anonymous token to logged in token

Session management tests

- Testing for Bypassing Session Management Schema (OTG-SESS-001)
- Testing for Cookies attributes (OTG-SESS-002)
- Testing for Session Fixation (OTG-SESS-003)
- Testing for Exposed Session Variables (OTG-SESS-004)
- Testing for logout functionality (OTG-SESS-006)
- Test Session Timeout (OTG-SESS-007)
- Testing for Session puzzling (OTG-SESS-008)

Summary

- Information gathering
- Injection attacks and solutions
- Session management attacks and solutions
- Next lecture
  - Cross-site attacks
  - OWASP 2017 attacks
  - HTML 5 security issues
  - Authentication and password security

To read before next lecture

- OWASP Testing guide
  - Authentication testing
  - CSRF testing
  - CSS injection testing
- Security engineering book
  - Chapter 2, content related to password
- Foundations of security book
  - Chapter 9 and 10