Shell Command Injection

Command Shell/ Command Interpreter

- "A shell is a piece of software that provides an interface for users of an operating system which provides access to the services of a kernel."
- It allows us to for example start programs, delete files, delete directories etc.

Command Shells

- Unix
  - sh, csh, bash
- Windows
  - cmd

Invoking shell commands from programs

- Dangerous.
- Especially if we are passing user input to the shell command.

Perl

- Early Web Applications often were implemented in a scripting language such as Perl.
- The command inside ` ` is executed in a shell and the output displayed by print

```
$username = $form{“username”}
print `finger ${username}`
```

Remove the root directory

- User enters username
  - abc ; rm –rf /
- The commands executed are
  - finger abc ; rm –rf /
- Recursive delete of the root of the file system.

Blame

- System administrator
  - The system account running the Web Server process should have only had read access to the file system.
- Programmer
  - For letting shell commands be passed to the shell.

Avoiding Shell Command Injection

- Identify how the shell commands are invoked.
- This will vary from language to language.

Perl

- `command`
  - Executes command
- open(program, ”| command”)
  - Pipes the output from program to command
- exec “command”
  - Replaces the currently running program
- system “command”
  - Starts command and continues executing the current process

Metacharacters in shells

- Different shells have different metacharacters (special characters).
- Also the meaning of a character will depend on the context.

Metacharacters in bash

- " $ & ' ( ) * ; < > ? [\ ] `{ | } ~ space tab cr lf

- Context

```
echo '\\'
\\
echo "\\"
\
```

Context in bash

- \ is the escape character in bash.
- Inside a single quotes it is treated as an ordinary character.

```
echo '\\'
\\
echo "\\"
\
```

Use of metacharacters/special characters

- We have already seen the exploitation of the metacharacter ; to enable execution of two commands instead of one.
- What we often need is to turn off the interpretation of these special characters as special, and have them treated simply as text.

Escaping metacharacters

- We can escape all special characters in bash using \.
- The PHP function escapeshellcmd does this, i.e. it takes a piece of string and returns a string with all special characters escaped by prefixing them with \.
- Blacklisting - can be dangerous as special characters can be missed.

bash - ' '

- Single quote encapsulation is very strict.
- Text is just text.
- Nothing is treated as a special character.
- There is no way to escape any character inside single quotes.
- To get the string O'Brien
- 'O'\''Brien'

" "

- Inside a double quoted string all characters except $, `, " and \ loose their special meaning.

Systems Within Systems

- When we use a shell to start a command, two programs start
  - the command interpreter itself
  - the program we want to run
- The program itself might have its own set of special characters.

Metacharacter Problems

- Two options when passing data to subsystems.
- Remove special characters
- Escape special characters
- If the metacharacter makes sense as an ordinary character, then escape it, otherwise remove it.

Metacharacter Problems (cont)

- For example, the quote is a valid character in a String (O'Connor) so it should be escaped.
- A non number has no valid meaning in a file that expects a number so it can be removed.
- If possible separate data from control (Prepared Statement in SQL). Then there are no metacharacters.

Architecture

- Strictly encapsulate all communication with other systems/subsystems.
- For example, object relational mapping.

Security in Depth

- Errors will be made.
- Have more than one line of defence if possible.
- Invest time in assigning appropriate permissions to subsystems.

Security in Depth (cont)

- For example,
  - permissions of the user that your Web server runs as
  - [defence after metacharacters have been handled]
- Defence before passing values to subsystems
  - User input validation.

Summary (cont)

- A Web Application might pass data to a variety of subsystems
  - databases
  - command shells
  - XML documents
  - file systems
  - libraries
  - legacy systems etc.

Summary (cont)

- Many subsystems treat certain characters (metacharacters) in a special way.
- These characters need to be escaped when passed to the subsystem as data.
- If they are not, hackers can use injection attacks to inject control information into data.
- This is generally a problem when control information and data are mixed together.

Summary (cont)

- Some systems provide a way to separate control information and data. (Prepared statements in SQL).
- Should strive for defence in depth.
  - Validate user input
  - Deal with meta-characters
  - Have appropriate permission for subsystems.