Injections

Alexander Tushinsky
Cybersecurity & Software Development Consultant

@ltmodcs alextushinsky.com

Overview

- What is an injection flaw?
  - Different types
  - What vulnerability does this cause?
- Examples
- Remediation

Injections

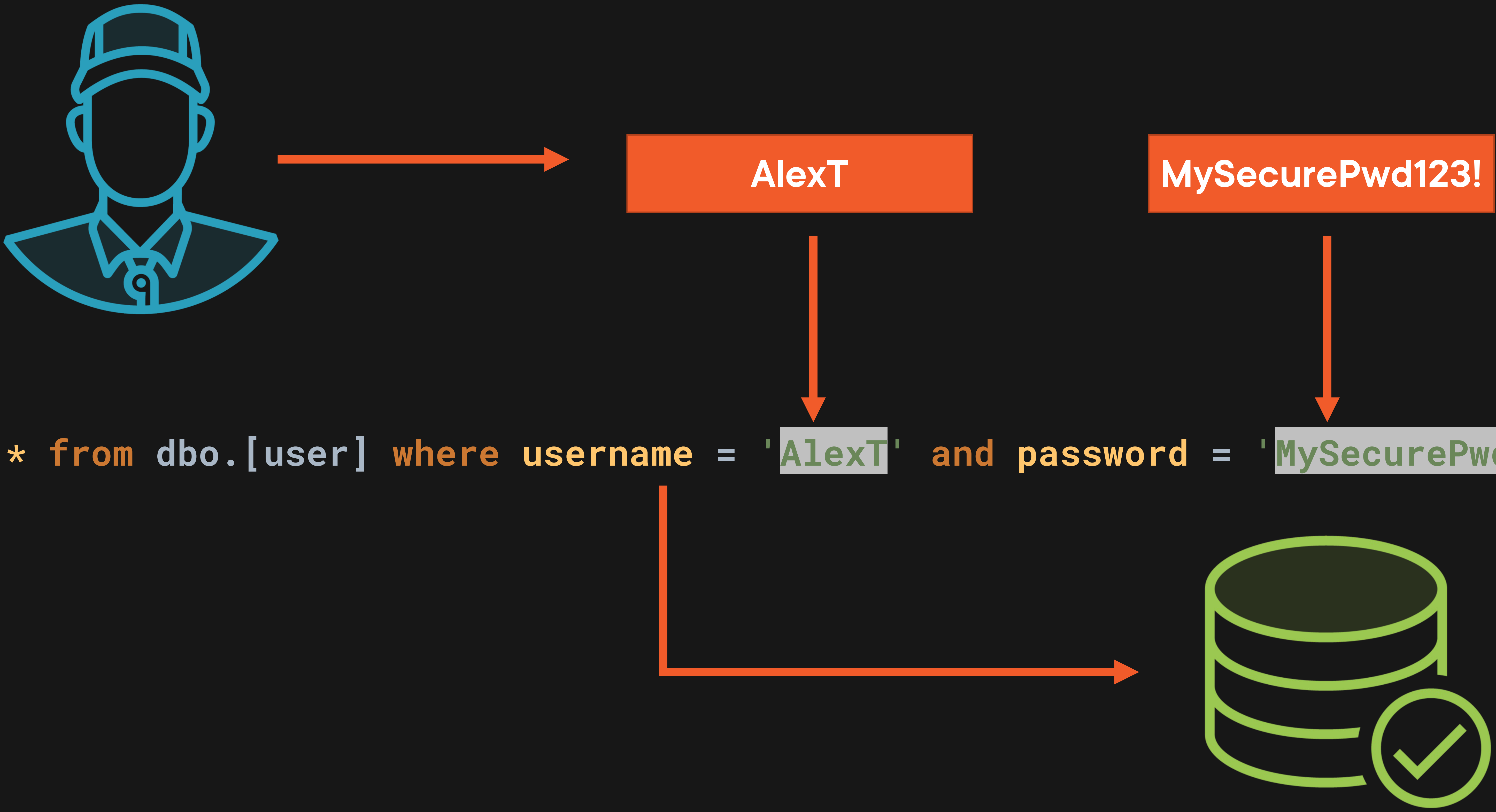
- Occurs anytime code comingles directly with user input
- Frequently seen with SQL queries, but affects LDAP, OS, and other technologies

Injections Example

- I feel {HAPPY}.
  {SAD}
  {ANGRY}

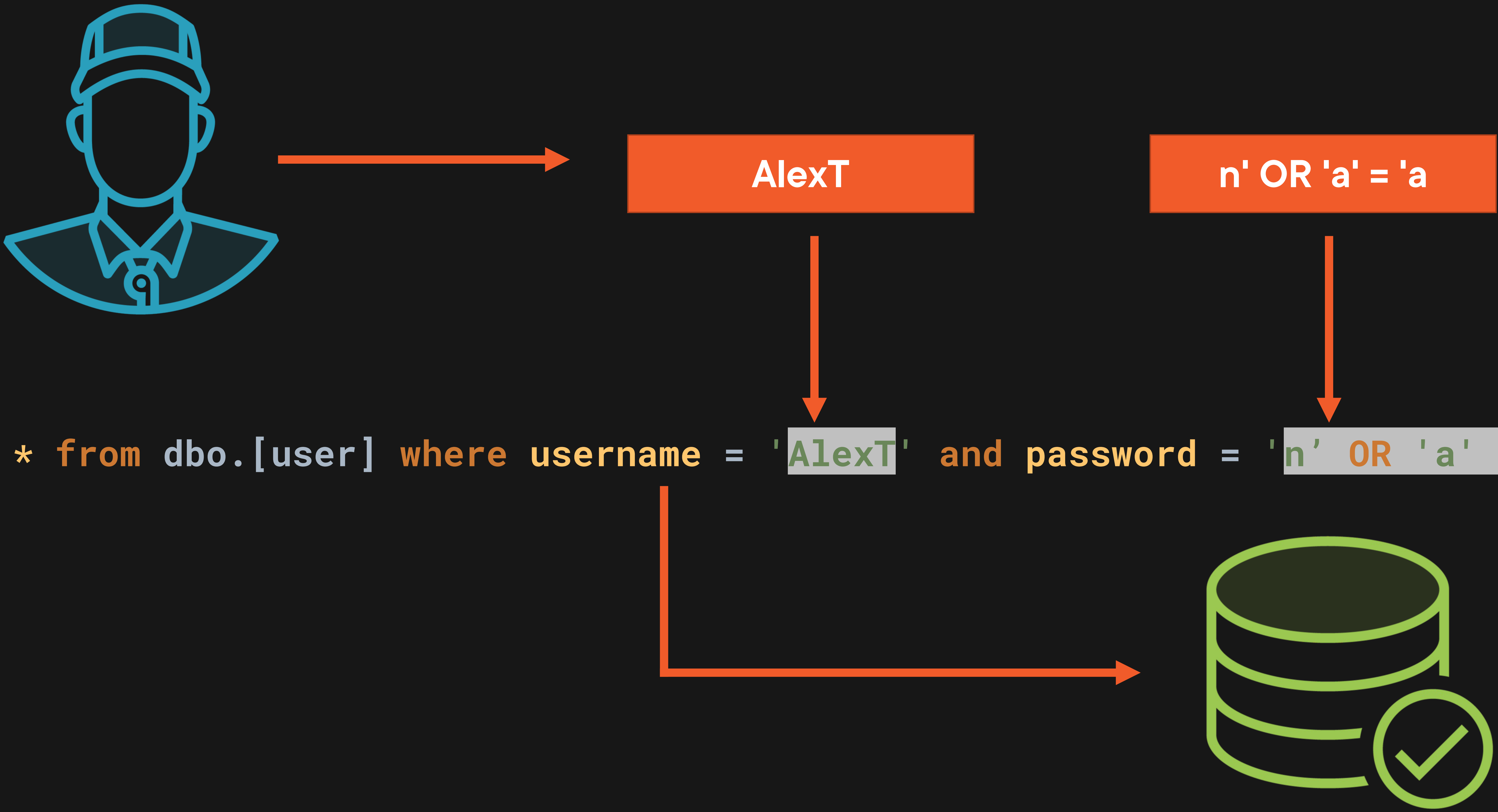
- I feel {that product X that I bought today is worth every penny}.

SQL Injection Example



```
select * from dbo.[user] where username = 'AlexT' and password = 'MySecurePwd123!'
```

SQL Injection Example

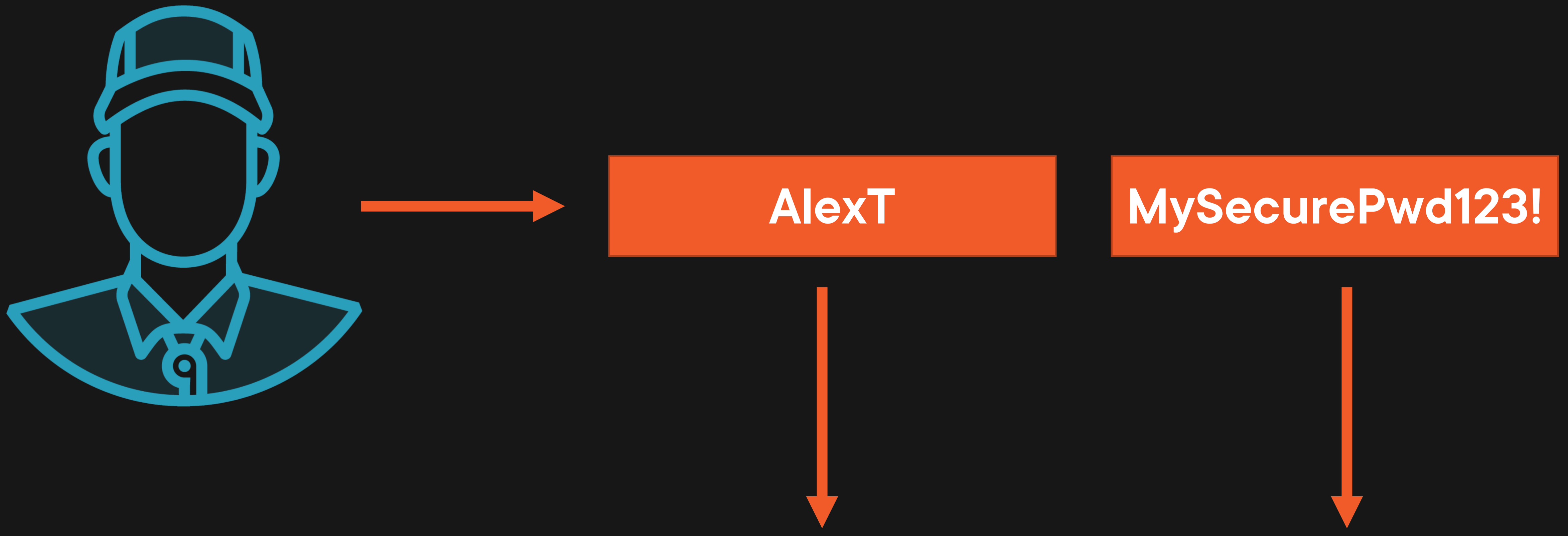


```
select * from dbo.[user] where username = 'AlexT' and password = 'n' OR 'a' = 'a'
```

LDAP Injections

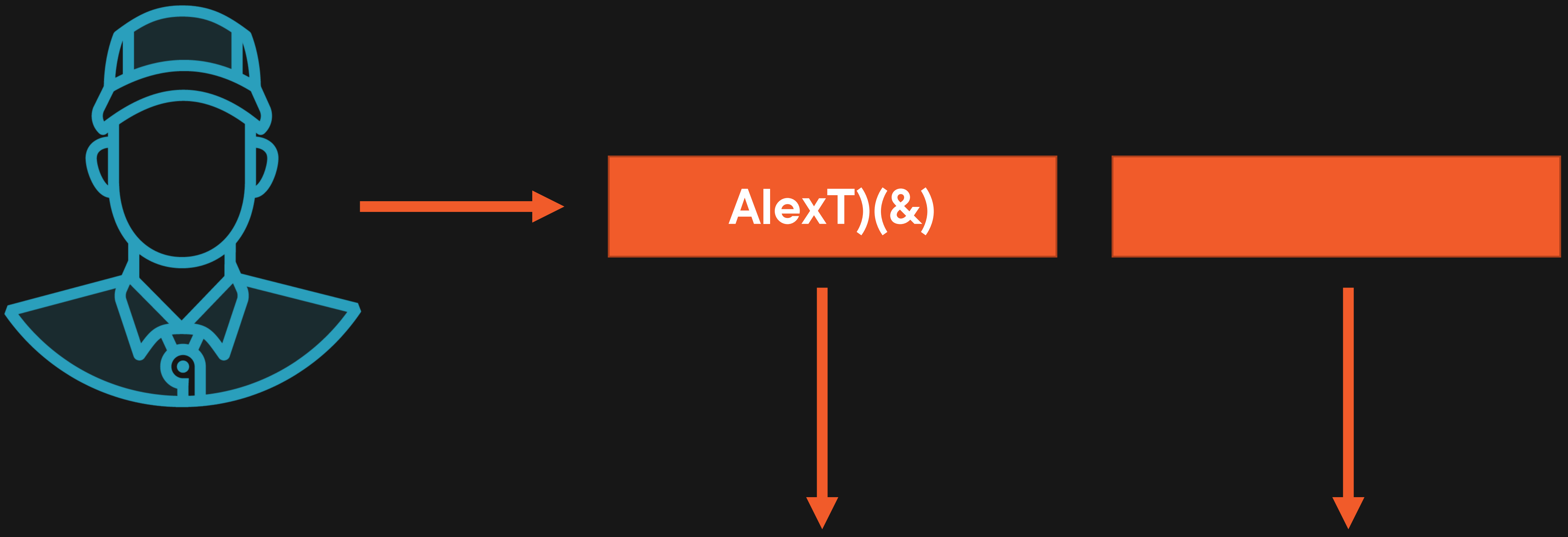
- Lightweight Directory Access Protocol (LDAP)
- Directory service protocol
  - Helps find resources on a network (people, files, end-points)
- Works with Active Directory and many other services
- Can be susceptible to injection

LDAP Injection Flaw



```
(&(USER=AlexT)(PASSWORD=MySecurePwd123!))
```

LDAP Injection Flaw



(&(USER=AlexT)(&))(PASSWORD=))

OS Injection

- C# can execute external commands via `System.Diagnostics.Process.Start`
- Used to trigger external processes from a web application
- Can be used to expose sensitive data or gain additional privileges

OS Injection

Code Snippet to Display .NET 6 Application Settings

```
var filePath = @"C:\Site\Pluralsight\VulnerableCart\appsettings.json";
var externalProcess = new Process
{
    StartInfo = new ProcessStartInfo
    {
        FileName = “cmd.exe",
        Arguments = "/C more " + filePath,
        UseShellExecute = false,
        RedirectStandardOutput = true,
        CreateNoWindow = true
    }
};
var output = "";
externalProcess.Start();
while (!externalProcess.StandardOutput.EndOfStream)
{
    output += externalProcess.StandardOutput.ReadLine();
}
```

Injection Scenarios

- Commands and user input combined
- We’re using input provided
- The result is used as a command

Demo

Injection Examples

- SQL Injection
- OS Injection

Injection Flaws Remediation

OWASP Injection Controls

Proactive Controls

C4: Encode & Escape Data

C5: Validate All Input

C8: Protect Data Everywhere

ASVS

V1.5: Input and Output Architecture

V5.1: Input Validation

V5.2: Sanitization and Sandboxing

V5.3: Output Encoding and Injection Prevention

V8.1: General Data Protection

V8.3: Sensitive Private Data

General Guidelines

- Never directly let user input interact with commands
- Parameterized queries
- Validated, and sanitized input
- Principal of least privilege
- Use a Visual Studio extension, such as PumaScan

SQL Injection

- Create parameters for all input
- Specify data type and length
- Instead of FromSqlRaw or ExecuteRawSql methods, use the FromSqlInterpolated method
- Avoid SQL altogether and use LINQ

```
var user = _context.Users
    .FromSqlRaw($"select * from dbo.[user] " +
            $"where username = '{username}' " +
            $"and password = '{password}'")
    .FirstOrDefault();
```

◀ **Avoid directly injecting input into the query.**

```
var userparam = new SqlParameter("userparam", SqlDbType.VarChar, 15)
{
    Value = username
};
var pwdparam = new SqlParameter("pwdparam", SqlDbType.VarChar, 15)
{
    Value = password
};


var user = _context.Users
    .FromSqlRaw("select * from dbo.[user] " +
                "where username = @userparam " +
                "and password = @pwdparam", userparam, pwdparam)
    .FirstOrDefault();
```

Parameterized Query

By using parameters, we avoid SQL Injections. The data passed via parameters is taken literally and does not become part of the SQL statement that executes against the database.

LDAP Injection

- Validate input
- Use AD to LINQ library

OS Injection

- Consider API instead of directly running commands
- Validate all input
- Process should run with least privilege
- Run command based on user input, not using user input

Demo

Remediation

- Fix the SQL Injection
- Validate input to external process

Summary

Injection Flaws

- Looked at SQL, OS, and LDAP injections
- Can be devastating to an application
- Never rely on raw user input
    - Validate and escape user input
    - Use parameters
- Injectable code is common, so be careful!
- Use a static code analysis tool to catch mistakes

Up Next:
Insecure Design