OWASP TOP 10

ILIA ALSHANETSKY
@ILIIA

HTTPS://JOIND.IN/15741
ME, MYSELF & I

- PHP Core Developer
- Author of Guide to PHP Security
- Security Aficionado
THE CONUNDRUM

USABILITY ⇄ SECURITY

YOU CAN HAVE ONE ;-)
OPEN WEB APPLICATION SECURITY PROJECT

• A set of best practices and recommendations around making web applications more secure

• General database of common vulnerability vectors

• A good place to keep yourself up-to-date on security

Not a Bible™
THE TOP 10
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.
WHAT NOT TO DO

// $_POST['login'] = "login";
$pdo->query("SELECT * from users WHERE login={$_POST['login']} AND password={$_POST['pwd']}");  

// $_POST['login'] = "' OR 1=1; --";
$pdo->query("SELECT * from users WHERE login='{$_POST['login']}' AND password='{$_POST['pwd']}'");  

// $_POST['login'] = chr(0xbf) . chr(0x27) . " OR 1=1; --";
// 0xbf27 + addslashes() == 0xbf5c27 == ê½œ + "'
$pdo->query("SELECT * from users WHERE login='' . addslashes($_POST['login']) . ''
    AND password='' . addslashes($_POST['pwd']) . ''");  

$pdo->query("SELECT * from users WHERE login='' . $pdo->quote($_POST['login']) . ''
    AND password='' . $pdo->quote($_POST['pwd']) . ''");

http://hakipedia.com/index.php/SQL_Injection
PREVENT INJECTION

- For databases use prepared statements
- White list inputs whenever possible
- Sanitize inputs (use filter extension)
- Don’t trust and always verify!
BROKEN AUTHENTICATION & SESSION MANAGEMENT
MITIGATION

• Enforce strong password policy
• Require periodic reset of password
• Use 2 factor authentication
• Use SSL and secure flag on cookies
• Don’t forget about auto-logout
• Don’t neglect failed-login detection & tracking
SESSION SECURITY

• Don’t trust new session ids
  session_regenerate_id(true)
  session.use_strict_mode (5.5.2+)

• Use unique session names (not PHPSESSID)

• Only use httpOnly cookies

• Ensure true randomness for session ids
CROSS SITE SCRIPTING -XSS

Attacker posts content that contains embedded JavaScript that is stored by vulnerable application.

User visits the compromised site that transmits expected content along with injected script.

User's browser renders the page and executes injected script.

User's data is transmitted to the attacking party.
PROTECT YOURSELF

- Use filter extension to filter inputs
- Ensure that outputs are HTML encoded
- Don’t reinvent the wheel
- Don’t consider any part of the request as being “safe”
INSECURE DIRECT OBJECT REFERENCES
PREVENTION

• Low level access controls

• Prevent user input in file/URL access commands

• No unsanitized input to execution commands (escapeshellarg() for arguments)

• Non-white-list input shouldn’t dictate logic
MORE SPECIFICALLY

- Usage of default, un-secure settings
- Not disabling initial accounts (especially those with admin rights)
- Failure to apply latest security patches
- Leaving un-used functions/modules enabled
- Exposed error handling
- Keeping “upgrade” scripts in accessible directories
PREVENTION > CURE

- Perform periodic security checks using automated tools
  - STATIC CODE ANALYSIS
  - NMAP
  - EXTERNAL VULNERABILITY SCANNERS
    http://sectools.org/tag/web-scanners/
  - DISTRO PACKAGE SECURITY CHECKS
SENSITIVE DATA EXPOSURE

GREAT SUCCESS!
SOME EXAMPLES

- Exposed PHP error messages
- Non-web related files stored inside web-root
- Application version exposure
- Un-encrypted sensitive data storage
- Not using SSL
MISSING FUNCTION LEVEL
ACCESS CONTROL

YOU SHALL

NOT PASS
WTF??

- Valid input processing without access controls
- Reliance on hidden fields for record identifiers
- Decentralized access control layer
- JavaScript driven access controls
CROSS-SITE REQUEST FORGERY (CSRF)

Attacker tricks the user into following a link to a trusted site with vulnerable payload

User visits the compromised site that renders desired content along with compromised payload

User's data is transmitted to the attacking party
PREVENTION

• Don’t perform data changes on GET
• Use secure (csrf) tokens for POST
• Dynamic Field Names
USING COMPONENTS WITH KNOWN VULNERABILITIES

• Using old vulnerable software

• Not keeping libraries up-to-date
  *cough*OpenSSL*cough*

• Forgetting to update JavaScript libraries
THE CURE

• On server do routine checks for software with known exploits

• Keep libraries up-to-date

• Compare utilized software versions to those listed on http://cve.mitre.org/
UNVALIDATED REDIRECTS AND FORWARDS

- Header Injection
- JavaScript Parameter Injection
- Reliance on HTTP_REFERER
THANK YOU FOR LISTENING

HTTP://ILIA.WS
@ILIAA

FEEDBACK @ HTTPS://JOIND.IN/15741