Ashley Hedberg
Comp 116
11/14/2013

Assignment 4: Technical Risk Analysis

Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
1	Code injection	Ability to view contents of directories and files on server by setting the id parameter equal to a system() call	VH	View files that you should not have access to; conduct system-level commands without proper authorization	Remove eval() command from index.php; validate user input before passing the id parameter to eval()	Code injected through id parameter is not executed
2	SQL injection	Any SQL query can be executed by passing it to the id parameter	H	Allows data to be viewed without proper authorization	Remove eval() command from index.php; validate user input before passing the id parameter to eval()	SQL queries passed to the id parameter are not executed
3	SQL injection	Username for admin page can be set to 1 'OR' 1 '=' 1 and used to gain access	H	Allows unauthorized users to gain administrator-level entry into the system	Verify user input before executing dynamically constructed SQL queries; use prepared statements instead of dynamically constructed queries	SQL commands injected through parameter inputs is not executed maliciously

Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
4	Credentials management	Password for MySQL database is hard-coded into index.php	M	Exposes database access credentials to anyone who views the source PHP, allowing them to gain unauthorized access to the data stored there	Store credentials in a configuration or properties file using encryption	Database credentials are not visible in plaintext in source files
5	Credentials management	Password for scoreboard database is hard-coded in ctf_scoreboard.php	M	Exposes database access credentials to anyone who views the source PHP, allowing them to gain unauthorized access to the data stored there and potentially hack the scoreboard	Store credentials in a configuration or properties file using encryption	Database credentials are not visible in plaintext in source files
6	Cross-site scripting	Can tamper with the login cookie (“lg” flag) to gain access to main.php	M	Gives users who have not correctly logged into the site unauthorized access	Encrypt or sign cookies to prevent tampering; store sessions on the server side, and not with cookies on the client side	Use TamperData to verify that the “lg” flag in the cookie cannot be modified to gain access
7	Cross-site scripting	Scripts written in post bodies are executed after the post is created	M	Allows visitors to deface website, which could range from merely annoying (backgrounds of cats and alert messages) to exceedingly malicious (inserting nearly invisible frames that contain malware)	Escape user input before creating their posts	Scripts injected through new posts do not run on page load

Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
8	Information leakage	Error message in case of failed database connection can print excessive information	L	Can give attackers information about potential vulnerabilities that can be used to further exploit the system	Display generic error messages to users instead of the detailed ones returned by mysql.error()	Intentionally fail to connect to database and inspect error messages displayed to user
9	Insecure password storage	Passwords stored in initial_data.sql in database/ are hashed, but not salted	H	Easier to crack passwords to gain unauthorized entry	Use a cryptography library that will salt passwords in addition to hashing them	Ensure that cryptography library is used correctly and that user login still works
10	Buffer overflow	Code for namegame contains dangerous strcpy() function	H	Leaves application vulnerable for buffer overflow exploitations, which (if done correctly) can allow arbitrary code to execute	Use strncpy() instead of strcpy(), which does bounds checking	Dangerous string functions, like strcpy() and gets(), no longer appear in source code
11	Stack smashing	The Makefile for runme.c contains the -fno-stack-pointer flag	H	Removes the safeguard against stack smashing and buffer overflows, allowing buffers to be overflowed and arbitrary code to potentially be executed	Compile programs using appropriate flags (i.e. not the -fno-stack-pointer	Ensure that all Makefiles, compile scripts, and other methods of compiling namegame do not contain the -fno-stack-pointer flag