Scripting Vulnerabilities

Scripting languages

- Construct commands (scripts) from predefined code fragments and user input at runtime
- Script is then passed to another software component where it is executed.
- It is viewed as a domain-specific language for a particular environment.
- It is referred to as very high-level programming languages
- Example:
  - Bash, PowerShell, Perl, PHP, Python, Tcl, Safe-Tcl, JavaScript

Vulnerabilities

- An attacker can hide additional commands in the user input.
- The system will execute the malicious command without any awareness

Example 1: Command Injection

Consider a server running the following command

- `system`: takes a string as input, spawns shell, and executes the string as command in the shell.

```
void display_file(char* filename) {
  char cmd[512];
  snprintf(cmd, sizeof(cmd), "cat %s", filename);
  system(cmd);
}
```

Normal case:

- A client sets `filename`=hello.txt

```
cat hello.txt
```

Compromised Input:

- The attacker sets `filename` = hello.txt; rm –rf /
- The command becomes:

```
cat hello.txt; rm -rf /
```

- After displaying file, all files the script has permission to delete are deleted!

Defenses against Command Injection

Avoid shell commands

Use more secure APIs

- Python: subprocess.run()
- C: `execve()`

Input inspection

- Sanitization: escape dangerous characters
- Validate and reject malformed input.
- Whitelist: only choose from allowed values

Drop privileges

- Run processes as non-root users.

Example 2: SQL Injection

Structured Query Language (SQL)

- A domain-specific language for managing data in a database

Basic syntax

- Obtain a set of records:

```
SELECT name FROM Accounts
SELECT * FROM Accounts WHERE name= 'Alice'
```

- Add or update data in the table:

```
INSERT INTO Accounts (name, age, password) VALUES ('Charlie', 32, 'efgh')
UPDATE Accounts SET password='hello' WHERE name= 'Alice'
```

- Delete a set of records or the entire table

```
DELETE FROM Accounts WHERE age >= 30
DROP TABLE Accounts
```

- Other syntax characters

  `--` single-line comments

  `;` separate different statements.

Accounts		
name	age	password
Alice	18	1234
Bob	23	5678
Eva	50	abcd

Example 2: SQL Injection

Consider a database that runs the following SQL commands

```
SELECT * FROM Accounts WHERE name= '$name'
```

- Requires the user client to provide the input `$name`

Normal case:

- A user sets `$name`=Bob:

```
SELECT * FROM Accounts WHERE name= 'Bob'
```

Compromised inputs

- The attacker sets `$name = ' OR 1=1 --`

```
SELECT * FROM client WHERE name= '' OR 1=1 --'
```

`1=1` is always true. The entire client database is selected and displayed!

- The attacker sets `$name = '; DROP TABLE Accounts --`

```
SELECT * FROM client WHERE name= ''; DROP TABLE ACCOUNTS --'
```

A new statement is injected, which deletes the entire table!