# CHAPTER 10

# Securing Data and Applications

TESTOUT SECURITY PRO

# Data Transmission Security

# Section Skill Overview

- Add SSL to a website.
- Allow SSL connections.
- Require IPsec for communications.

# Key Terms

- Secure Sockets Layer (SSL)
- Transport Layer Security (TLS)
- Secure Shell (SSH)
- Hyper Text Transfer Protocol Secure (HTTPS)
- Secure Hypertext Transfer Protocol (S-HTTP)
- Internet Protocol Security (IPsec)
- Authentication Header (AH)
- Encapsulating Security Payload (ESP)
- Security Association (SA)

# Key Definitions

- **Secure Sockets Layer (SSL):** A protocol that secures messages being transmitted on the internet.
- **Transport Layer Security (TLS):** A protocol that secures messages being transmitted on the internet. It is the successor to SSL 3.0.
- **Secure Shell (SSH):** A protocol that allows for secure interactive control of remote systems.
- **Hyper Text Transfer Protocol Secure (HTTPS):** A secure form of HTTP that uses either SSL or TLS to encrypt sensitive data before it is transmitted.
- **Secure Hypertext Transfer Protocol (S -HTTP):** An alternate protocol that is not widely used because it is not as secure as HTTPS.

# Key Definitions

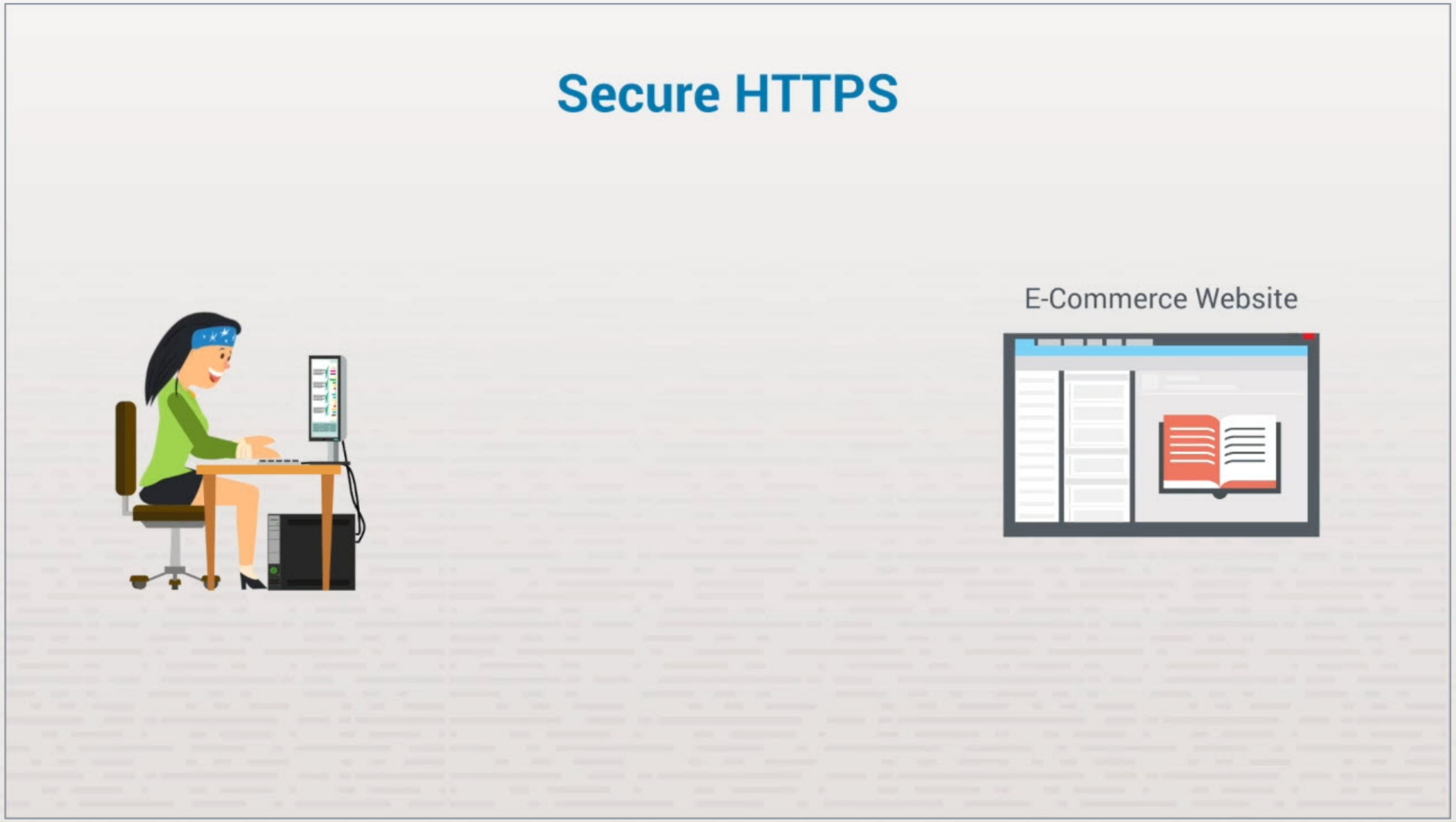
- **Internet Protocol Security (IPsec):** A set of protocols that provides secure data transmission over unprotected TCP/IP networks.
- **Authentication Header (AH):** A protocol within IPsec that provides authenticity, non-repudiation, and integrity.
- **Encapsulating Security Payload (ESP):** A protocol within IPsec that provides all the security of AH plus confidentiality.
- **Security Association (SA):** The establishment of shared security information between two network entities to support secure communications.

# Secure Protocols

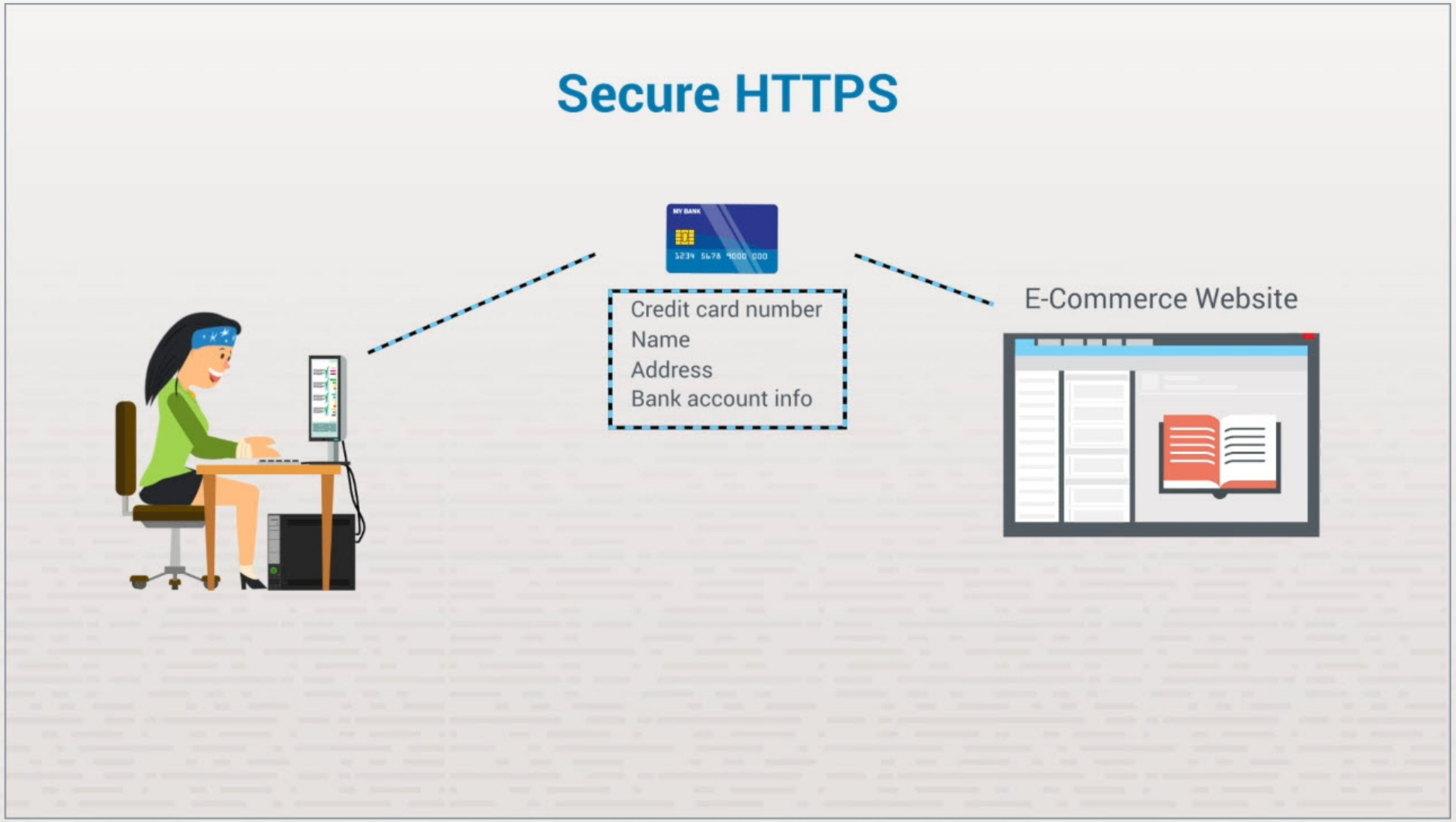
# Secure Sockets Layer

- Provides security for TCP/IP
- HTTP utilizes SSL and TLS

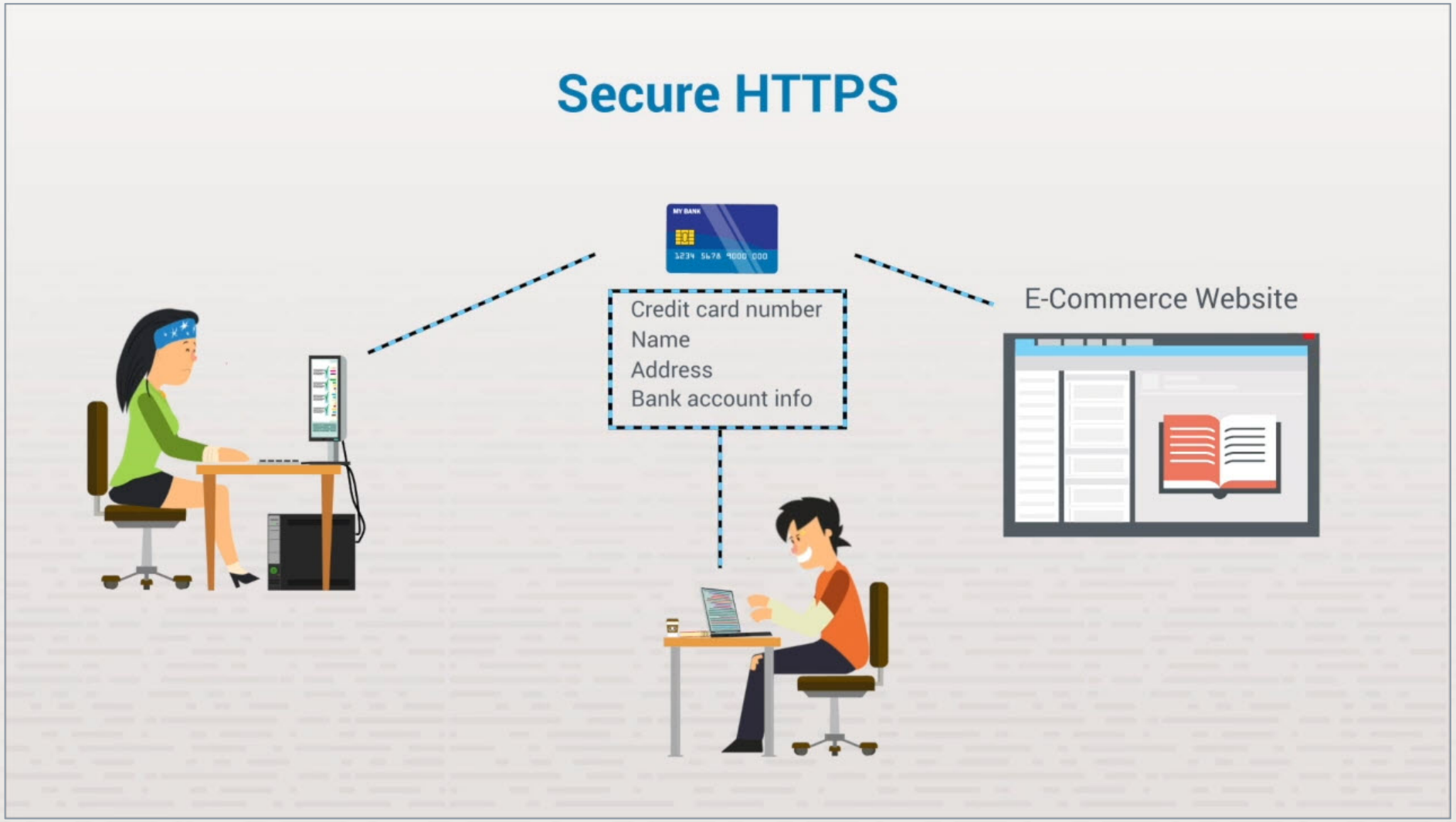
# Secure Protocols



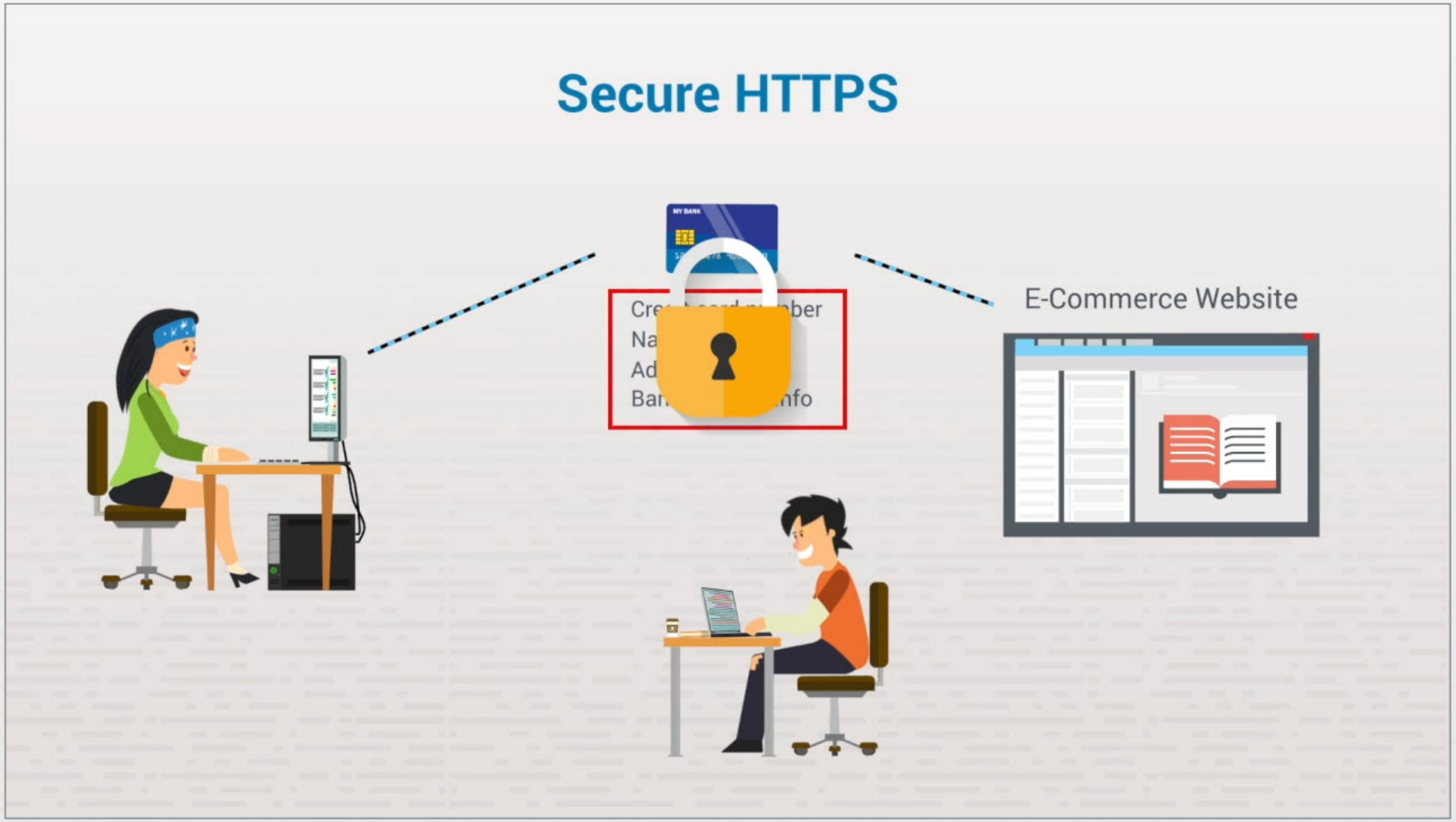
# Secure Protocols



# Secure Protocols



# Secure Protocols



# Secure Sockets Layer

- Provides security for TCP/IP
- Many app protocols use SSL:
  - HTTPS
  - LDAPS
  - FTPS

# Secure Protocols

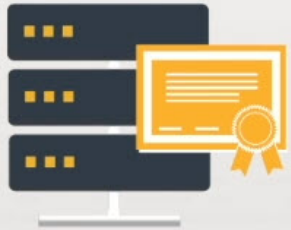
# Secure Protocols

## Certificates

Client

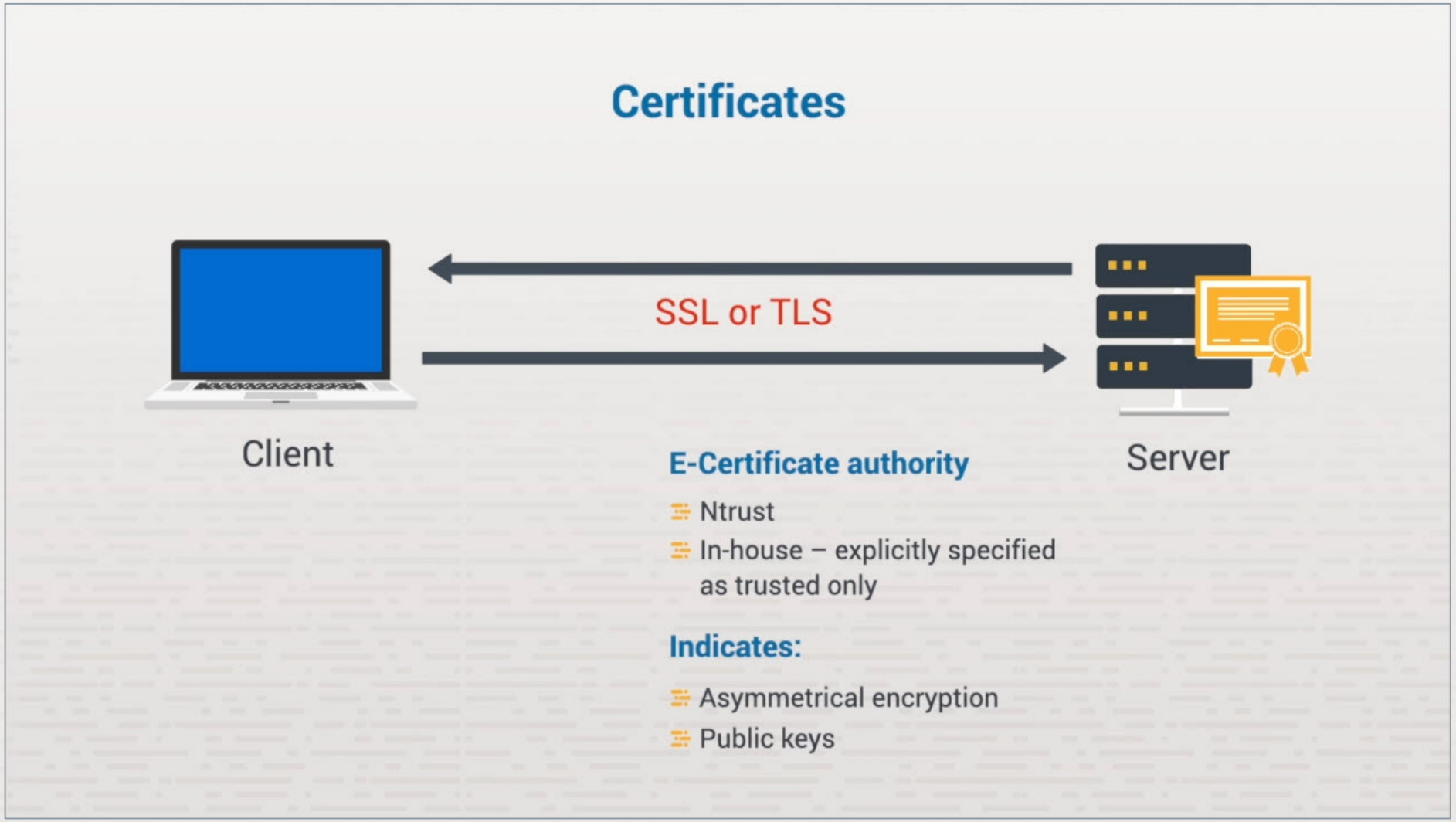
**E-Certificate authority**

- Ntrust
- In-house – explicitly specified as trusted only



Server

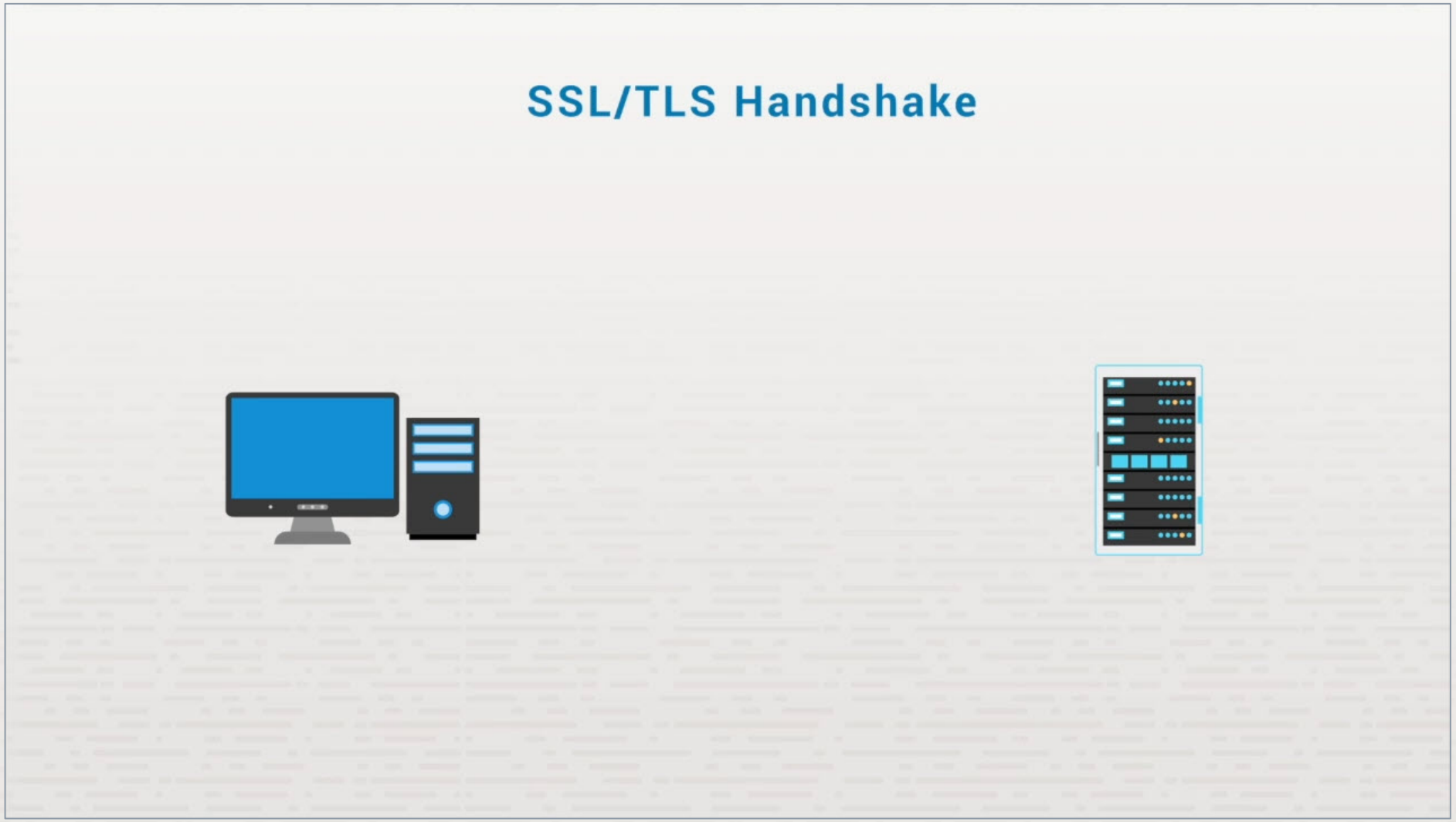
# Secure Protocols



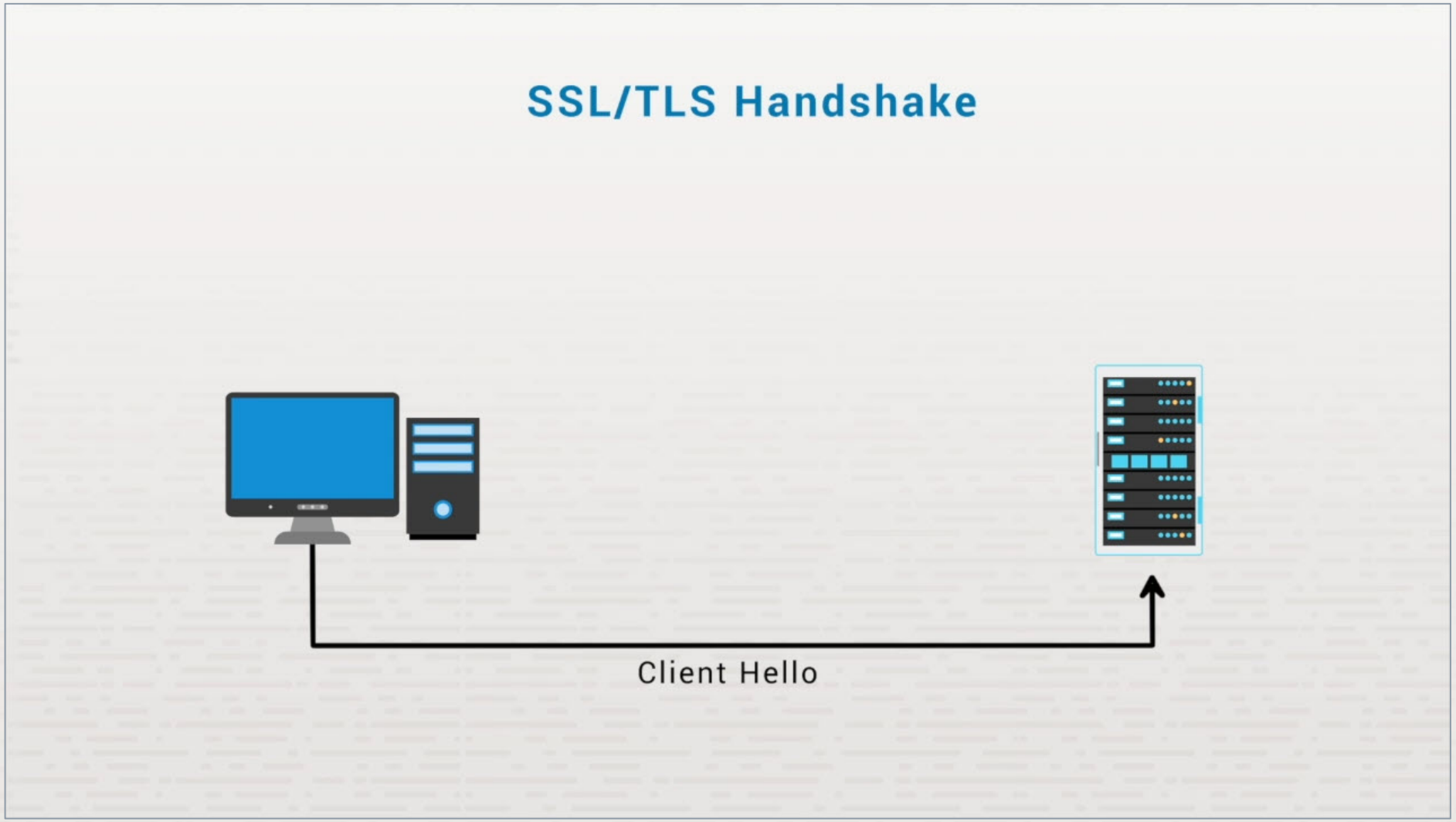
# TLS Facts

- More secure than SSL
- Not backwards compatible
- Application-independent
- App decides TLS handshake
- Uses Diffie-Hellman
- A session key is used

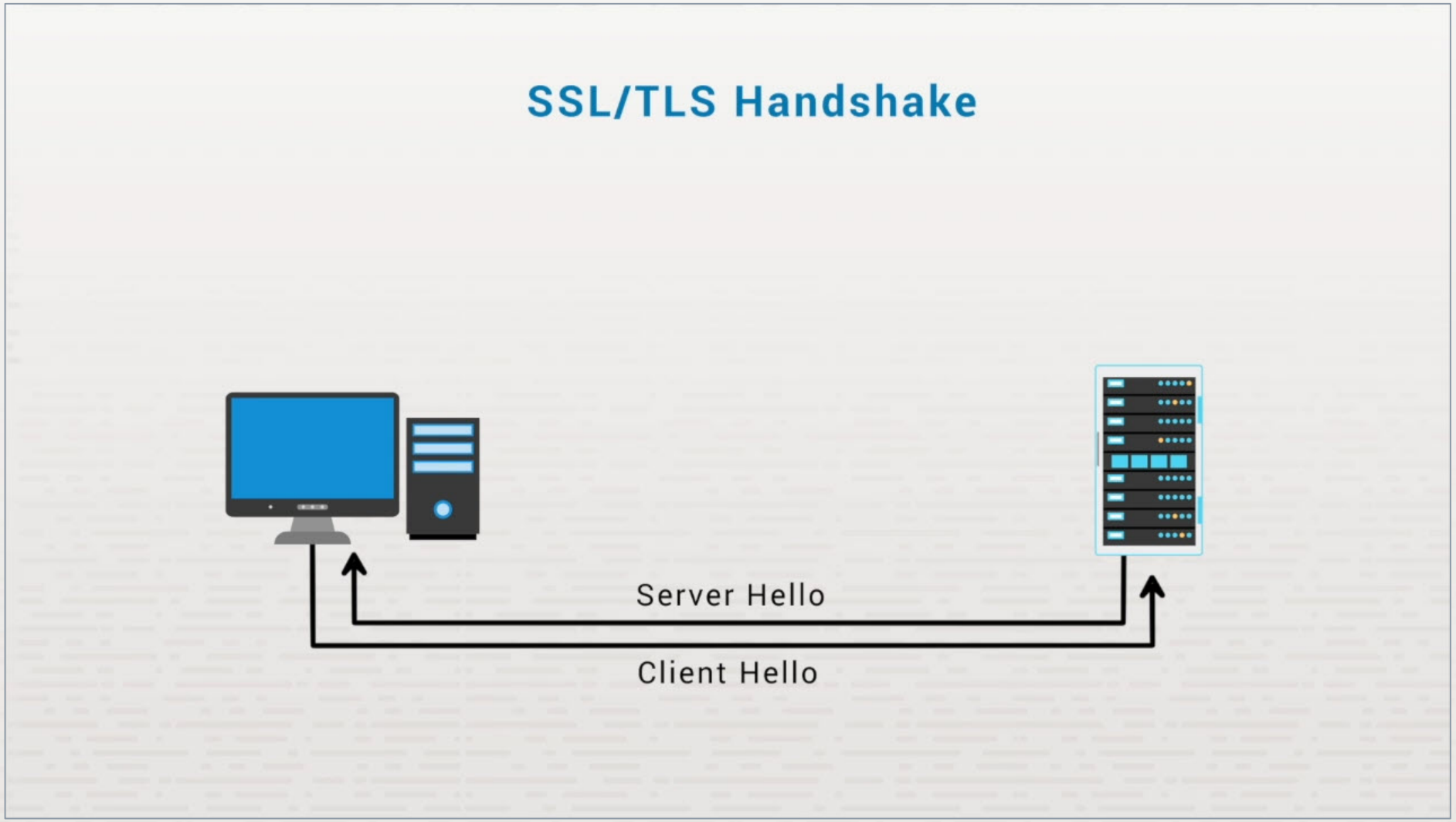
# Secure Protocols



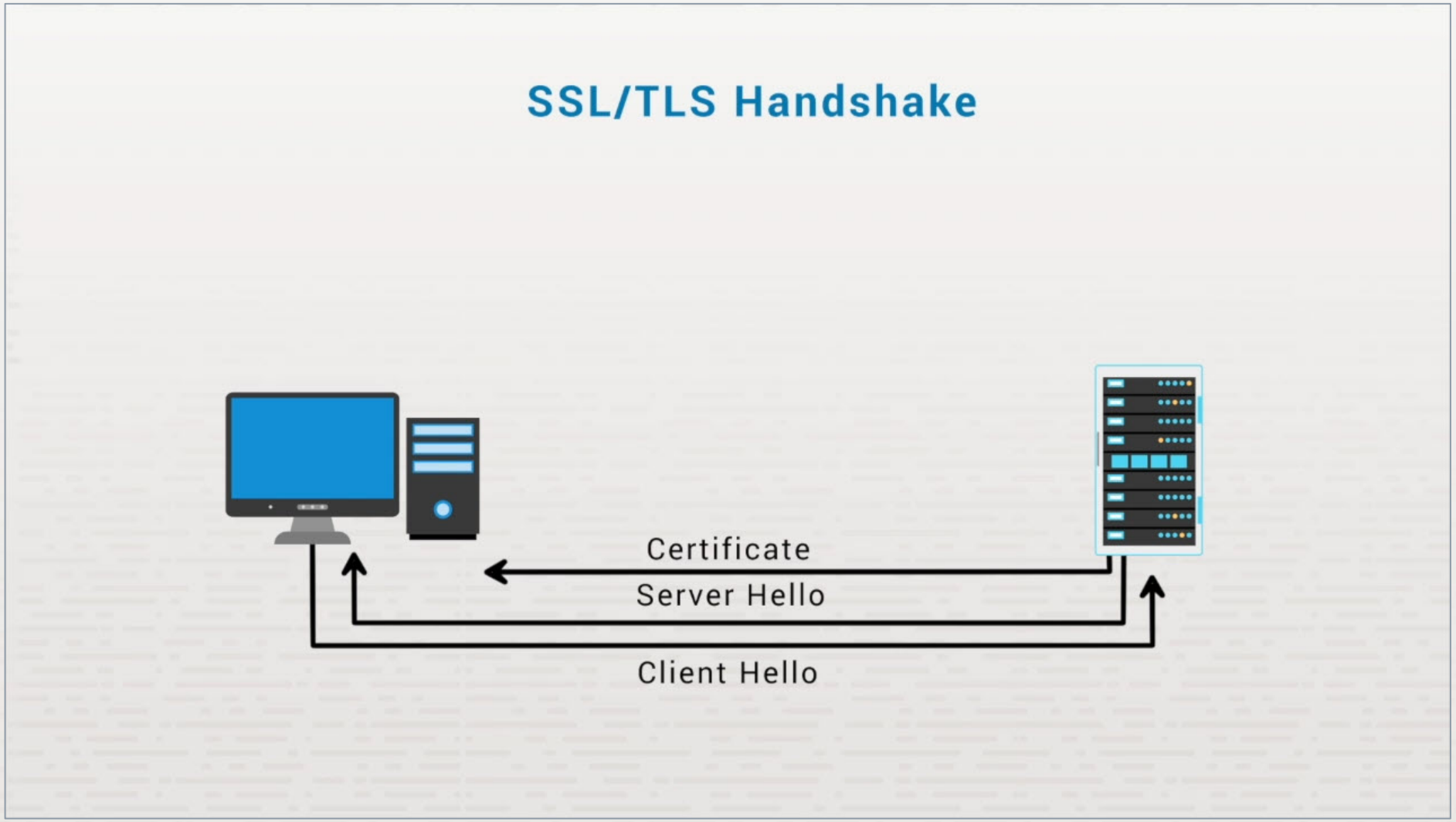
# Secure Protocols



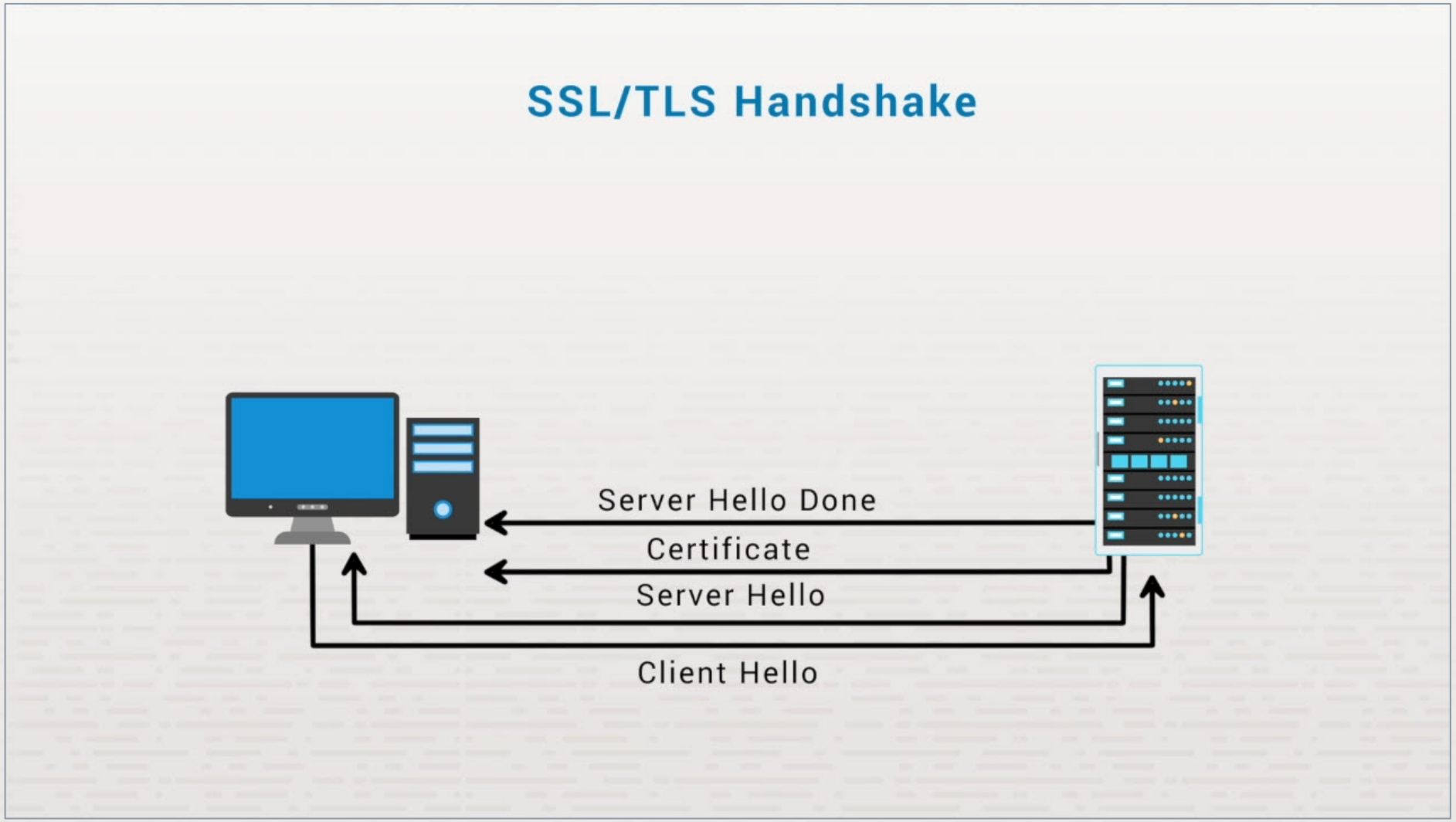
# Secure Protocols



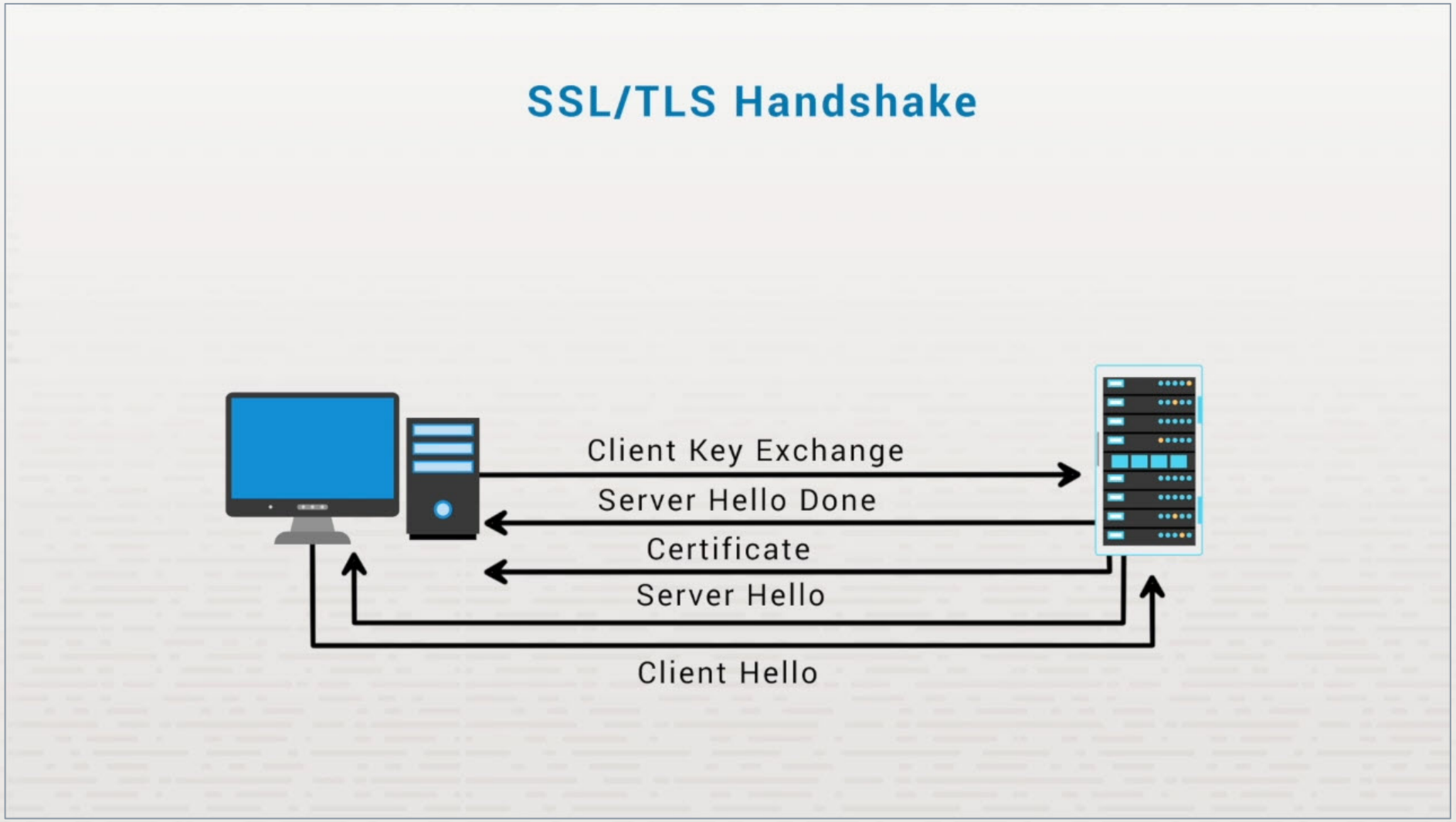
# Secure Protocols



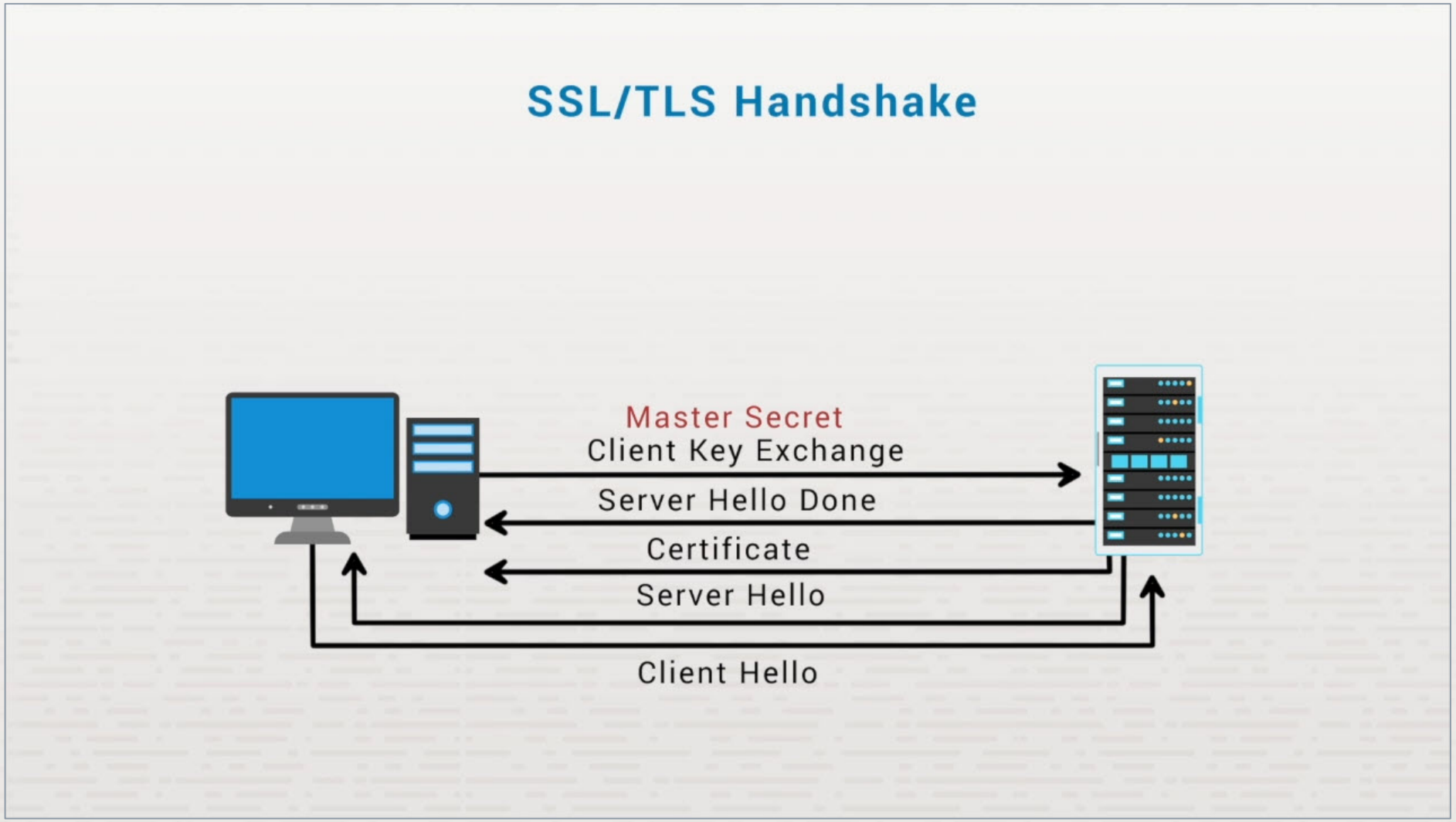
# Secure Protocols



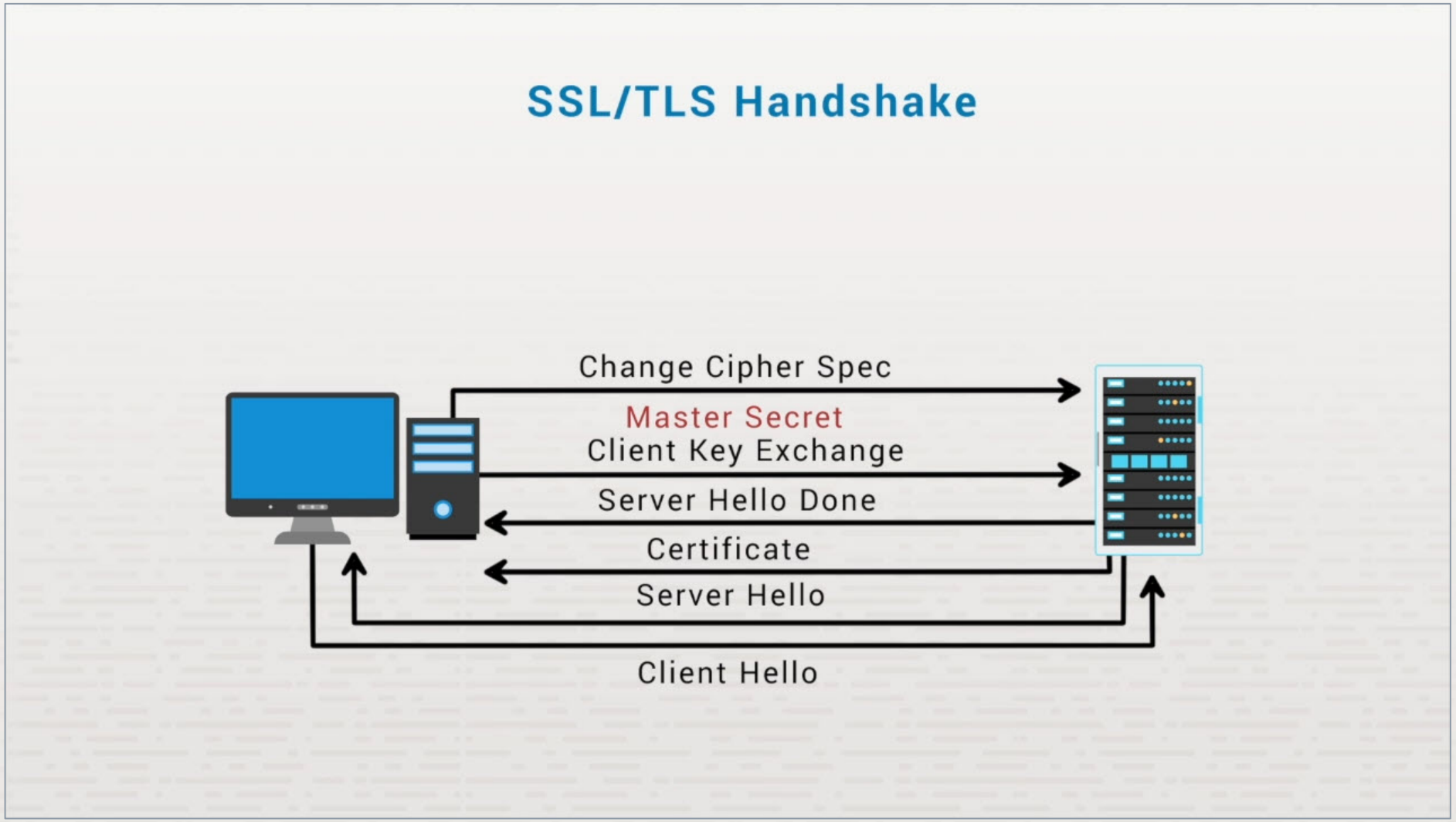
# Secure Protocols



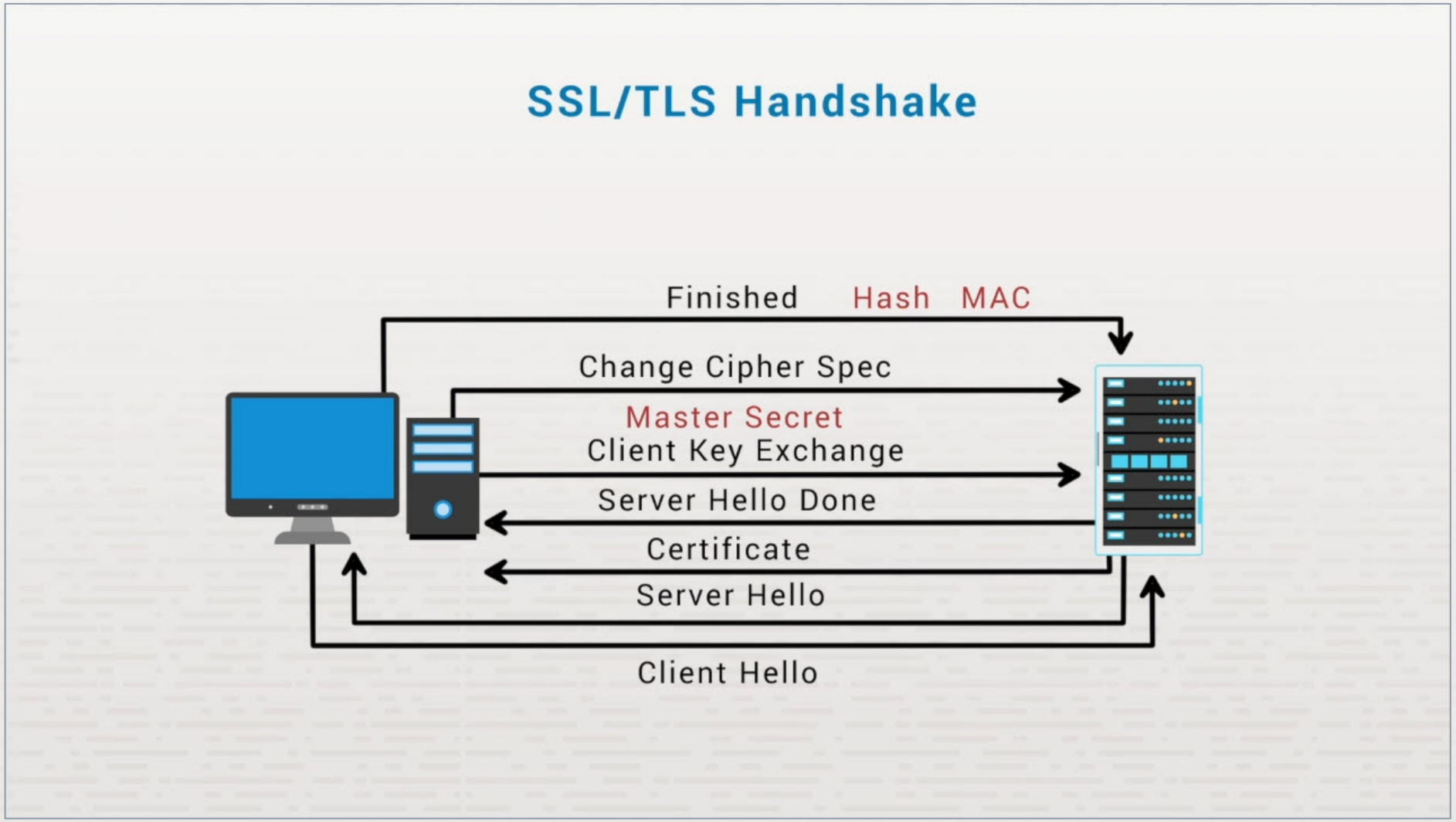
# Secure Protocols



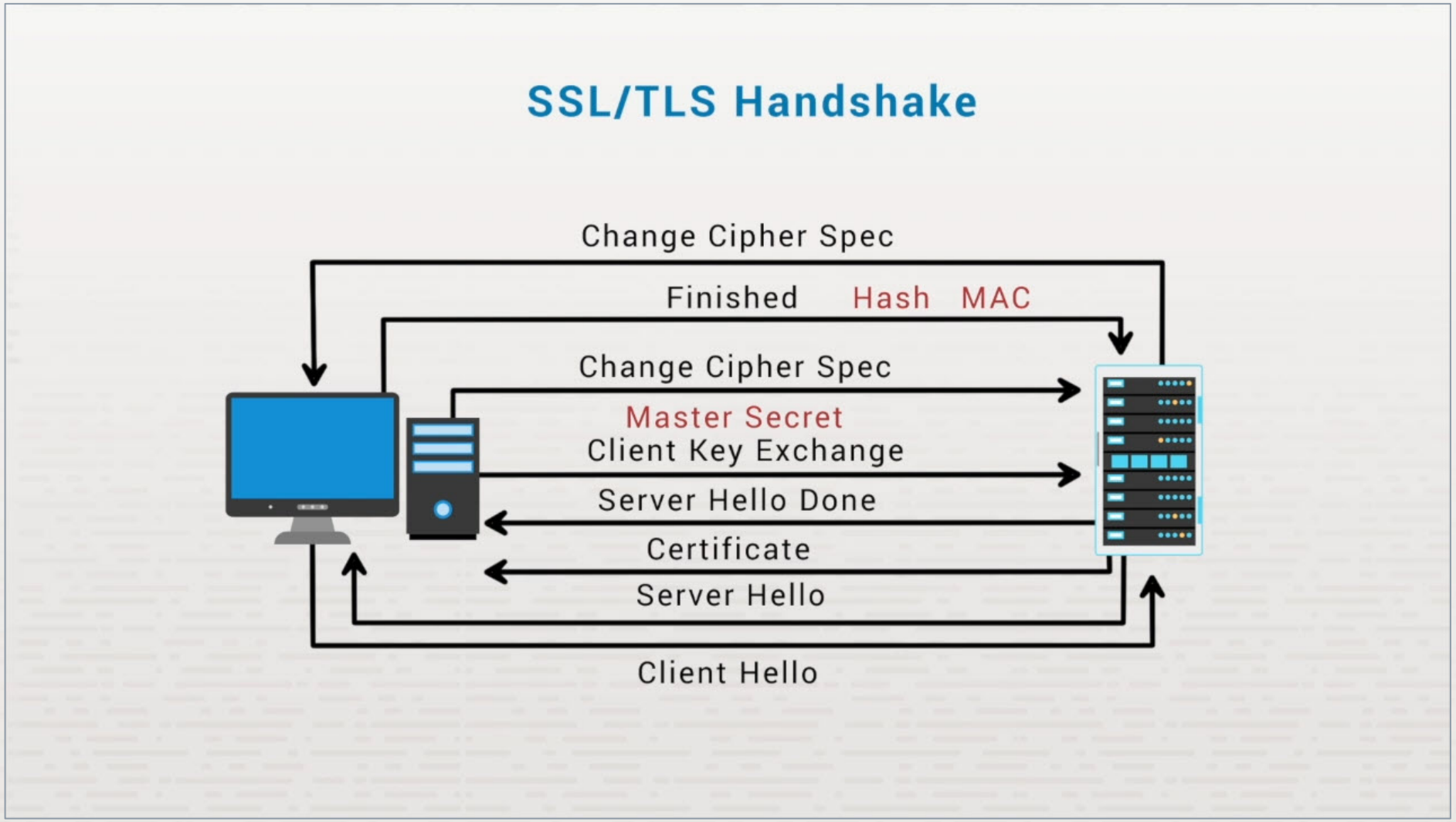
# Secure Protocols



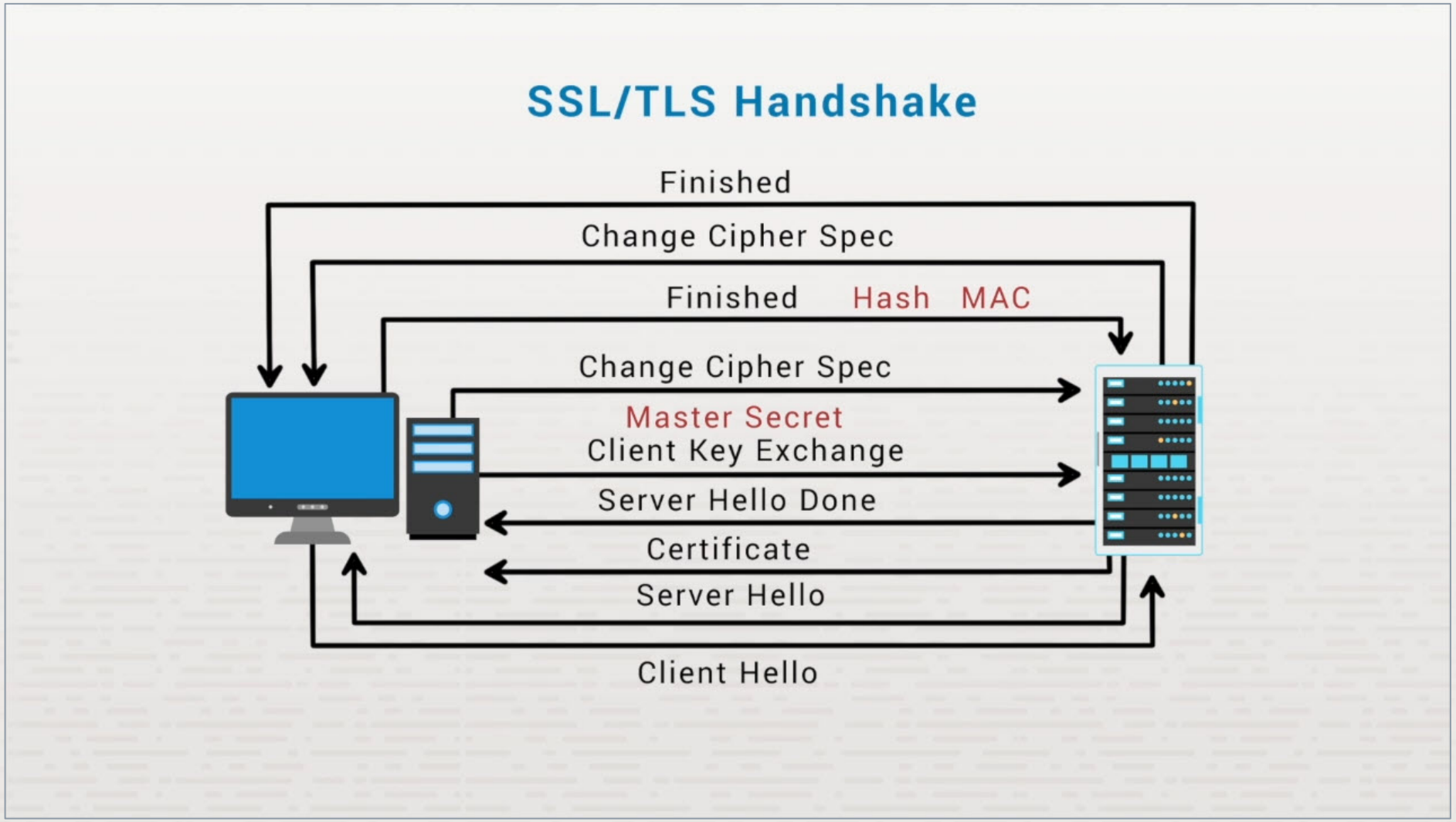
# Secure Protocols



# Secure Protocols



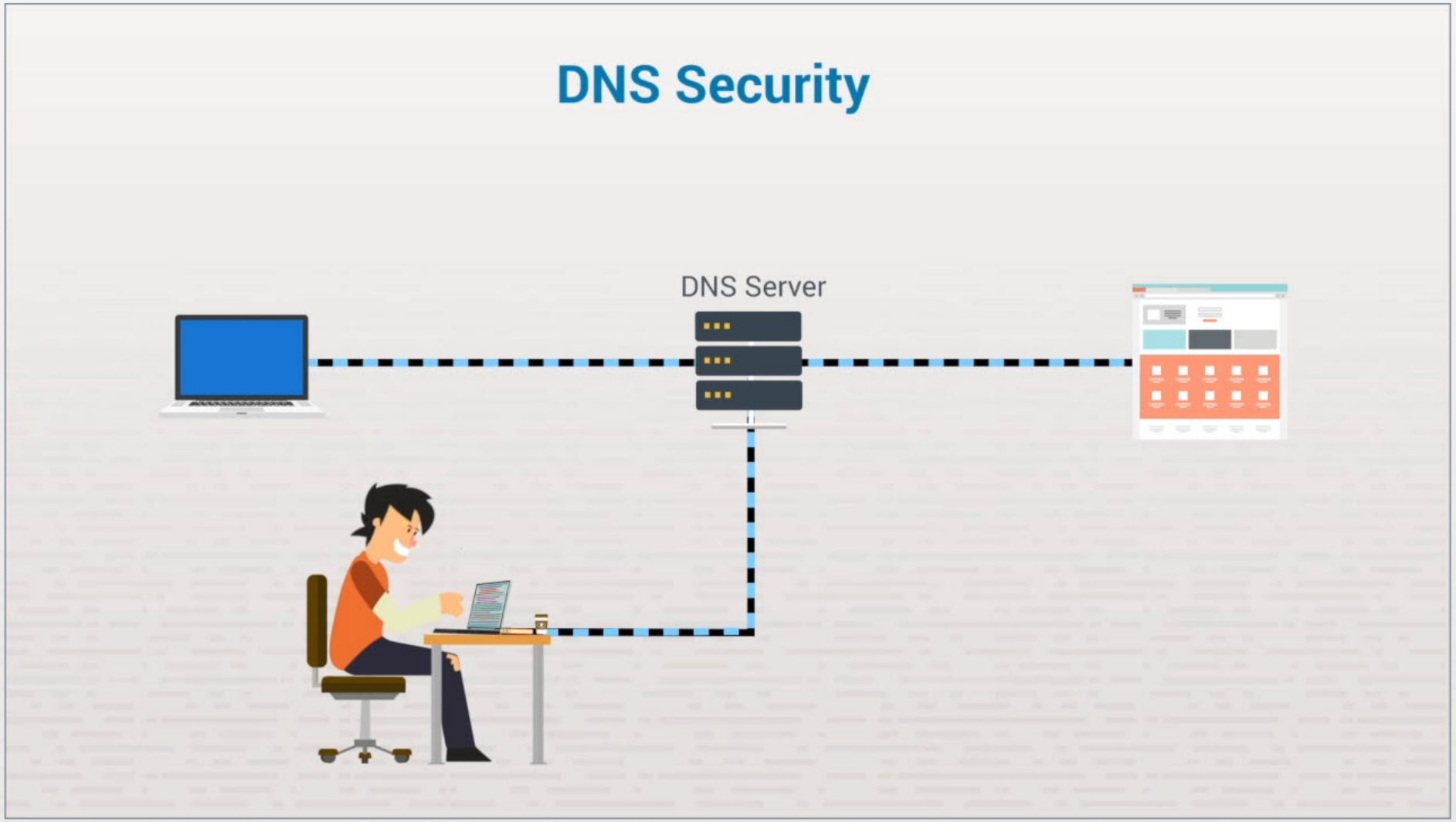
# Secure Protocols



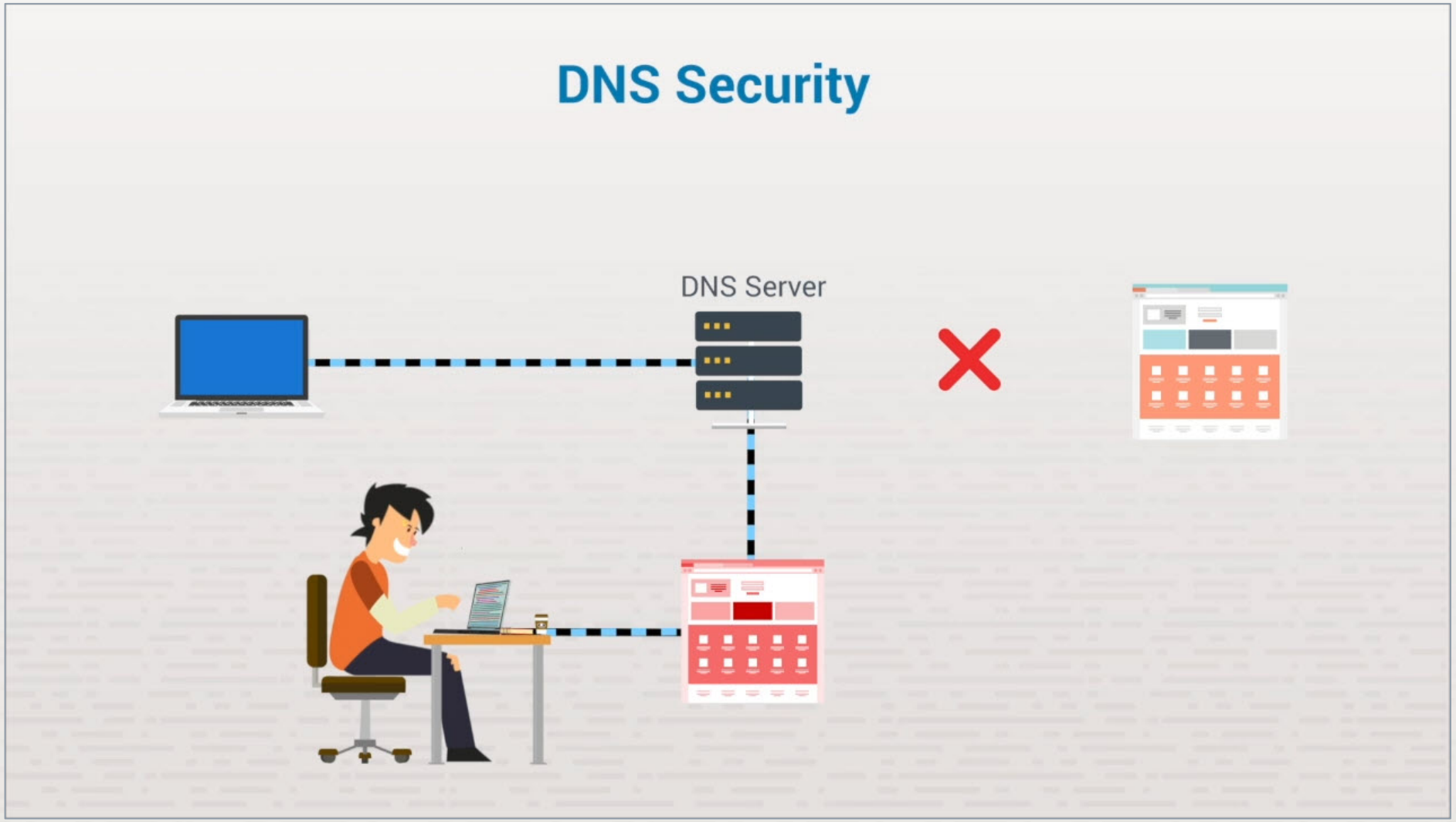
# DNS Security

- Internet's "phone book"
- Communications in cleartext
- Open to attack

# Secure Protocols



# Secure Protocols



# DNS Protection

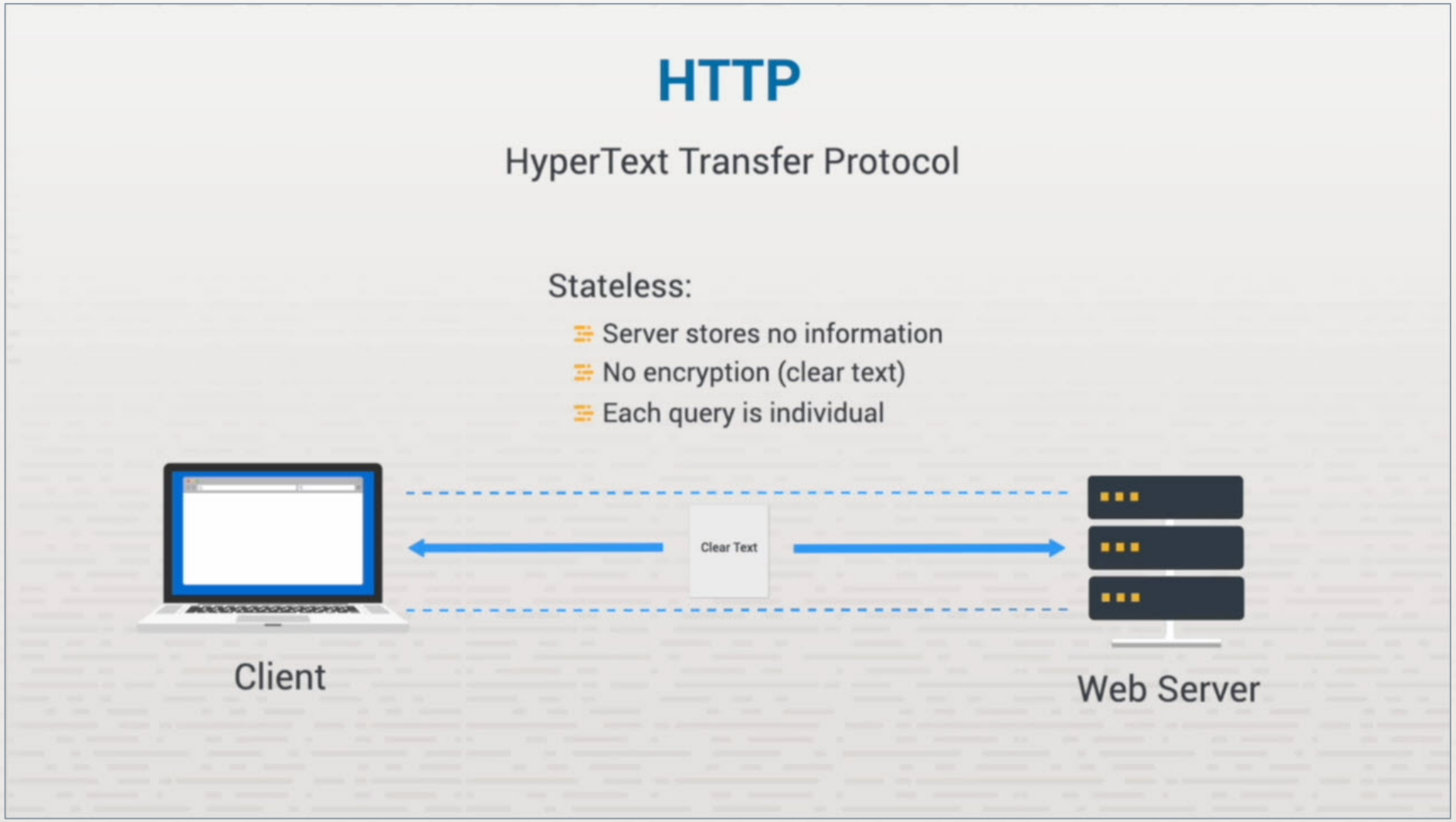
- DNS over TLS (DoT)
  - Encryption added to TCP
- DNS over HTTPS ( DoH)
  - Encrypts queries and responses
  - Integrated traffic flow with HTTP

# Summary

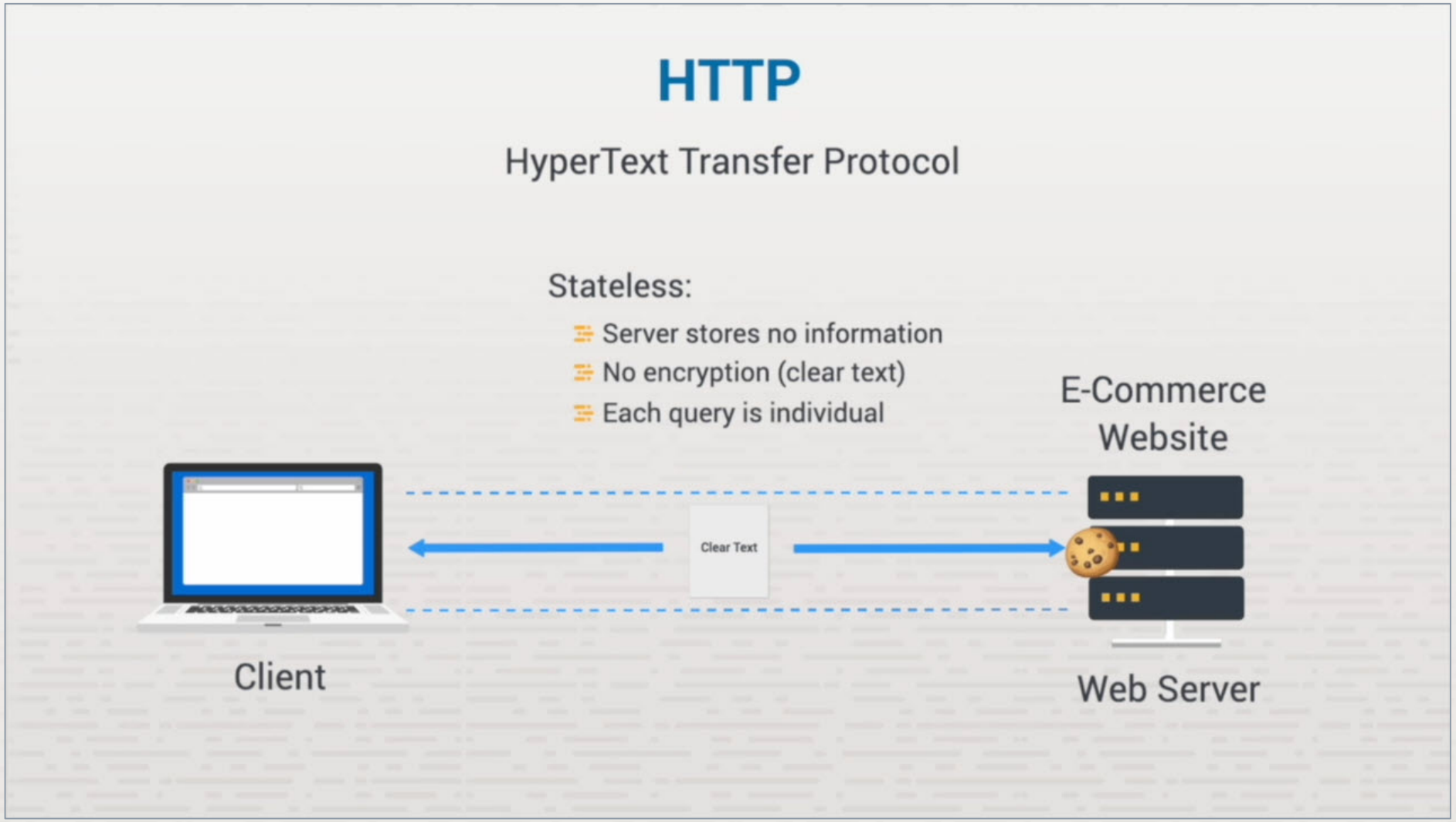
- SSL/TLS
- SSL handshake T
- LS facts
- DNS security

# Secure Protocols 2

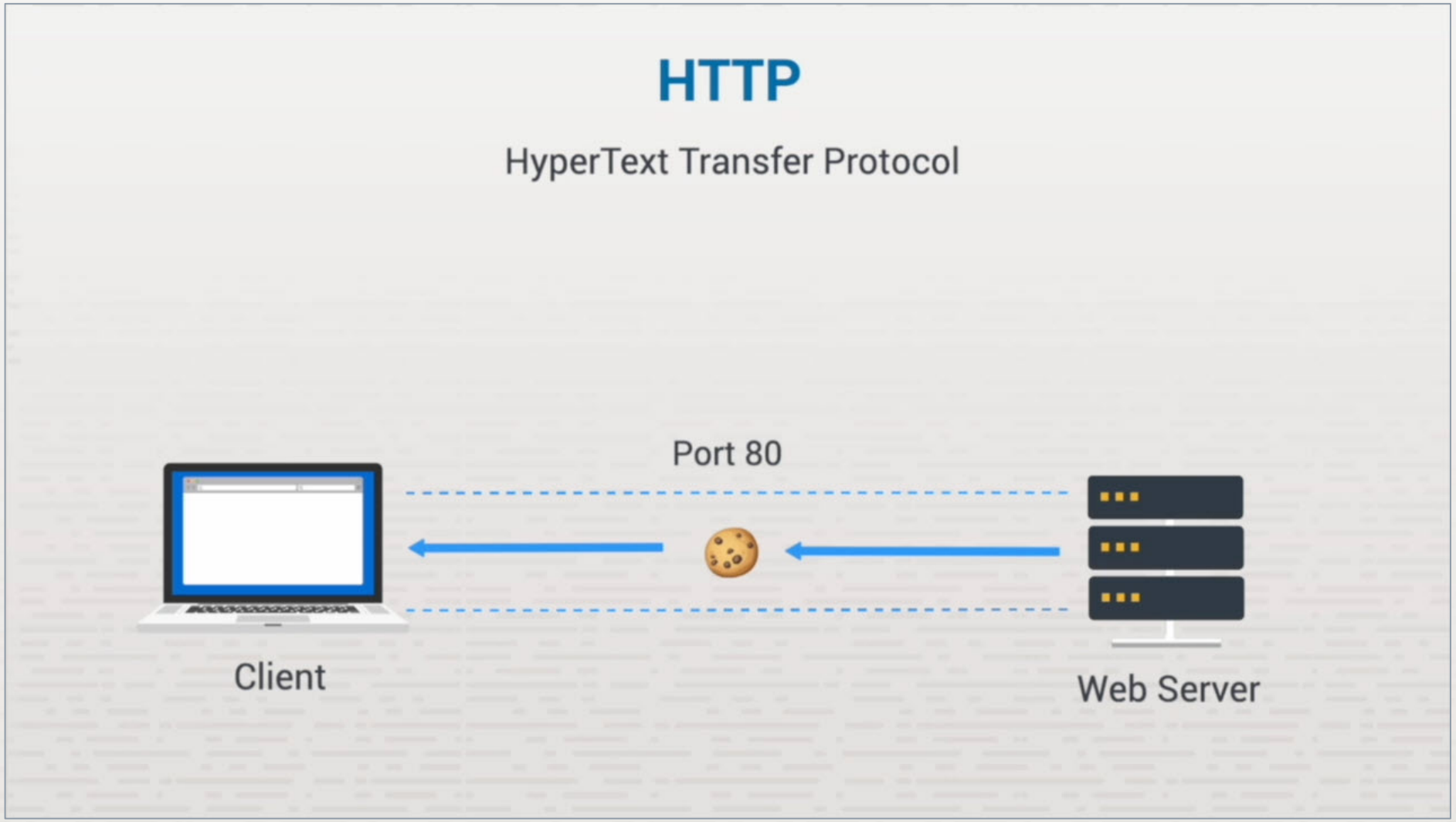
# Secure Protocols 2



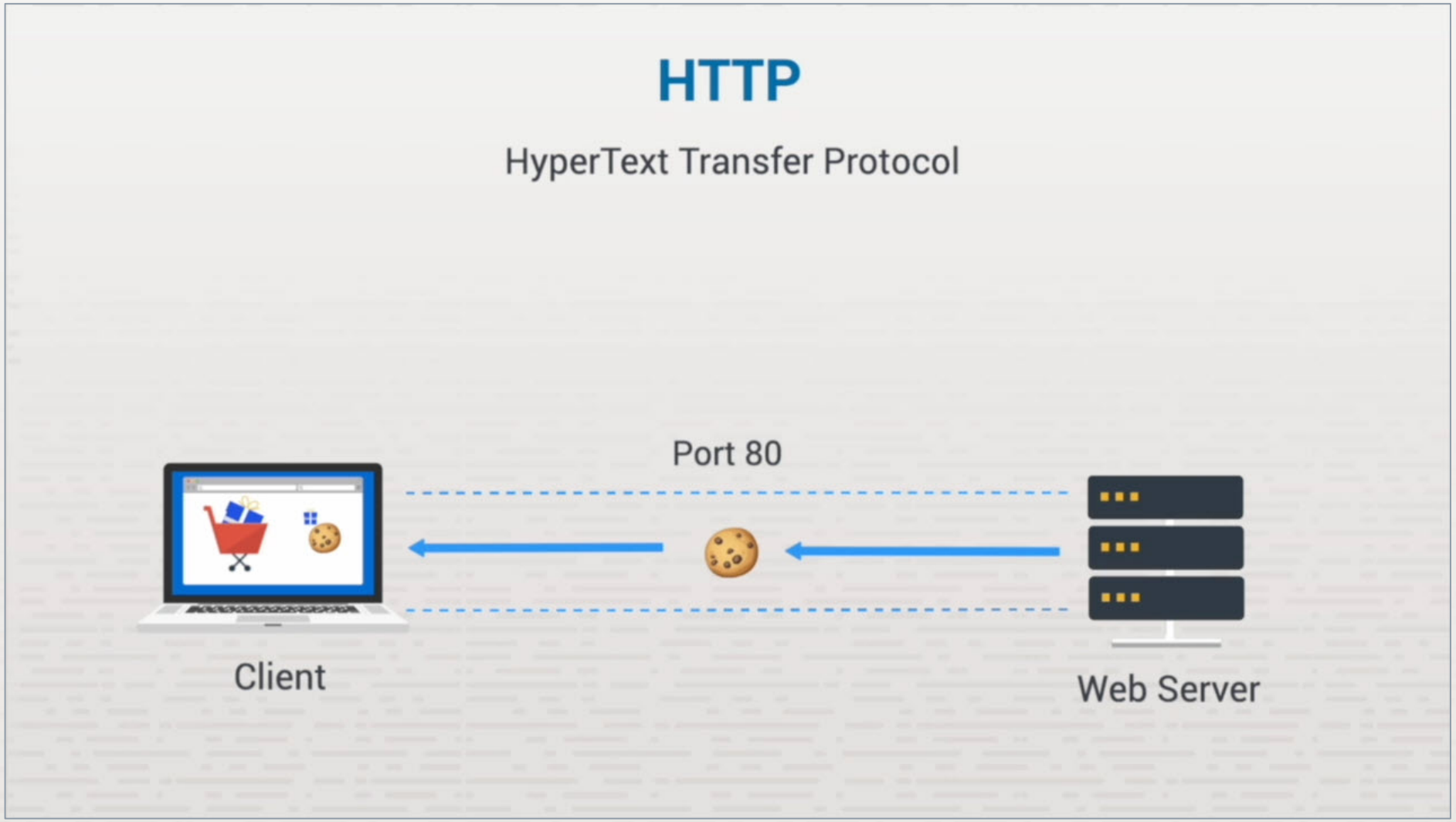
# Secure Protocols 2



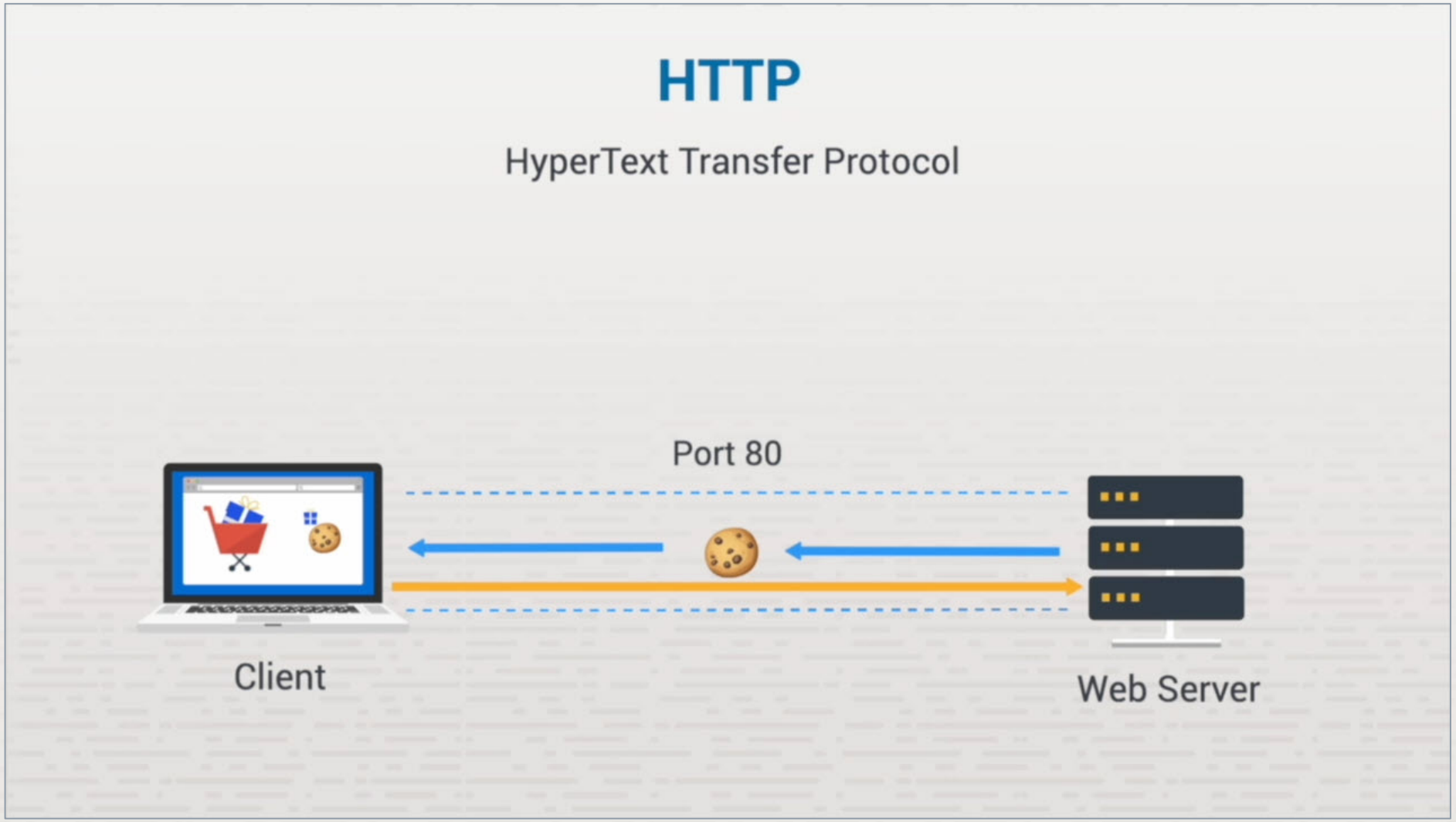
# Secure Protocols 2



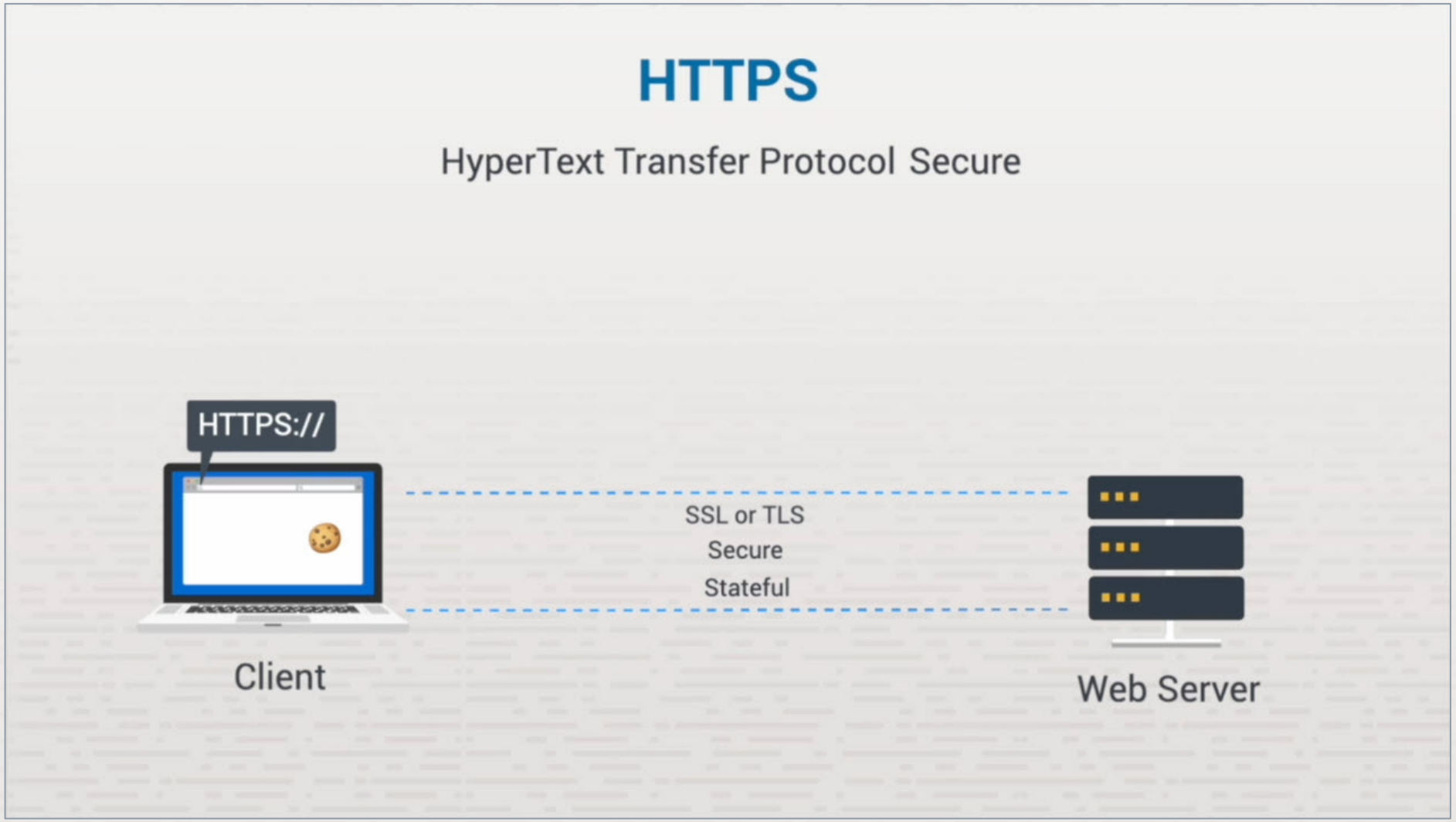
# Secure Protocols 2



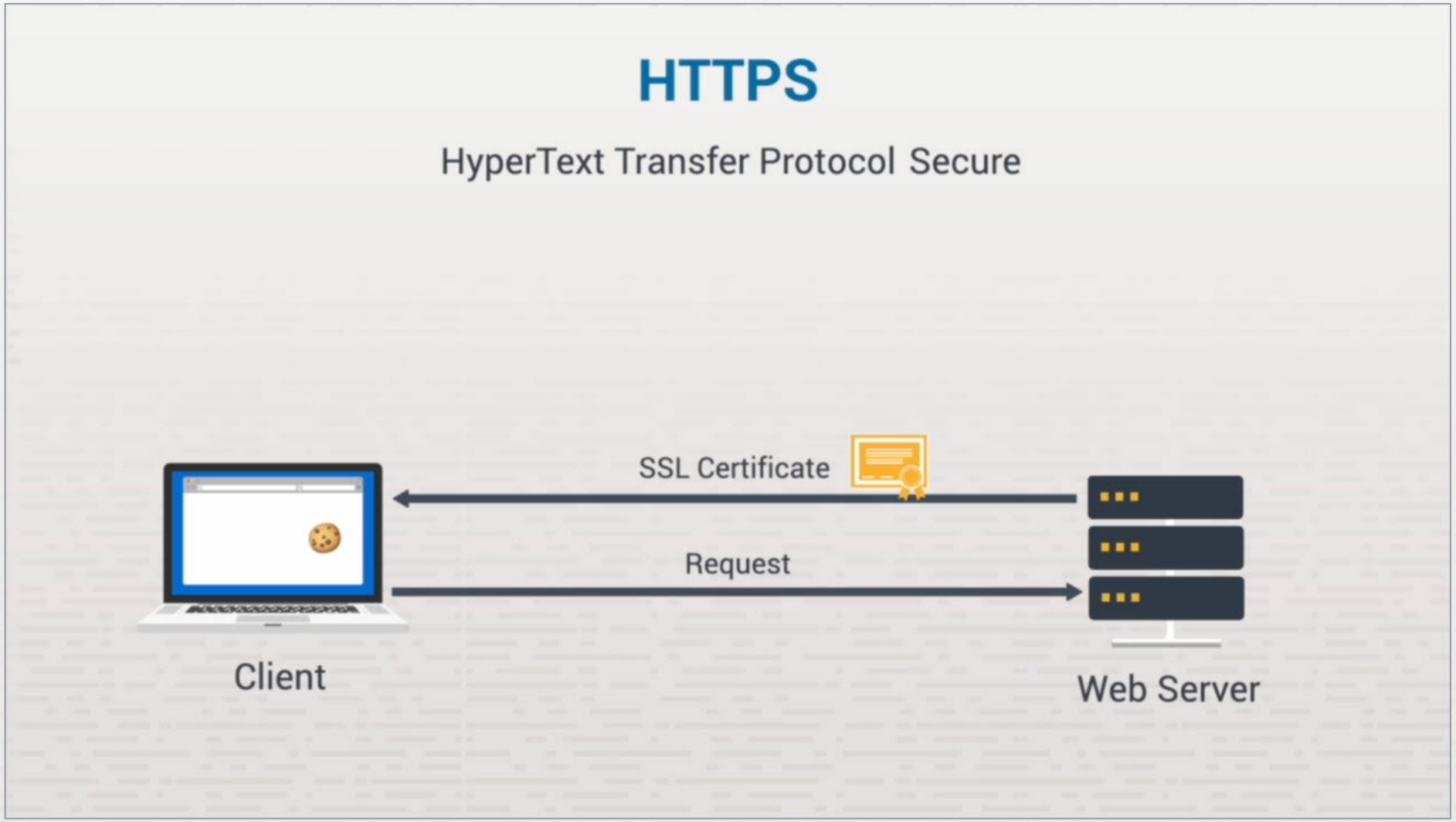
# Secure Protocols 2



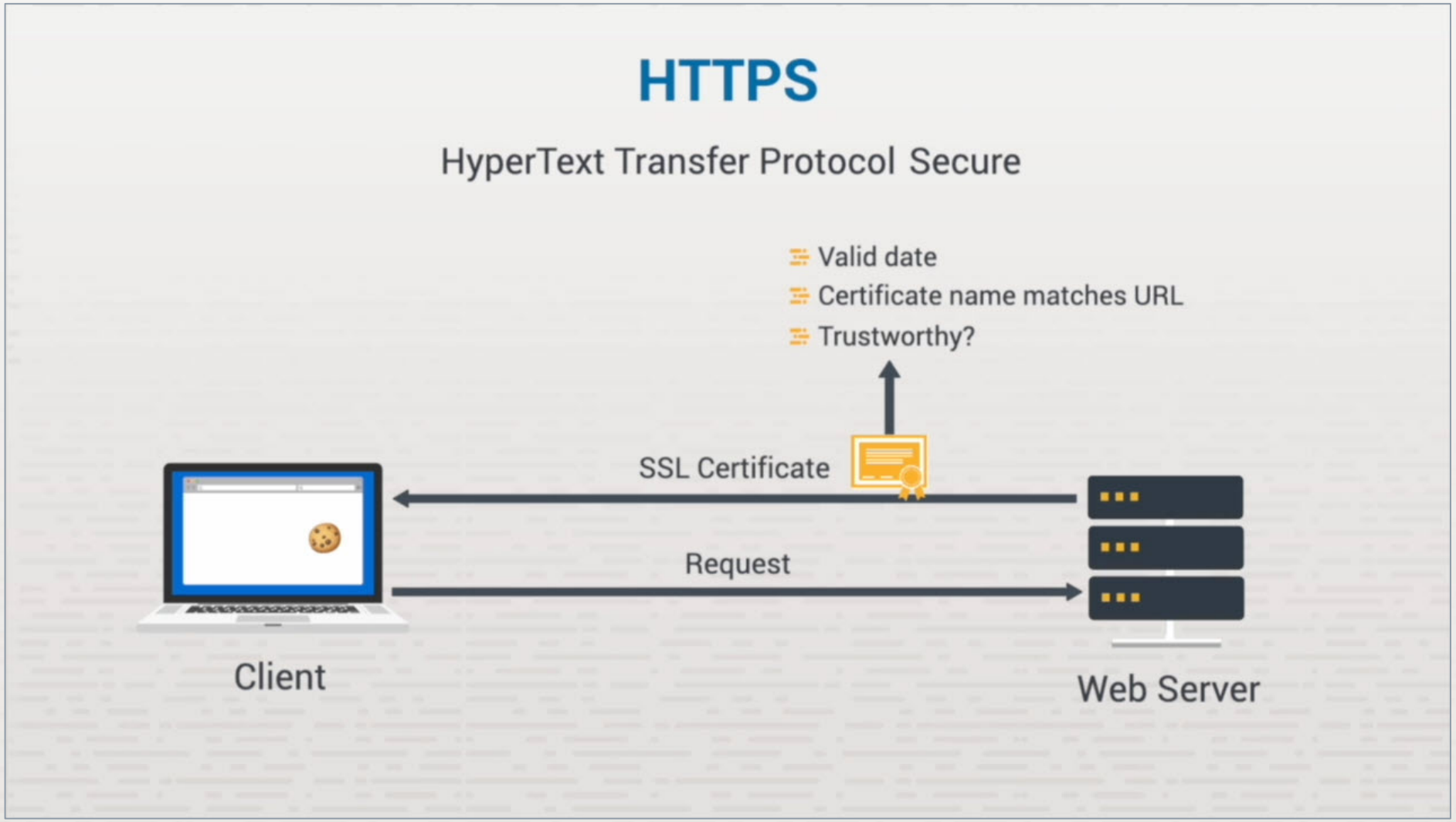
# Secure Protocols 2



# Secure Protocols 2



# Secure Protocols 2

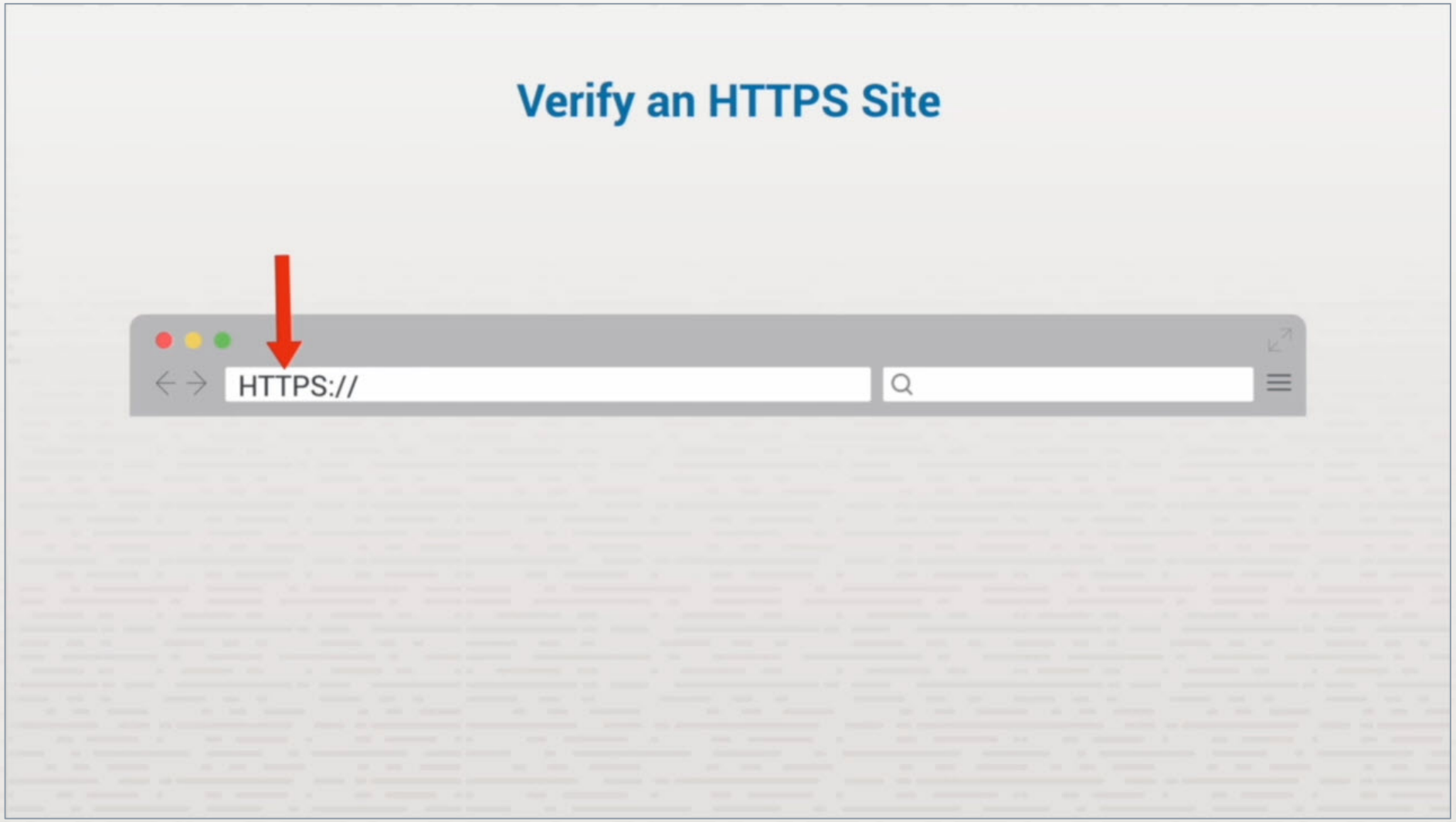


# Secure Protocols 2

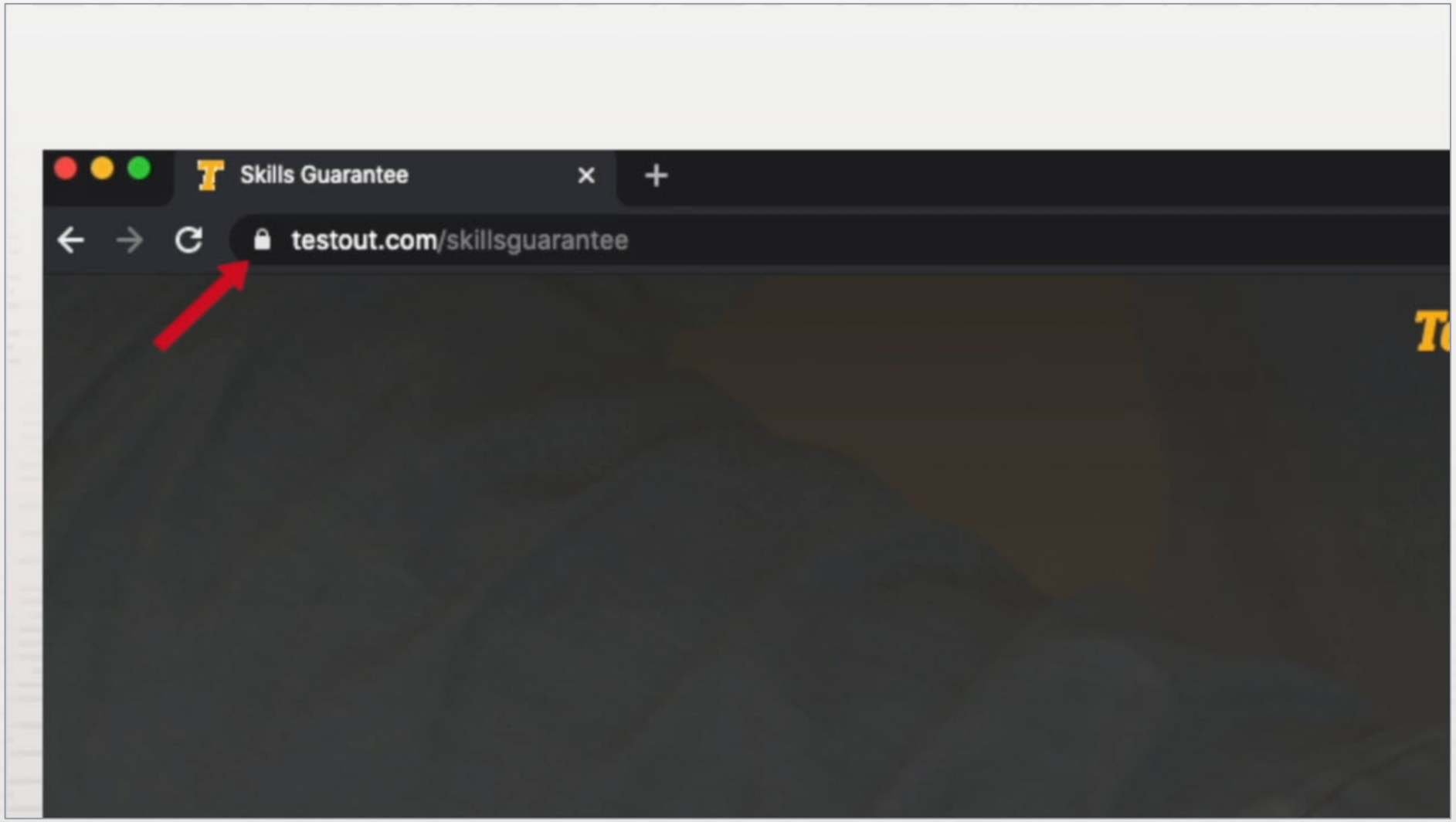
## Certificate Authorities

- **Trusted certificate authorities**
  - Reputable organizations
  - Pre-configured browser lists
- **Untrusted certificate authorities**
  - Not on pre-configured lists
  - Browser sends warning

# Secure Protocols 2

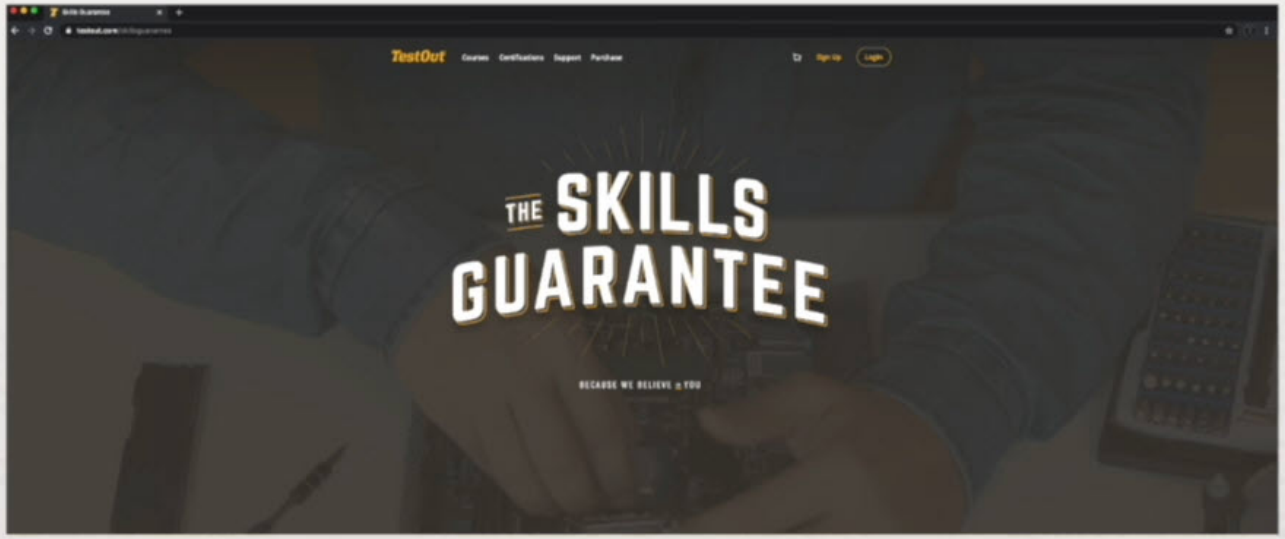


# Secure Protocols 2



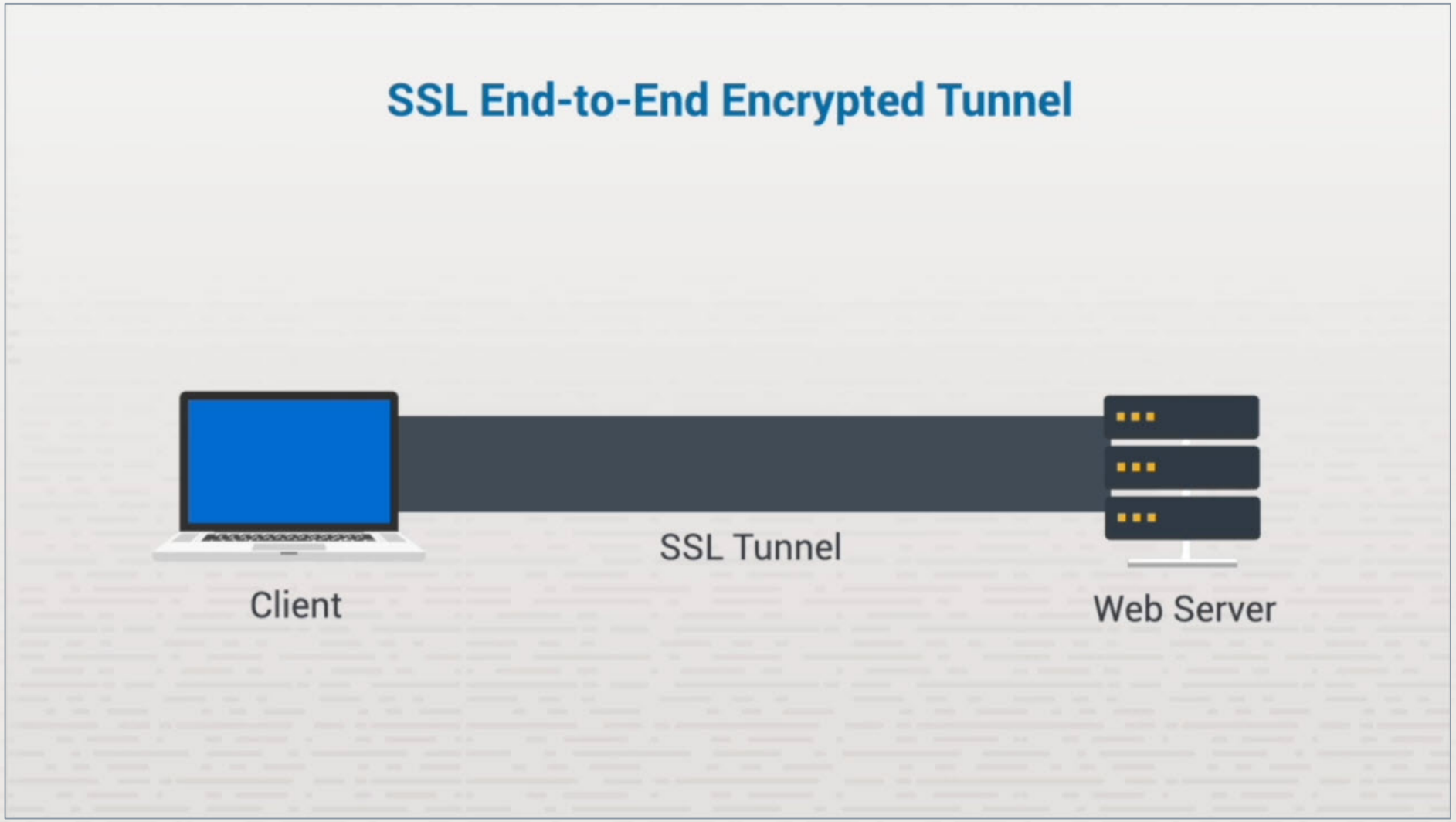
# Secure Protocols 2

## Verify an HTTPS Site

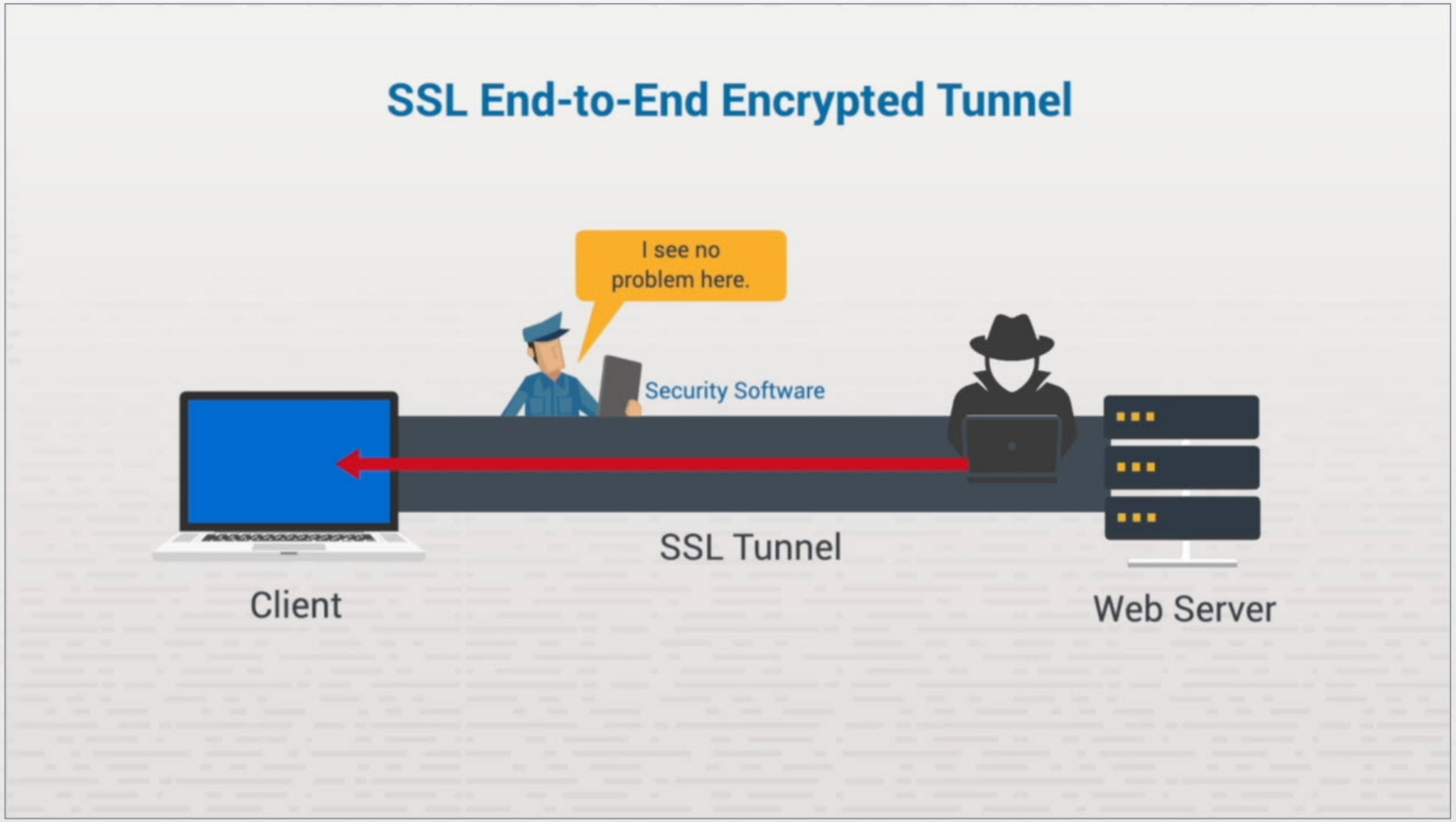



- HTTPS runs on port 443
- Configure firewall to allow

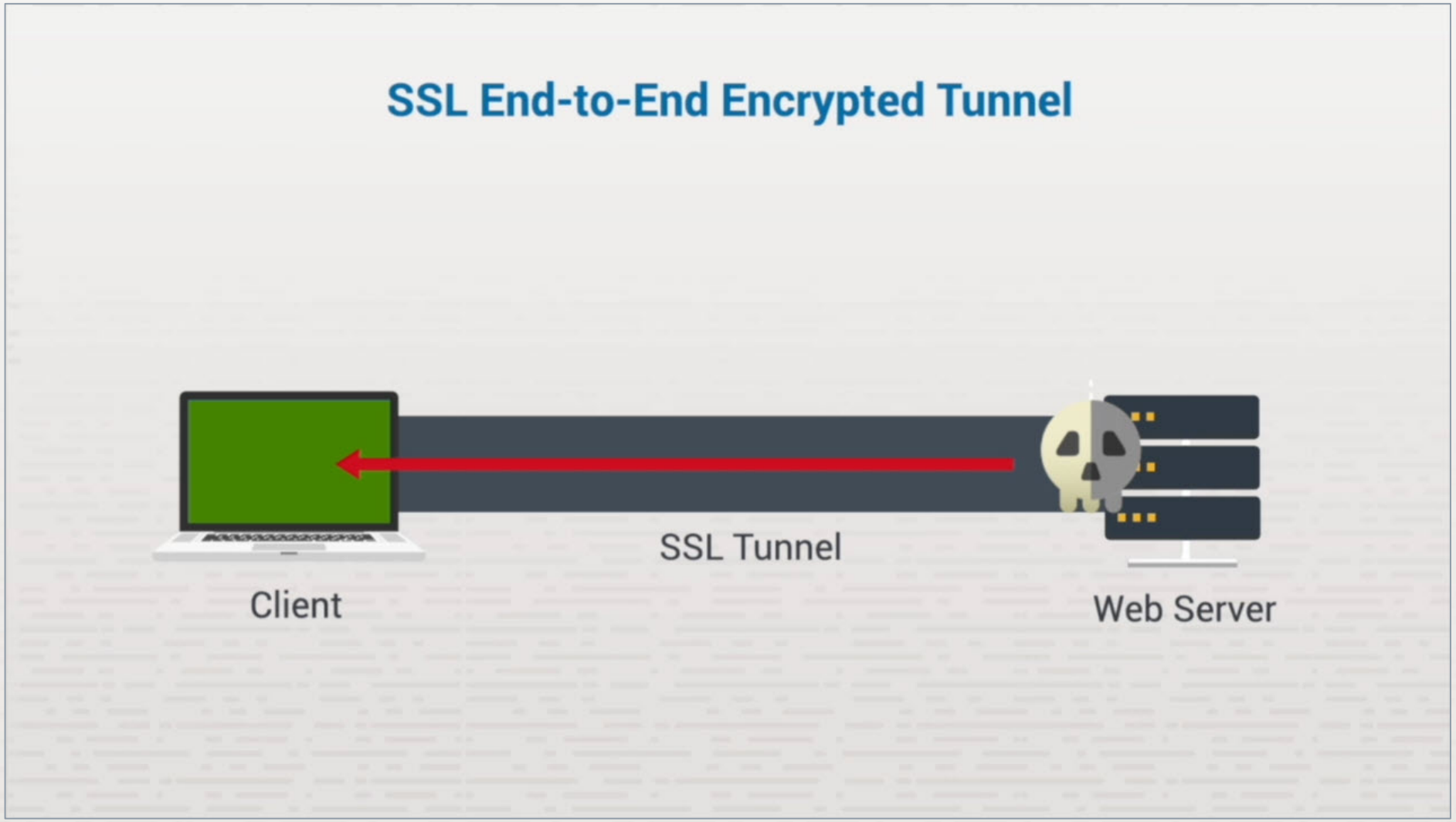
# Secure Protocols 2



# Secure Protocols 2

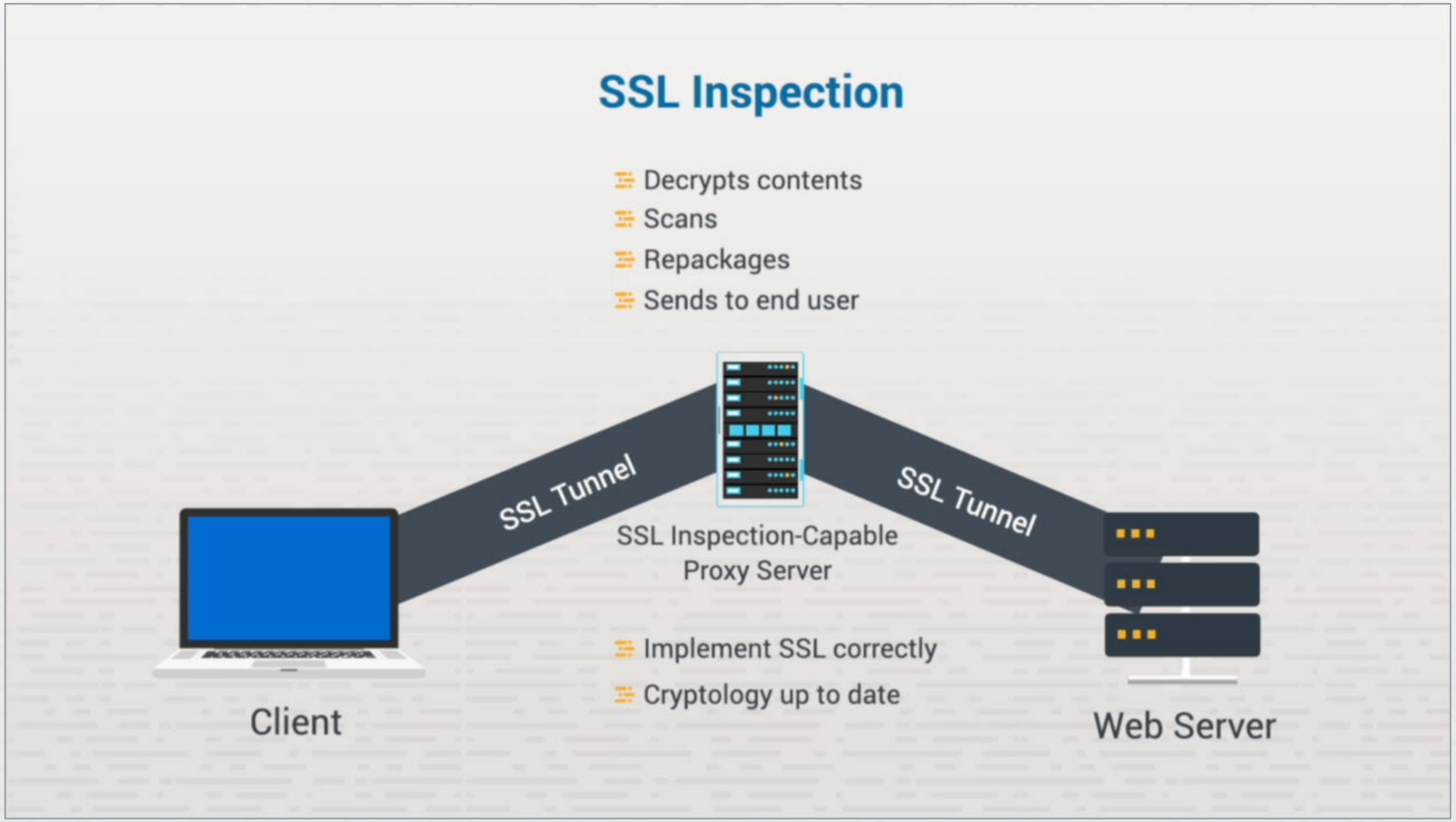


# Secure Protocols 2



# Secure Protocols 2

# Secure Protocols 2

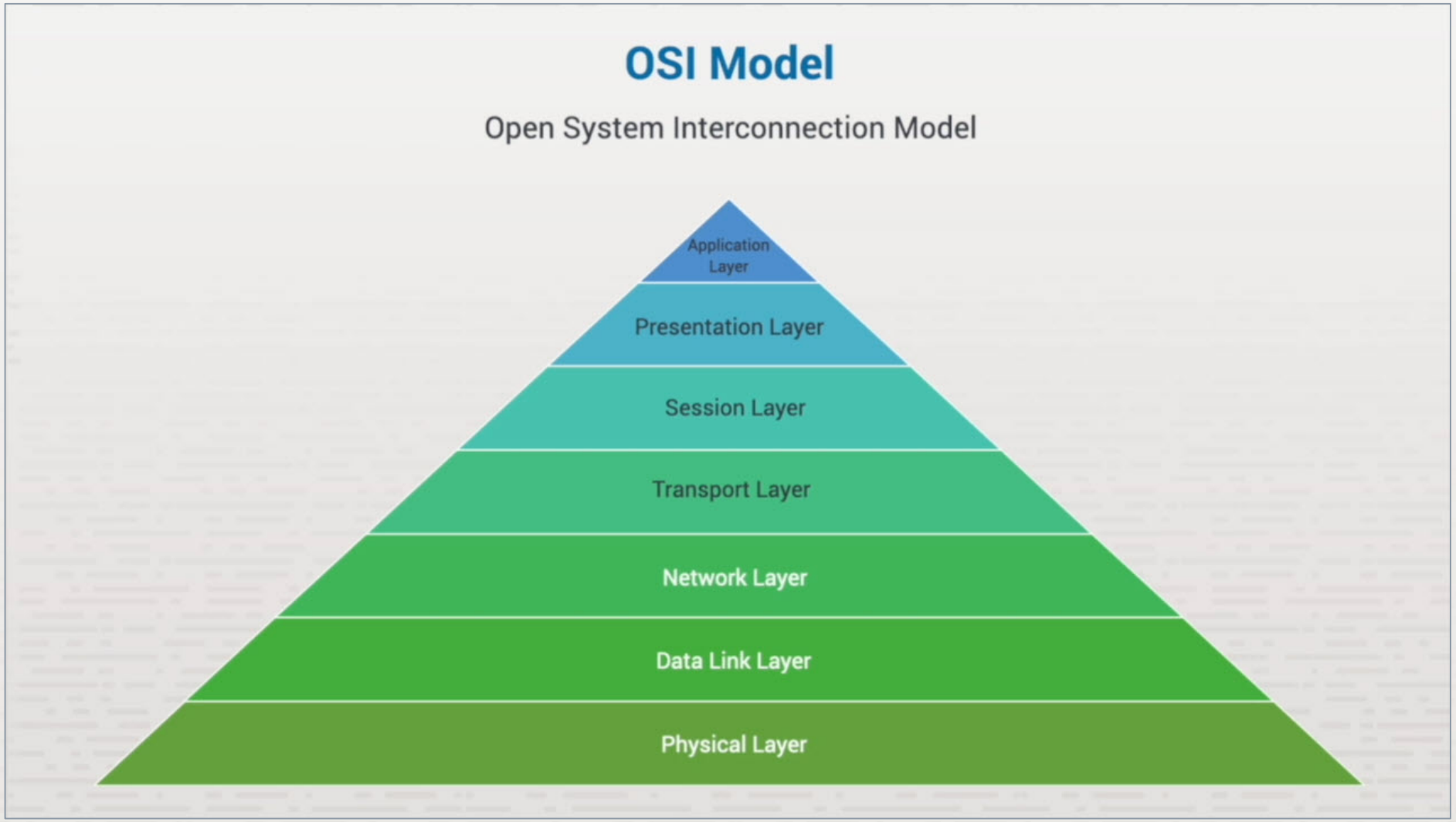


# Summary

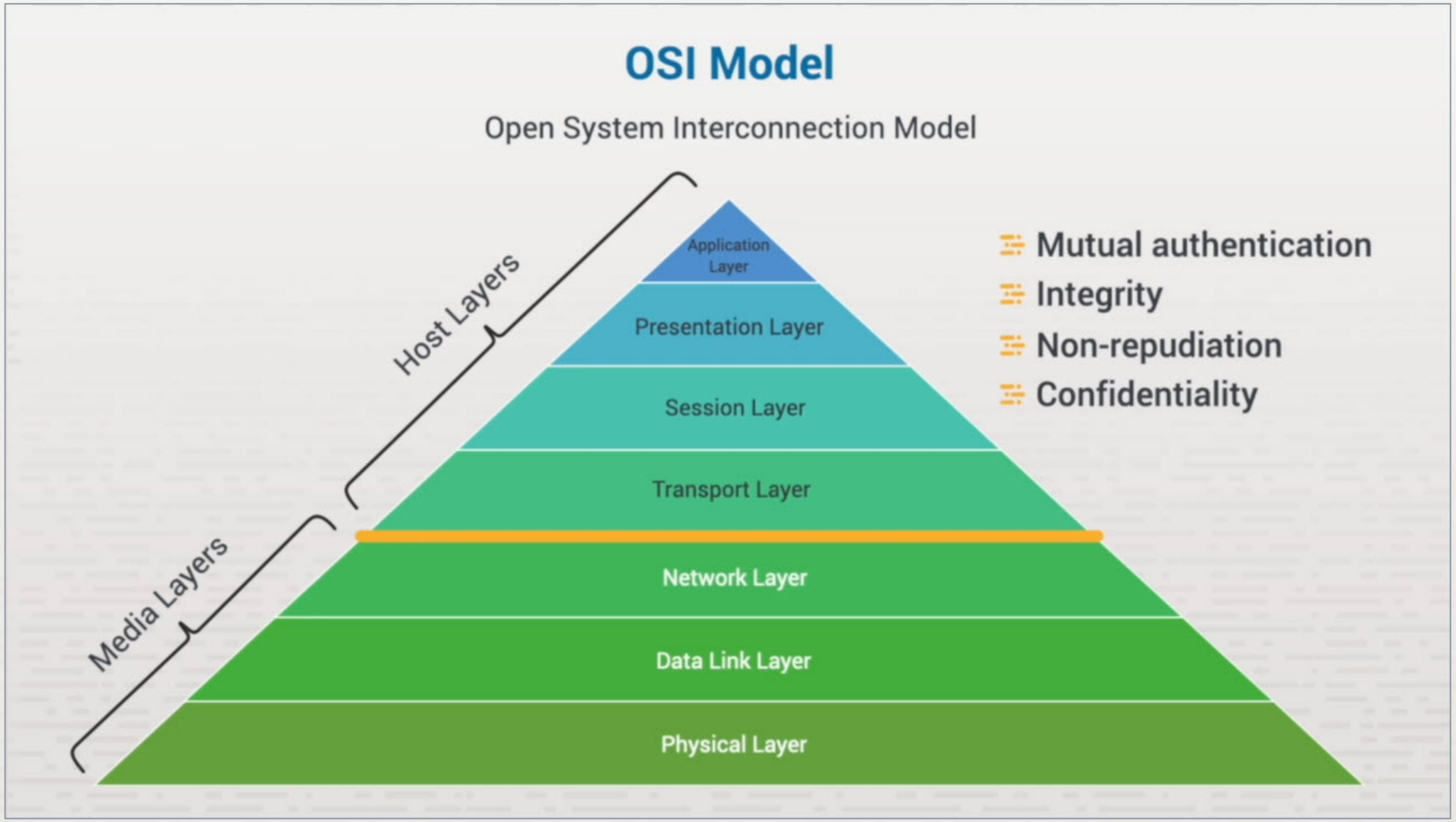
- HTTPS with SSL/TLS
- How HTTPS works
- SSL/TLS security issues

# IPsec

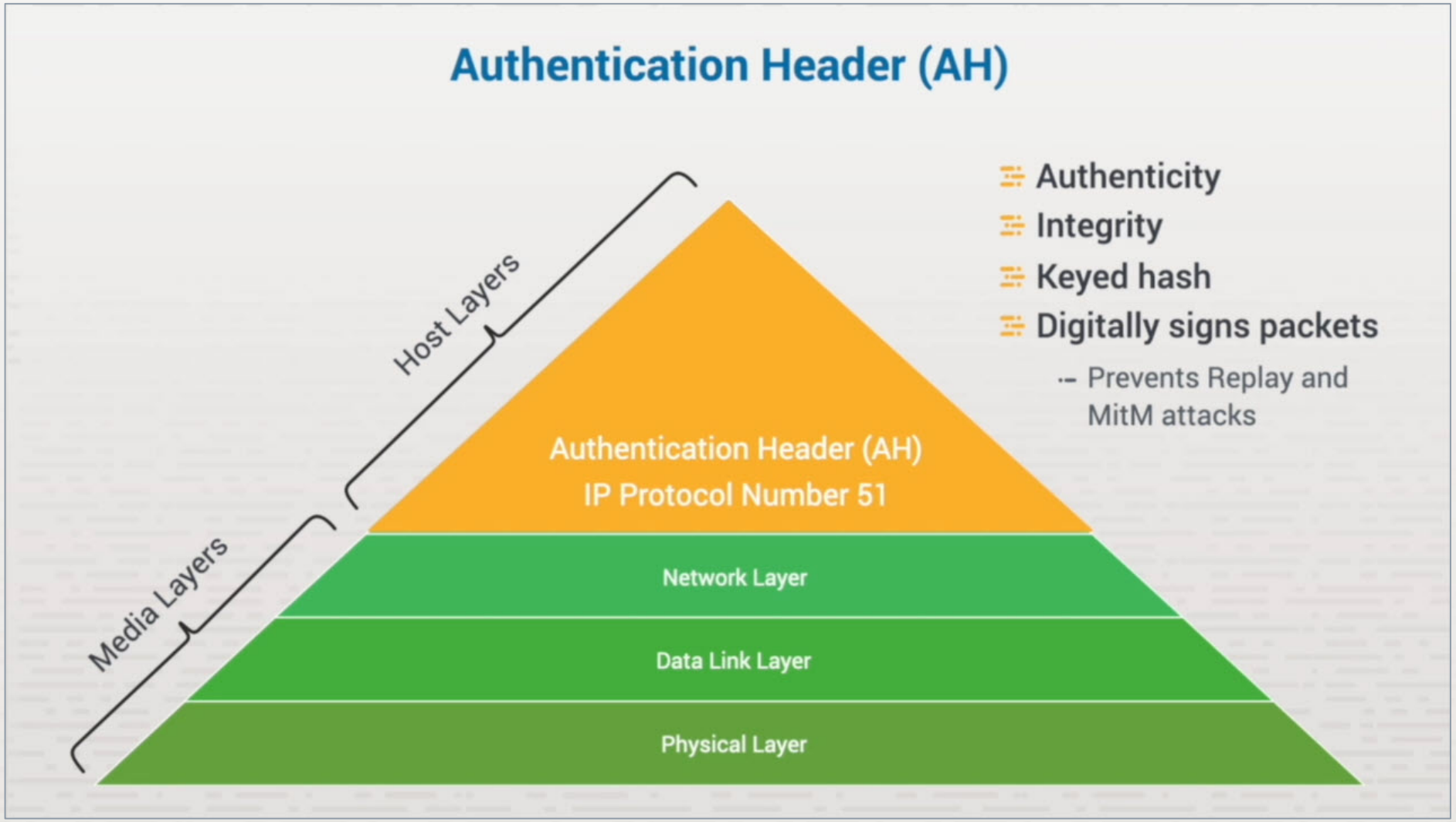
# IPsec



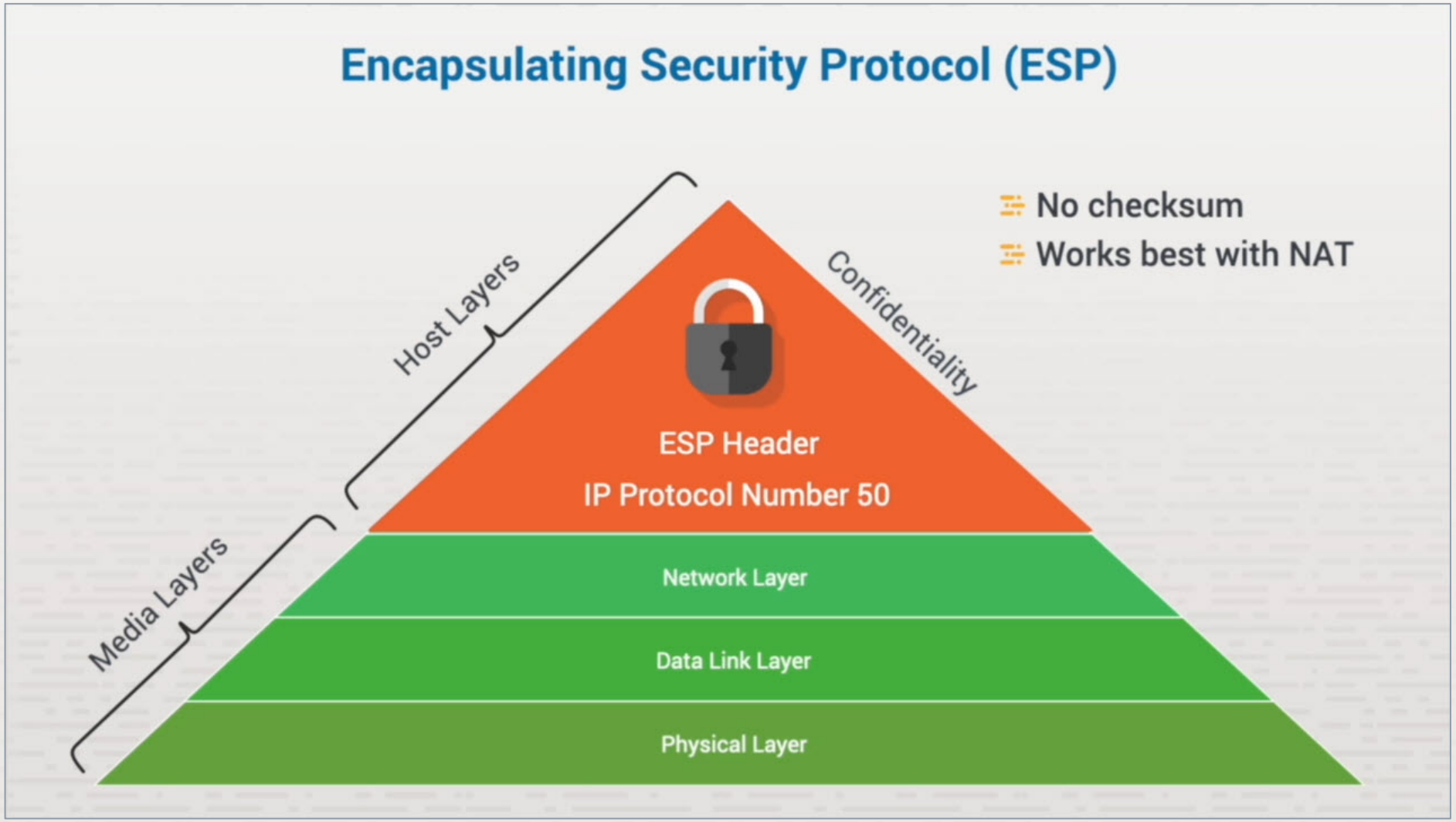
# IPsec



# IPsec


Authentication Header (AH)
Authenticity
Integrity
Keyed hash
Digitally signs packets
Prevents Replay and MitM attacks
Host Layers
Media Layers
Authentication Header (AH)
IP Protocol Number 51
Network Layer
Data Link Layer
Physical Layer

# IPsec

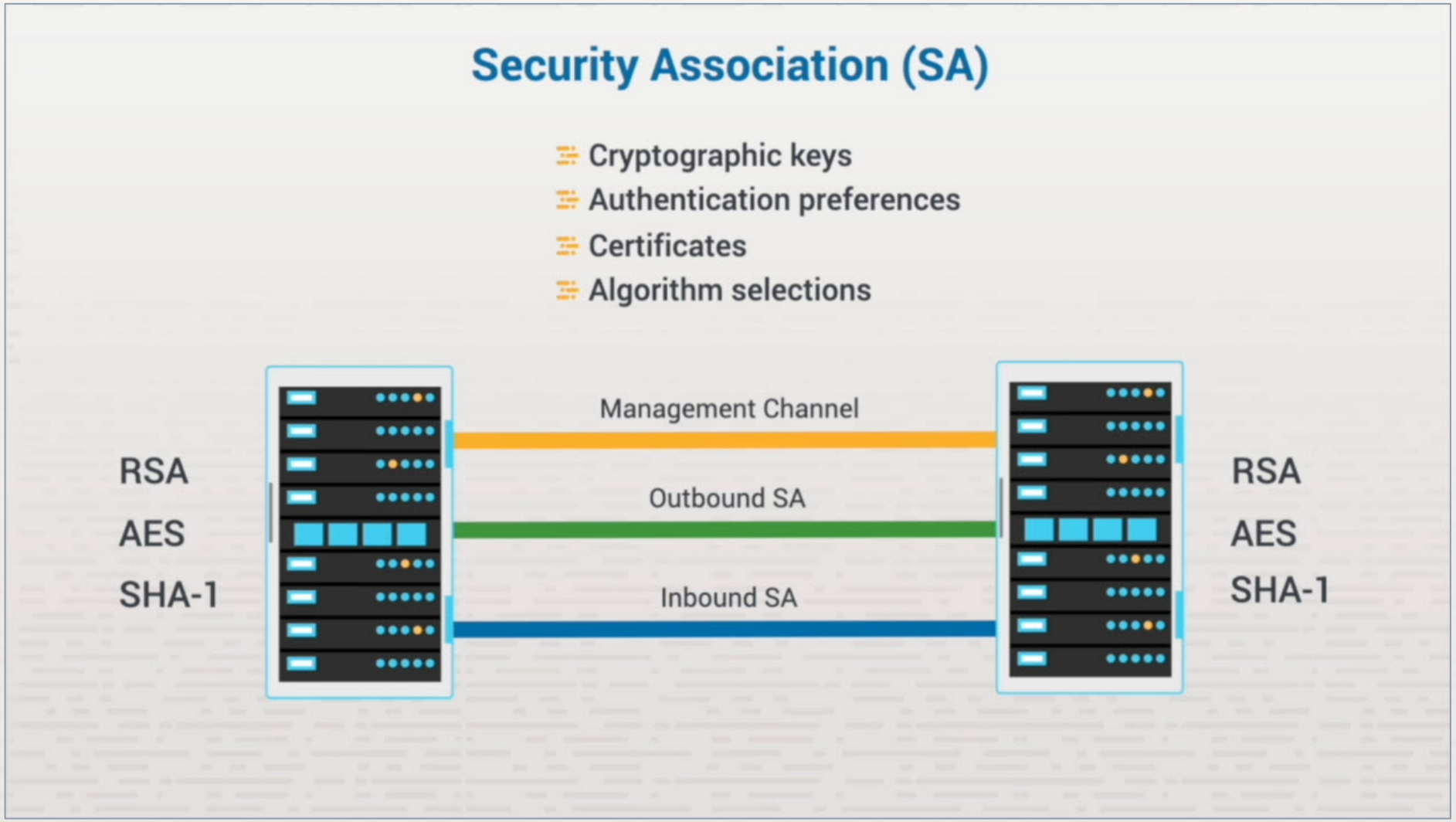

Encapsulating Security Protocol (ESP)
No checksum
Works best with NAT
Host Layers
Media Layers
Confidentiality
ESP Header
IP Protocol Number 50
Network Layer
Data Link Layer
Physical Layer

# IPsec

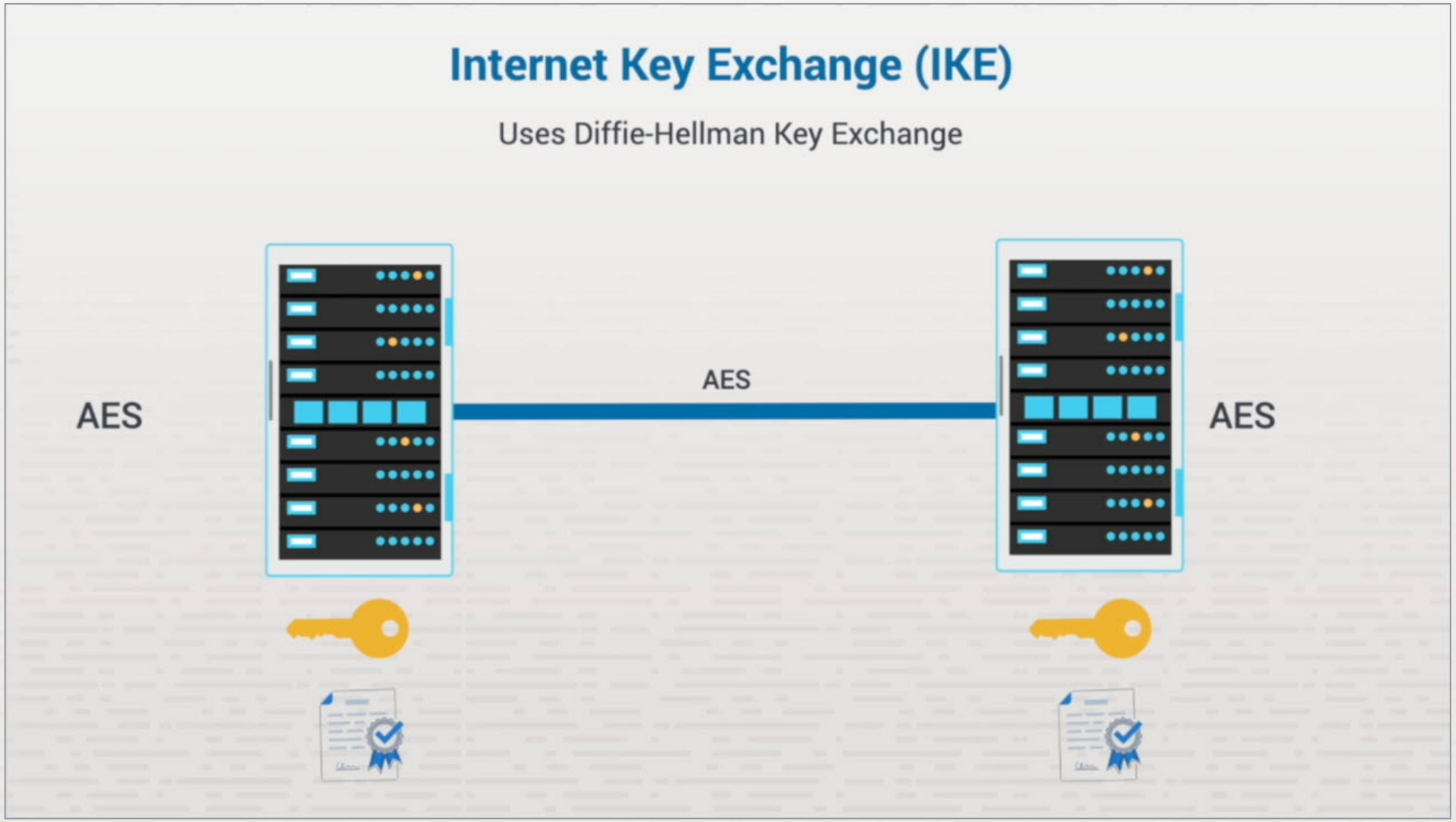
## Network Address Translation (NAT)

- **Can have problems with IPsec**
  - Tampers with packets to change source destination
- **Uses NAT Transversal (NAT-T)**
  - Allows IPsec to function properly
    - Encapsulates ESP packets inside a UDP packet
    - Uses UDP port 4500

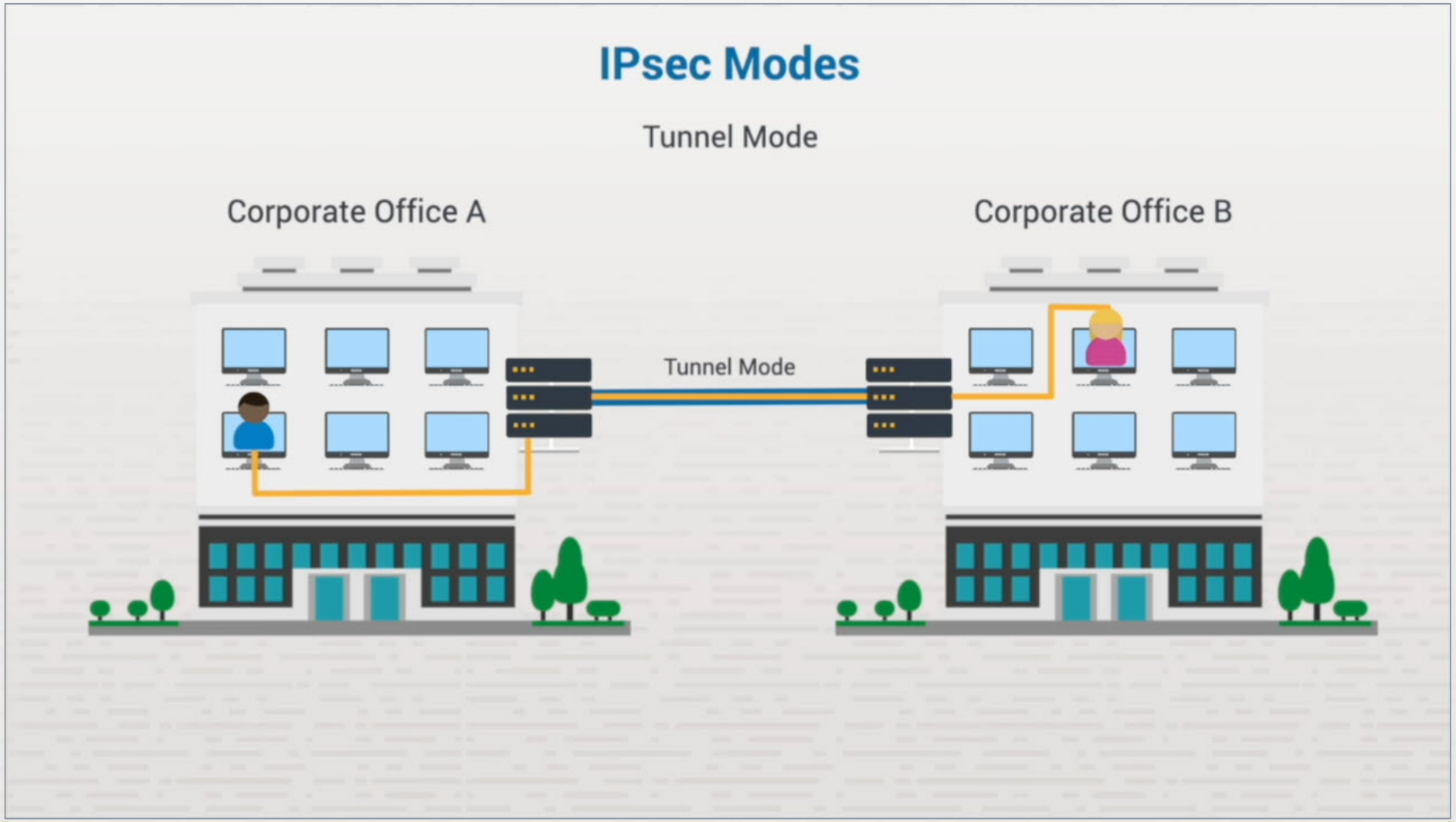
# IPsec



# IPsec


Internet Key Exchange (IKE)
Uses Diffie-Hellman Key Exchange
AES
AES
AES

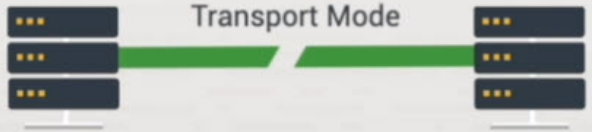
# IPsec


IPsec Modes
Tunnel Mode
Corporate Office A
Corporate Office B
Tunnel Mode

# IPsec

## IPsec Modes

Transport Mode



# Summary

- Authentication Headers (AH)
- Encapsulating Security Payload (ESP) Security Association (SA)
- Internet Key Exchange (IKE)
- IPsec modes

# In-Class Practice

Do the following labs:

- 10.1.5 Allow SSL Connections

# Class Discussion

- How does SSL verify authentication credentials?
- What protocol is the successor to SSL 3.0?
- How can you tell that a session with a web server is using SSL?
- What is the difference between HTTPS and S-HTTP?
- What does it mean when HTTPS is stateful?
- What is the difference between IPsec tunnel mode and transport mode?

10.2 SECURING DATA AND APPLICATIONS

# Data Loss Prevention

TESTOUT SECURITY PRO

# Section Skill Overview

- Understand DLP, masking, encryption, tokenization, and rights management.

# Key Terms

- Data loss prevention (DLP)
- Network DLP
- Endpoint DLP
- File-level DLP
- Cloud DLP
- Masking
- Encryption
- Tokenization
- Rights management

# Key Definitions

- **Data loss prevention (DLP):** A system that attempts to detect and stop breaches of sensitive data within an organization.
- **Network DLP:** A software or hardware solution that is typically installed near the network perimeter that analyzes network traffic in an attempt to detect transmission of sensitive data in violation of an organization's security policies.
- **Endpoint DLP:** DLP Software that runs on end-user workstations and servers.
- **File-level DLP:** DLP software that is used to identify sensitive files in a file system and then to embed the organization's security policy within the file so that it travels with the a moved or copied file.

# Key Definitions

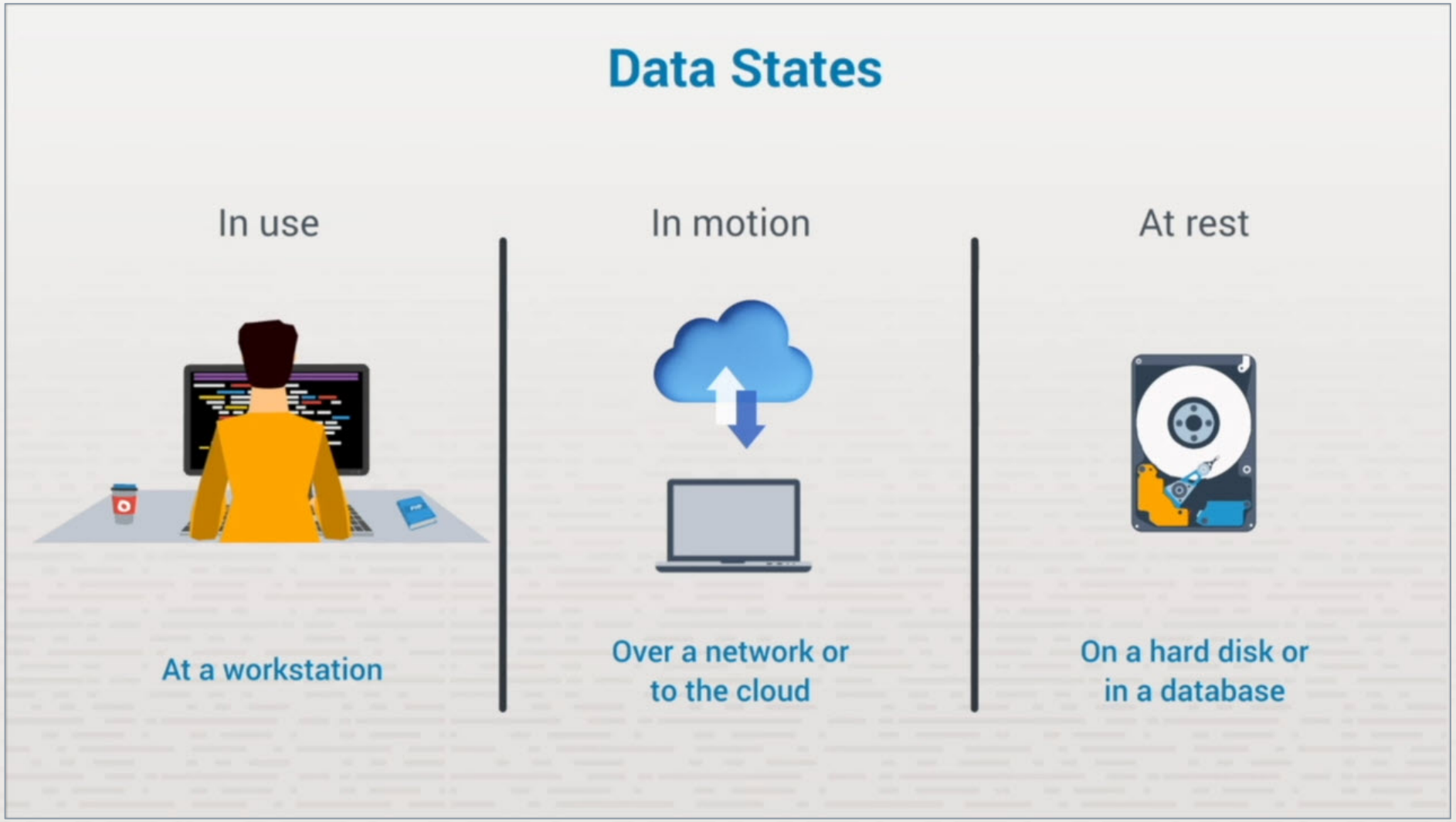
- **Cloud DLP:** A software solution that analyzes traffic to and from cloud systems in an attempt to detect sensitive data that is being transmitted in violation of an organization's security policies.
- **Masking:** The process of replacing sensitive data with realistic fictional data.
- **Encryption:** The process of changing plain text through an algorithm into unreadable ciphertext.
- **Tokenization:** The process of replacing original data with a randomly generated alphanumeric character set called a token.
- **Rights management:** A system of data protection at the file level that uses various forms of permissions, rules, and security policies.

# Data Loss Prevention

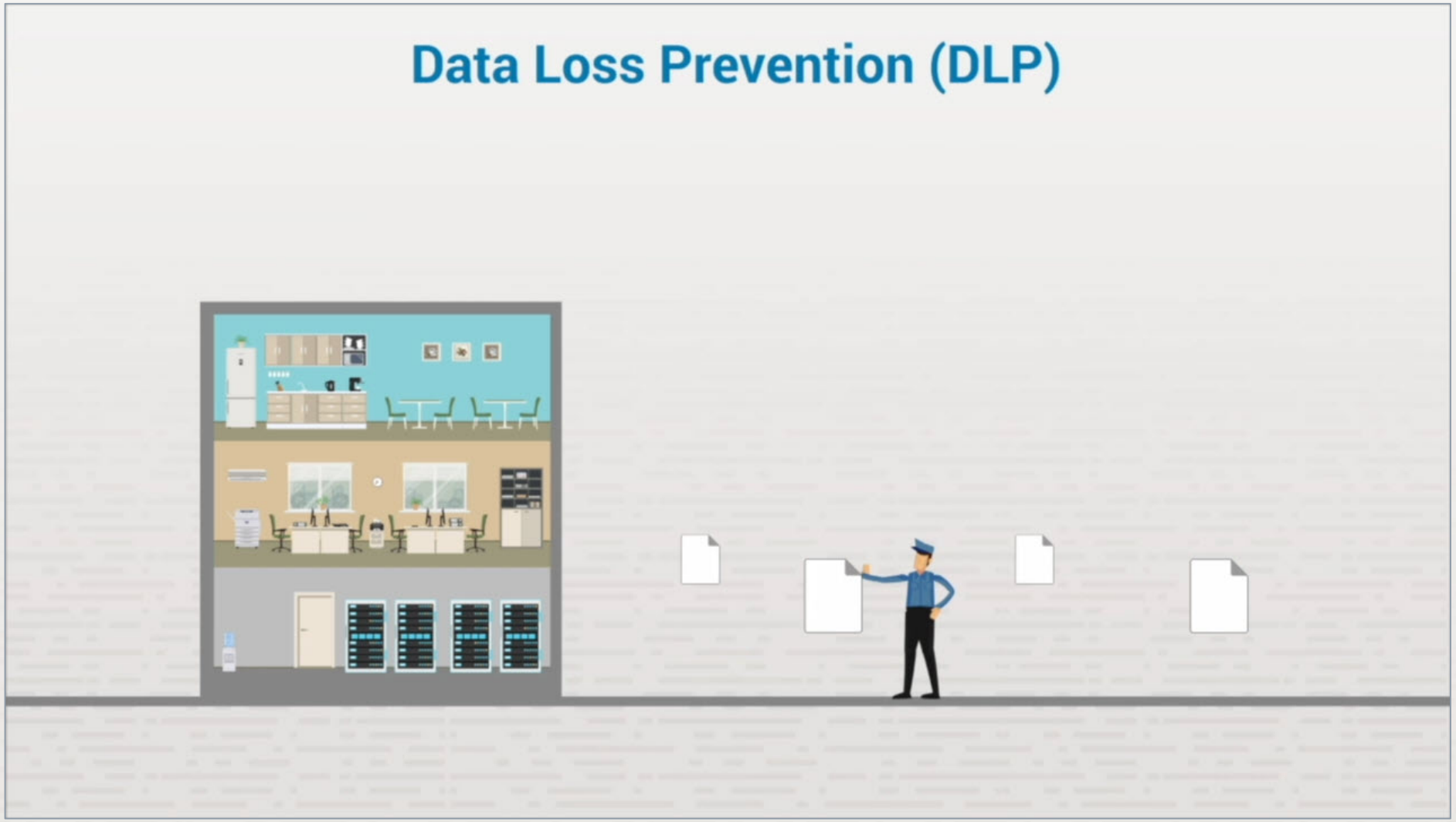
# Data Protection

- Data loss prevention (DLP)
- Masking It Encryption
- Tokenization
- Rights management

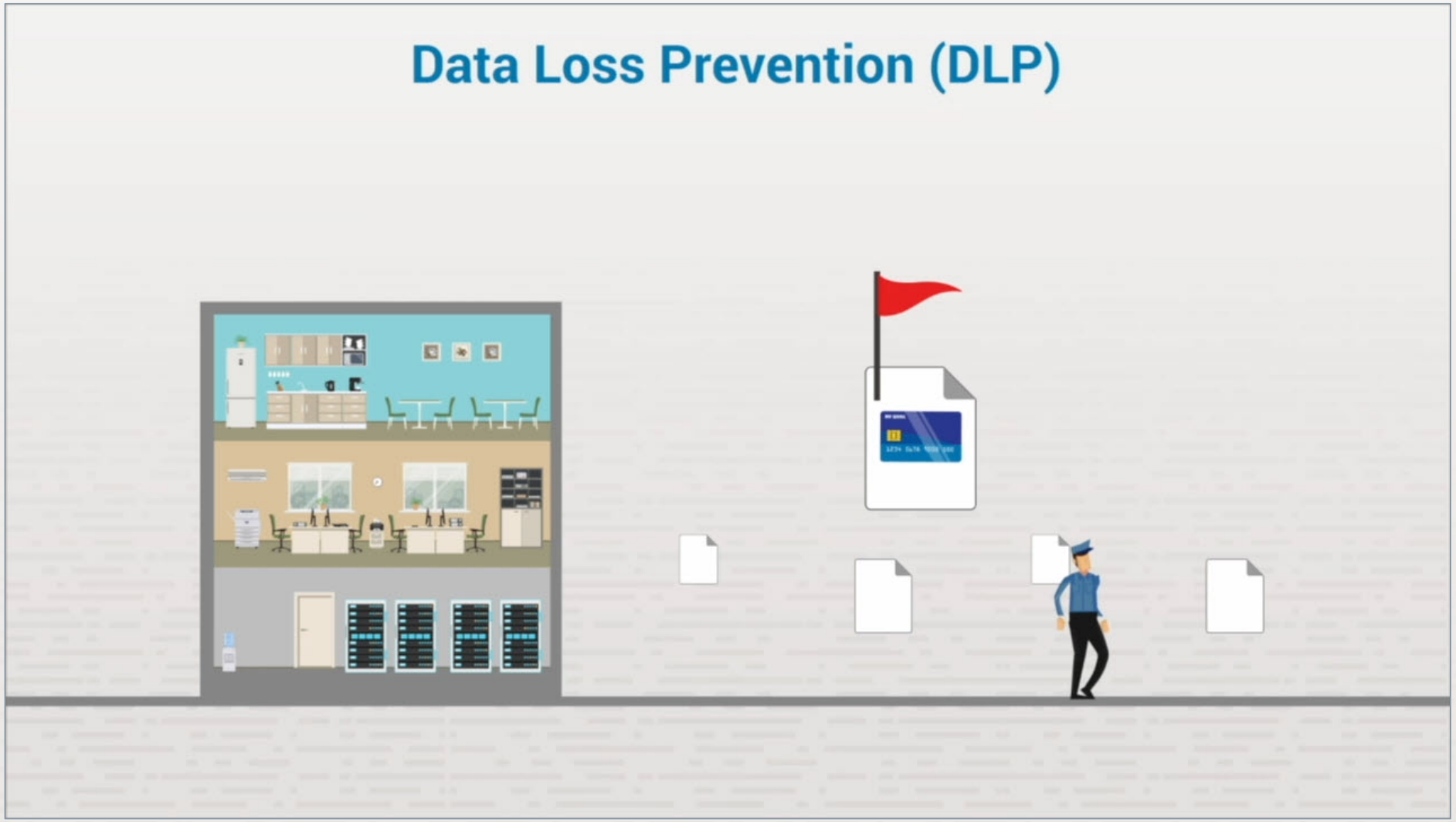
# Data Loss Prevention



# Data Loss Prevention



# Data Loss Prevention



# Data Loss Prevention

## Masking

- Replaces sensitive data with realistic fictional data
- Dynamic data masking
- Static data masking

# Data Loss Prevention

## Dynamic Data Masking

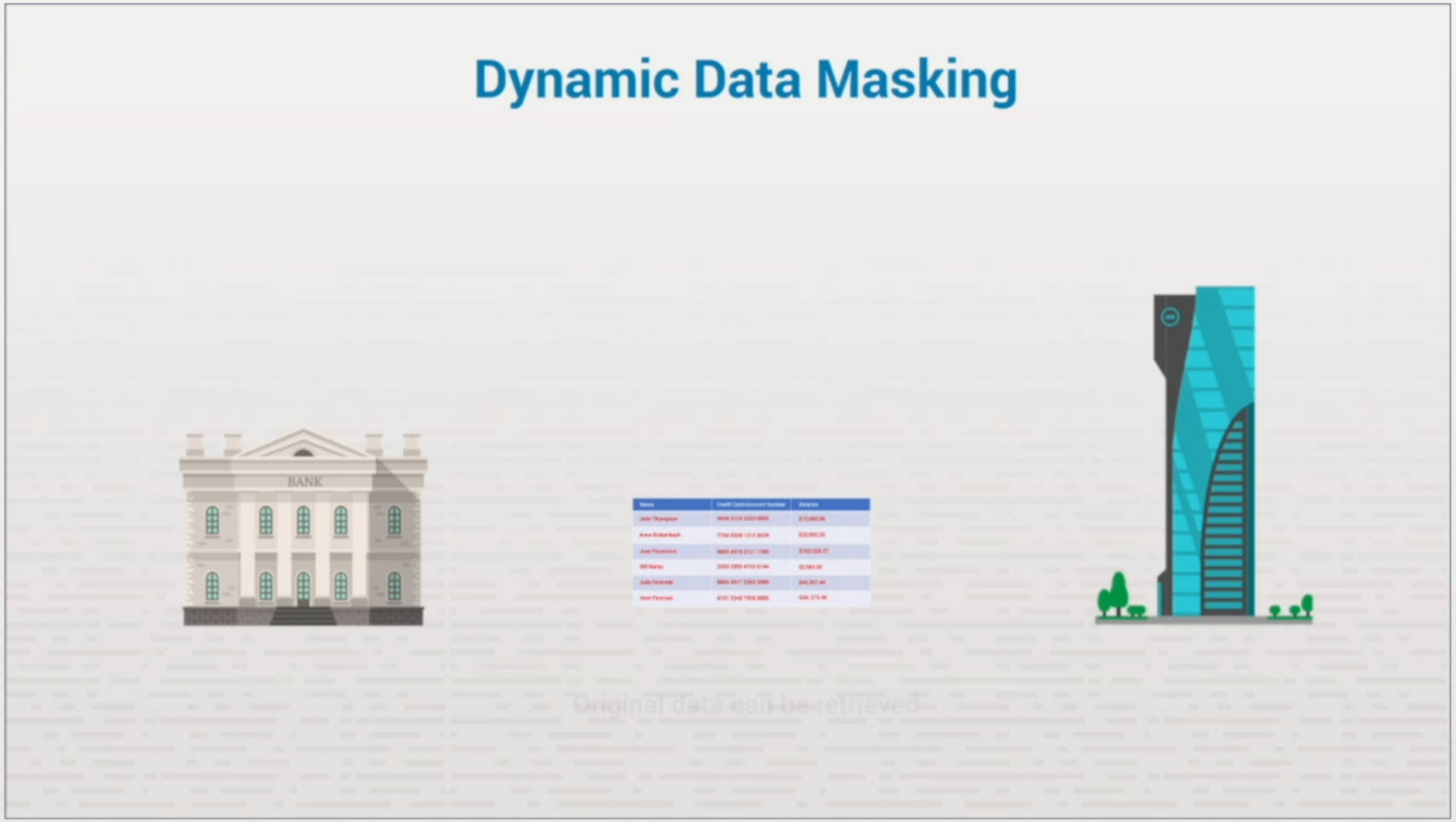
| Name | Credit Card/Account Number | Balance |
|---|---|---|
| James Johnson | 3421 6578 9812 6327 | $12,583.56 |
| Cindy Franklin | 3579 5128 6493 7418 | $25,092.03 |
| Greg Brown | 6593 2587 7536 6248 | $103,528.27 |
| Joe Hansen | 9865 2457 8653 4677 | $2,985.82 |
| Emily Stevens | 4378 2738 4950 2245 | $45,367.44 |
| Paul Allen | 5644 9876 4082 2284 | $38, 275.96 |

# Data Loss Prevention

## Dynamic Data Masking

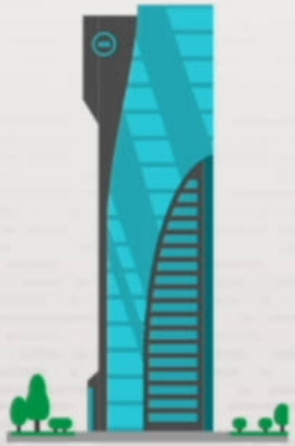
| Name | Credit Card/Account Number | Balance |
|---|---|---|
| John Thompson | 4558 2124 2322 0892 | $12,583.56 |
| Anne Rickenbach | 7754 5528 1212 5629 | $25,092.03 |
| Jose Fiorentino | 6895 4515 2121 1785 | $103,528.27 |
| Bill Bailey | 3359 2895 4743 6144 | $2,985.82 |
| Juila Kennedy | 8855 4517 2562 3989 | $45,367.44 |
| Kent Petersen | 4151 2548 7996 5885 | $38, 275.96 |

# Data Loss Prevention



# Data Loss Prevention

## Dynamic Data Masking

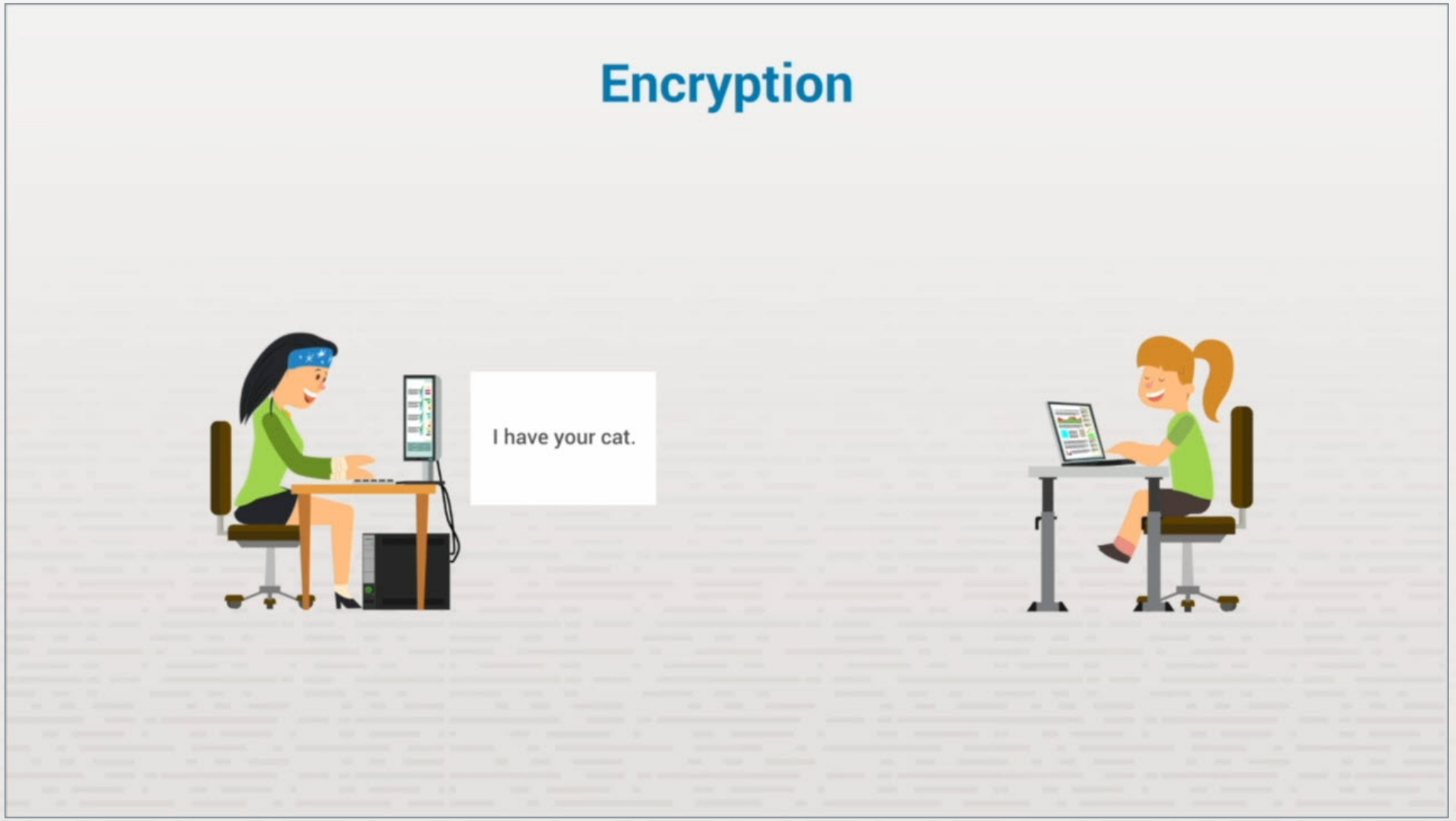
Original data can be retrieved

# Data Loss Prevention

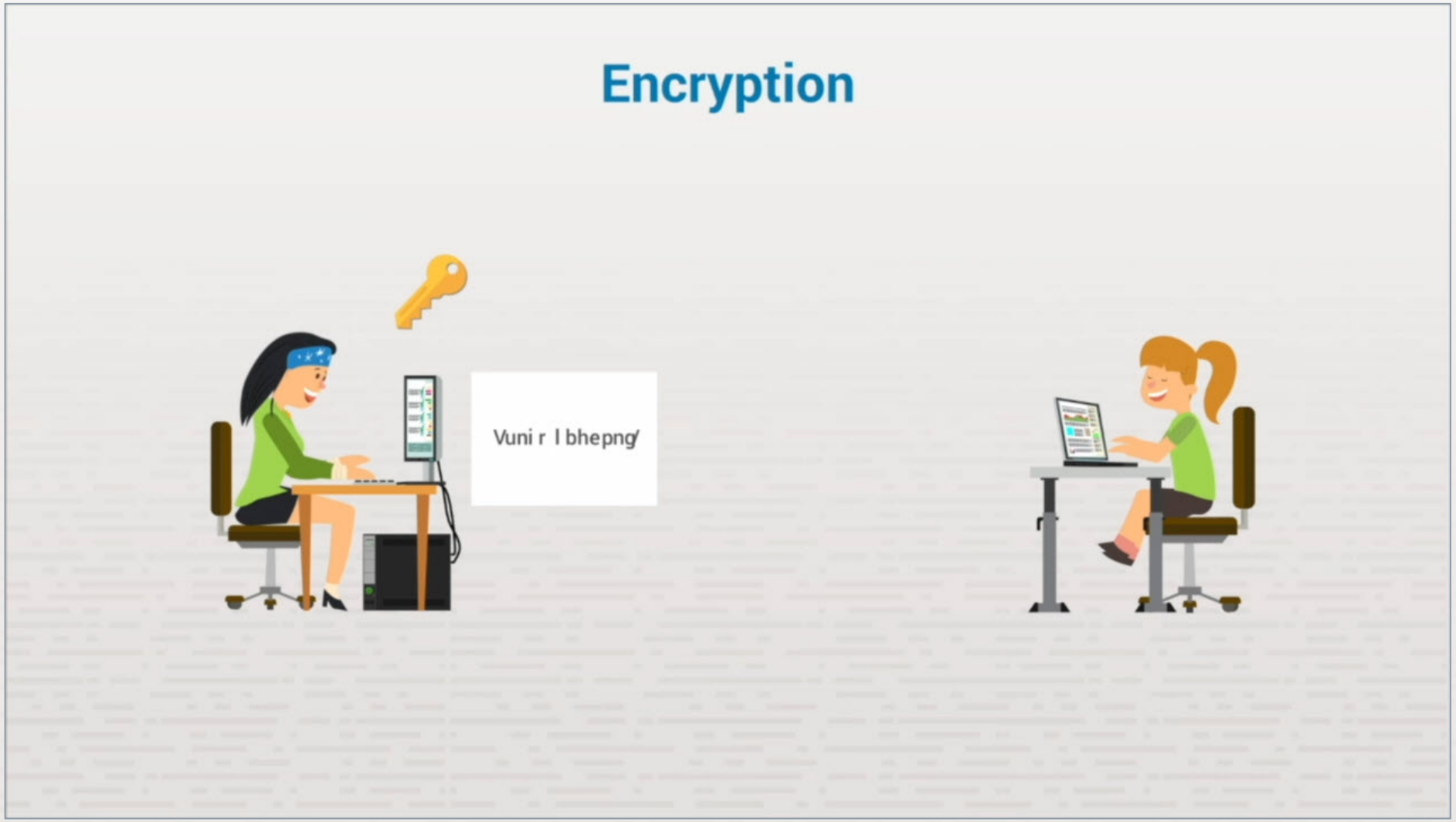
## Static Data Masking

- Helpful for data at rest
- Can specify field or columns
- Original data is irretrievable

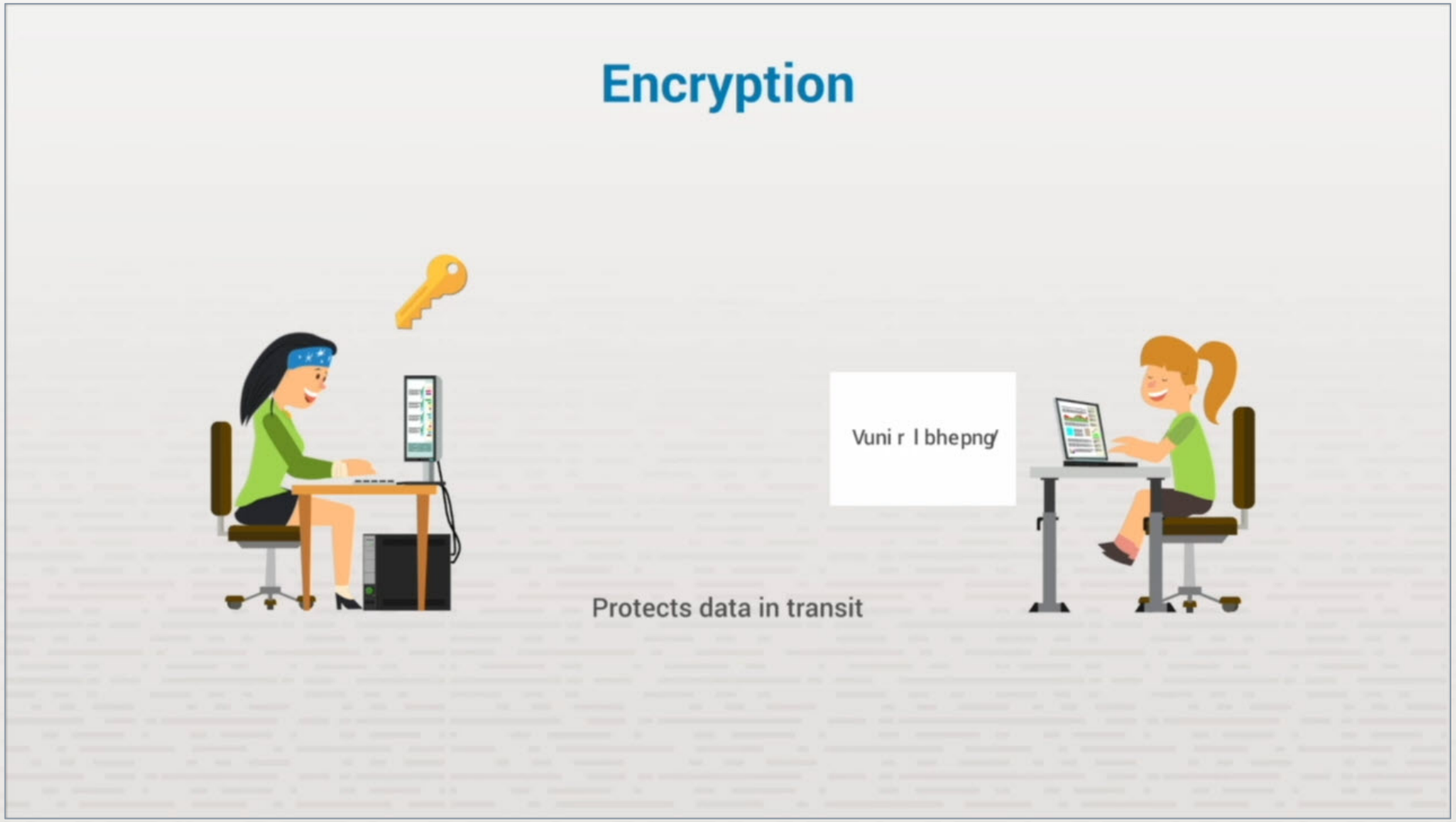
# Data Loss Prevention



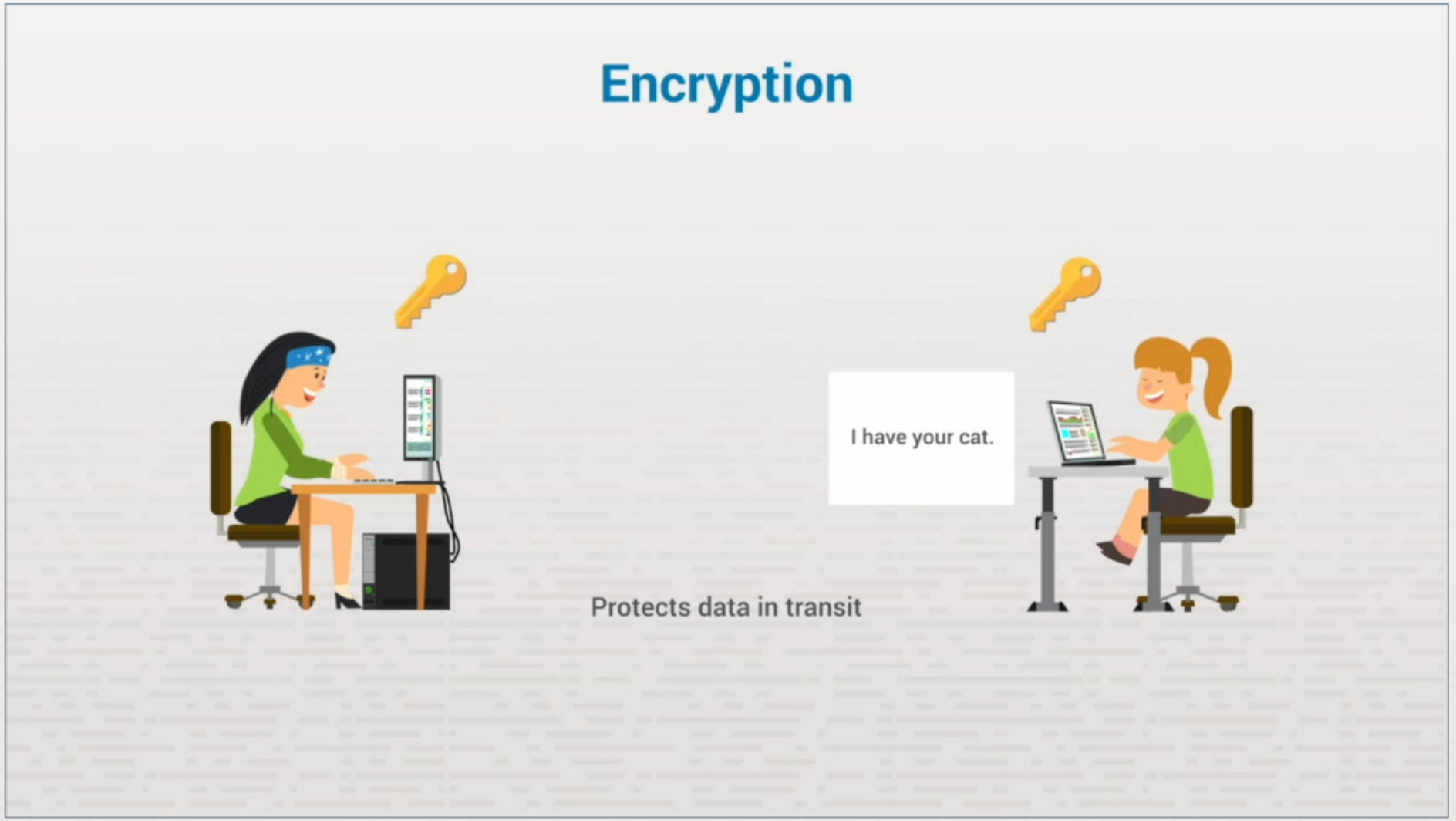
# Data Loss Prevention



# Data Loss Prevention



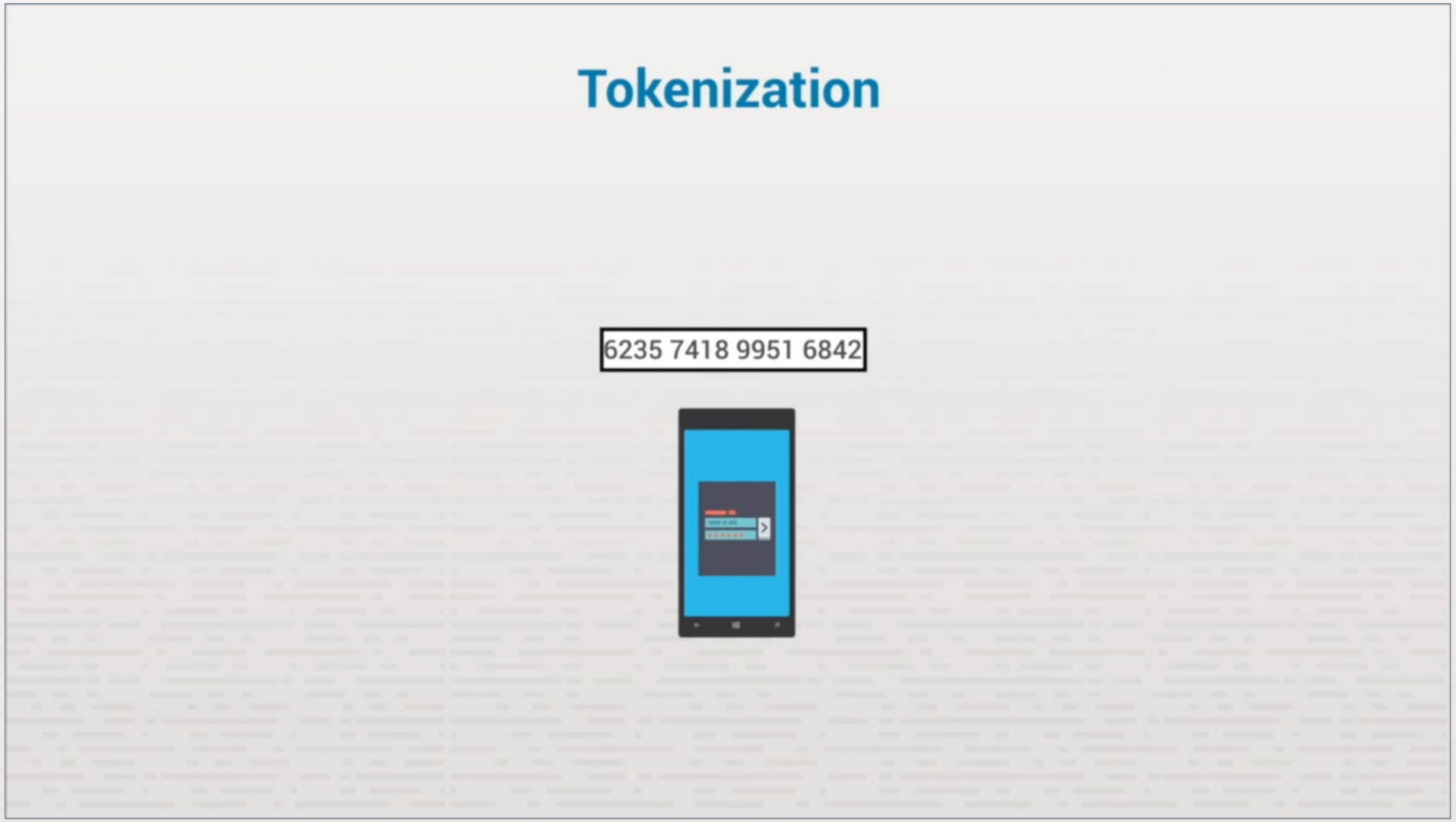
# Data Loss Prevention



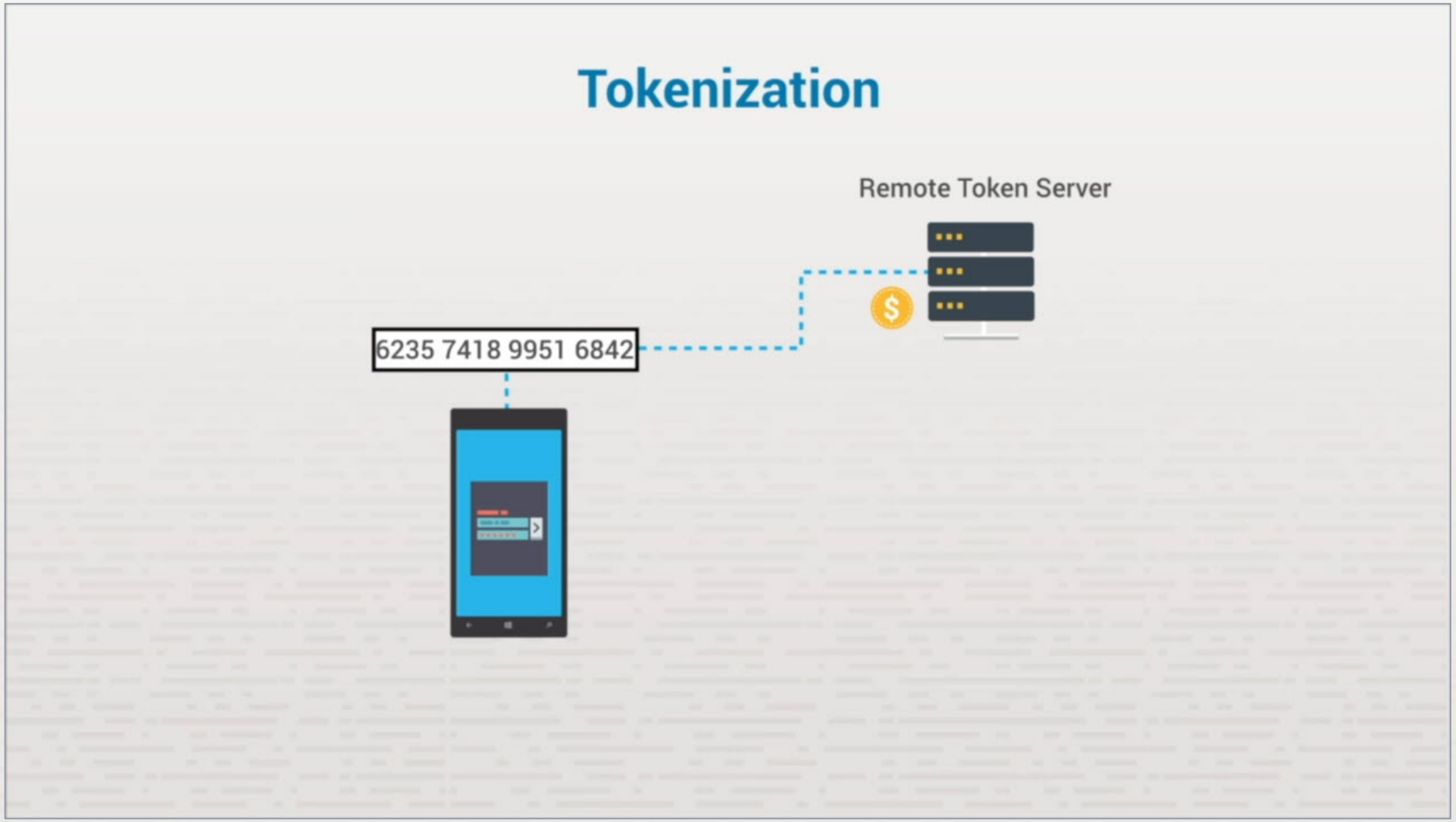
# Tokenization

- Alphanumeric character set
- Token server protected
- Token presented to disclose data

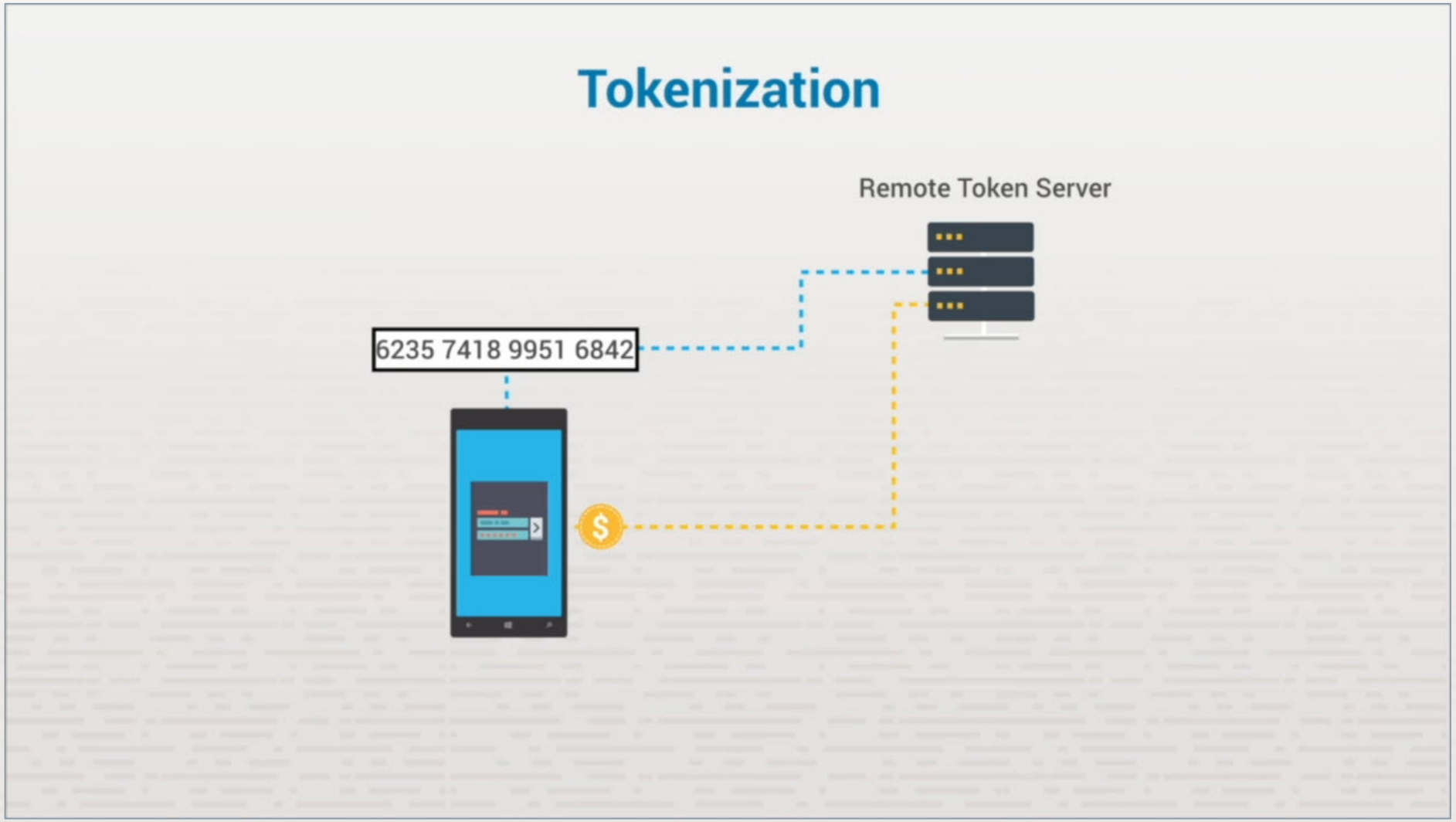
# Data Loss Prevention



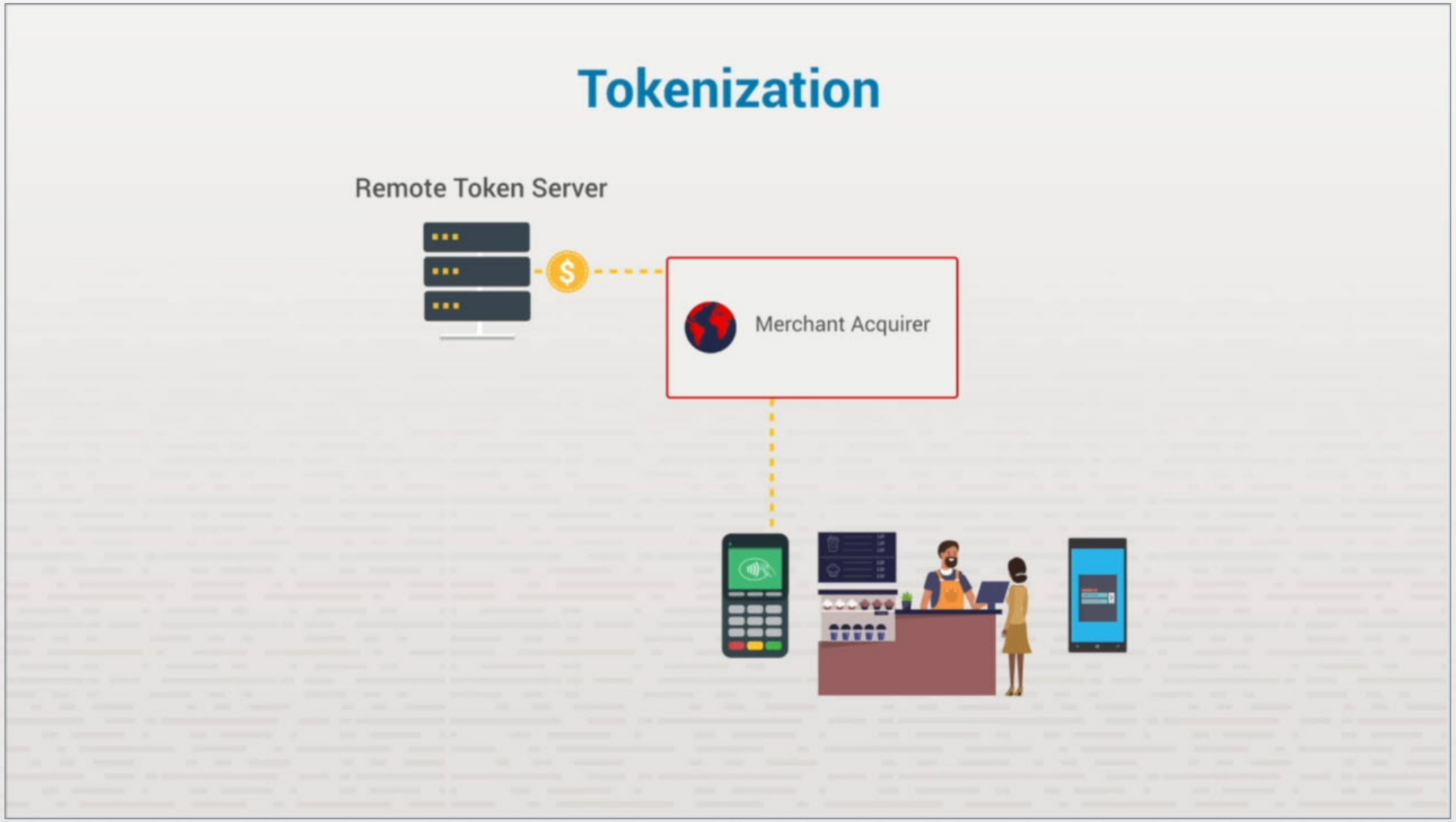
# Data Loss Prevention



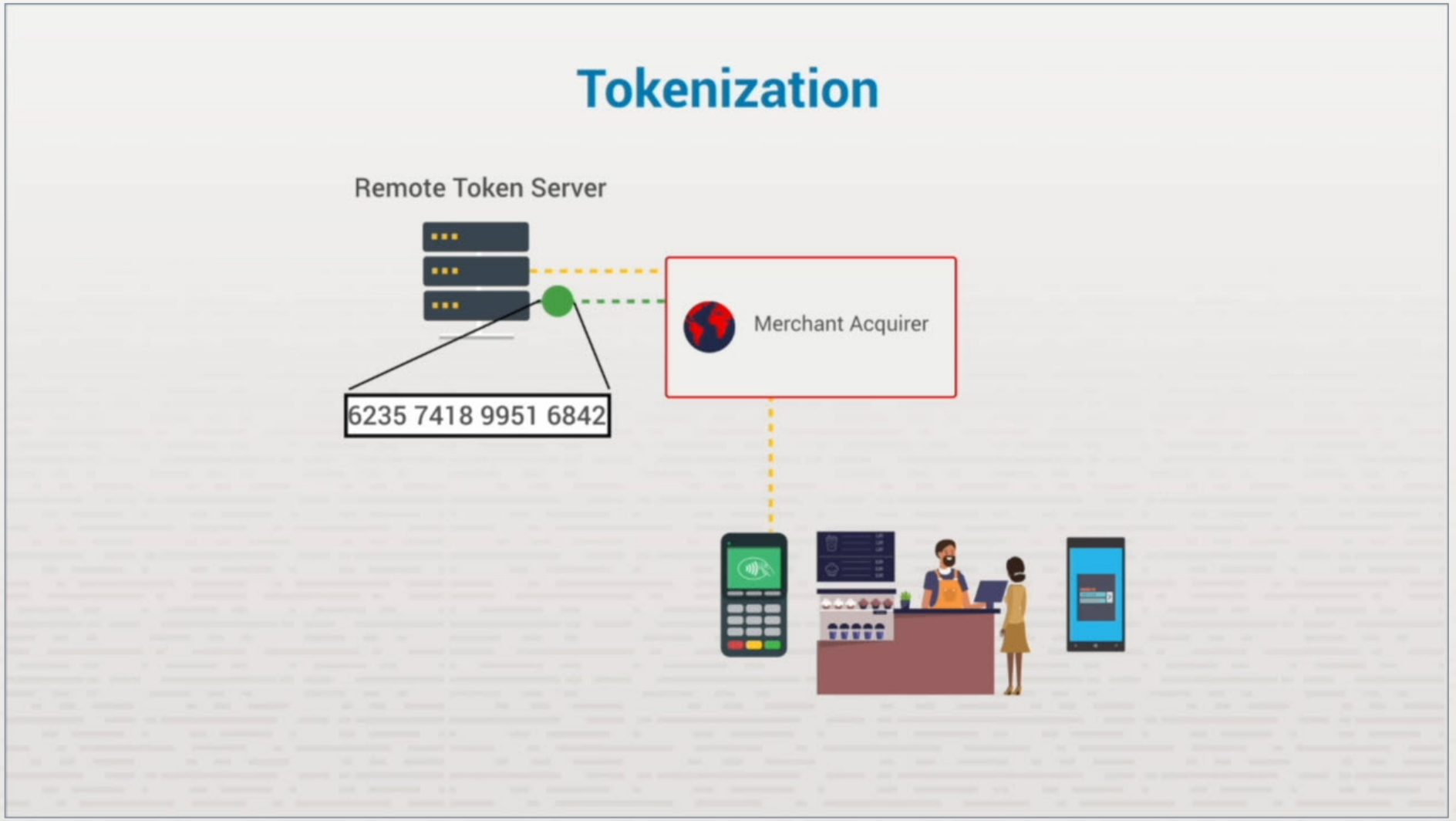
# Data Loss Prevention



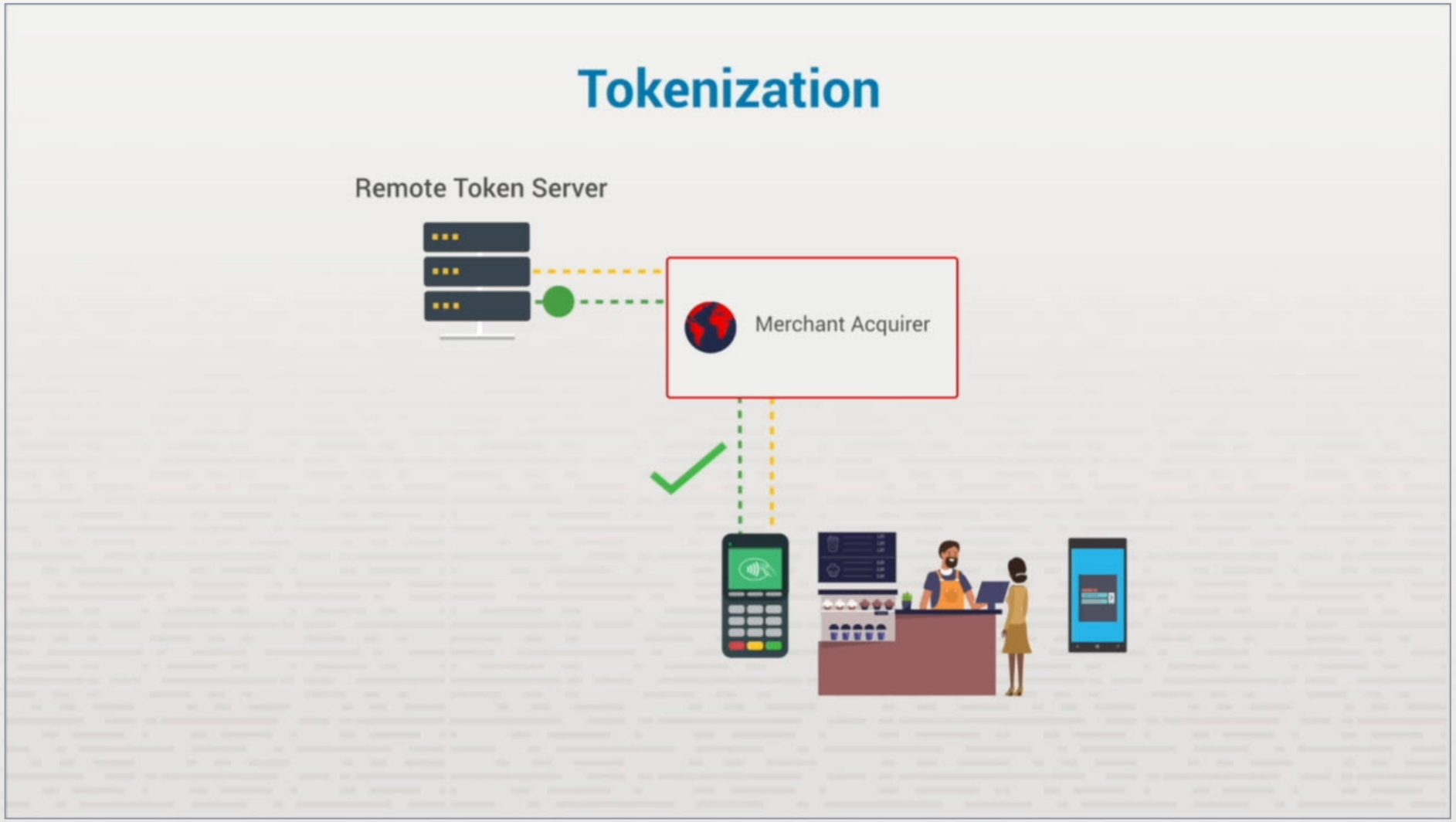
# Data Loss Prevention



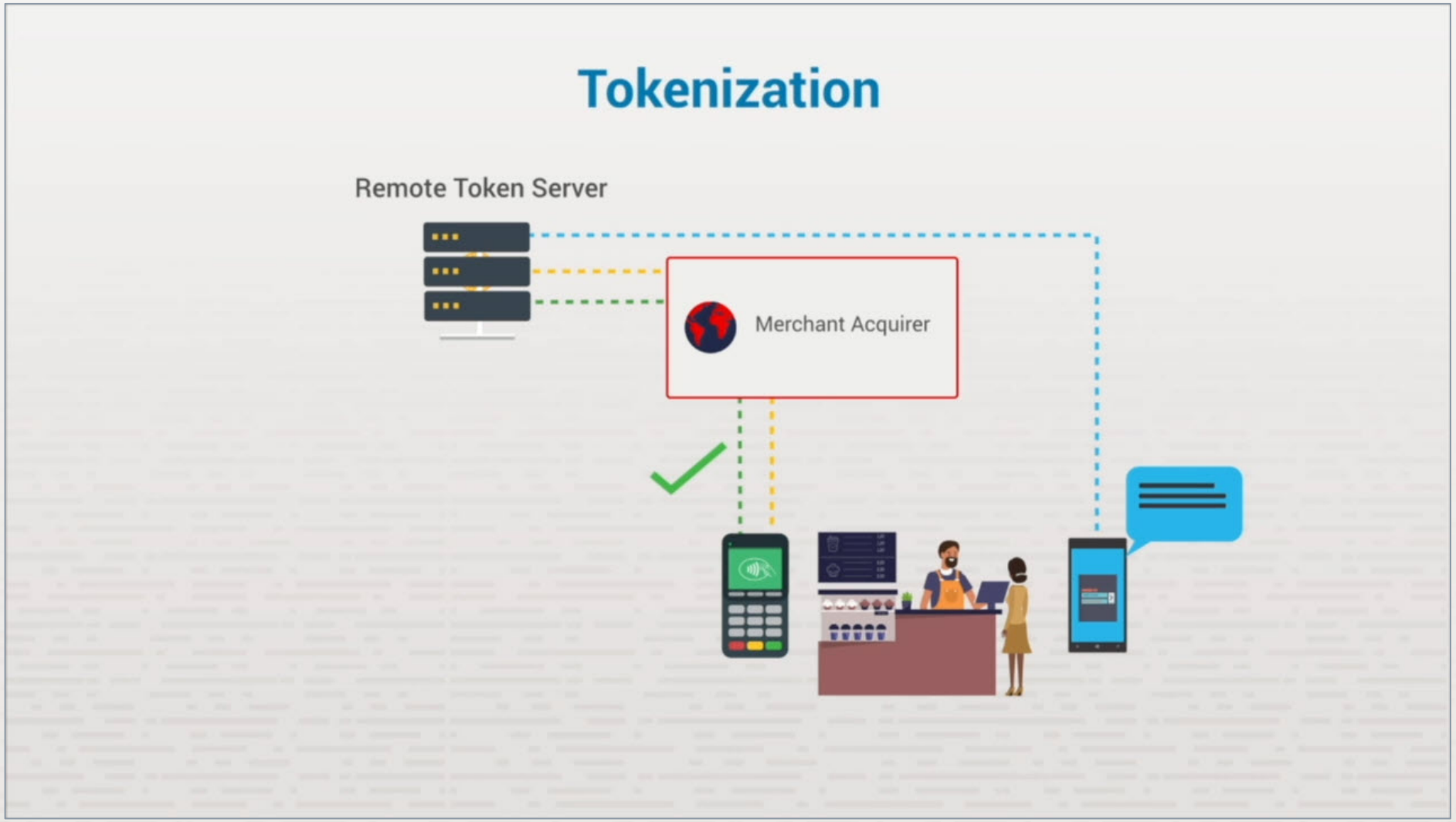
# Data Loss Prevention



# Data Loss Prevention



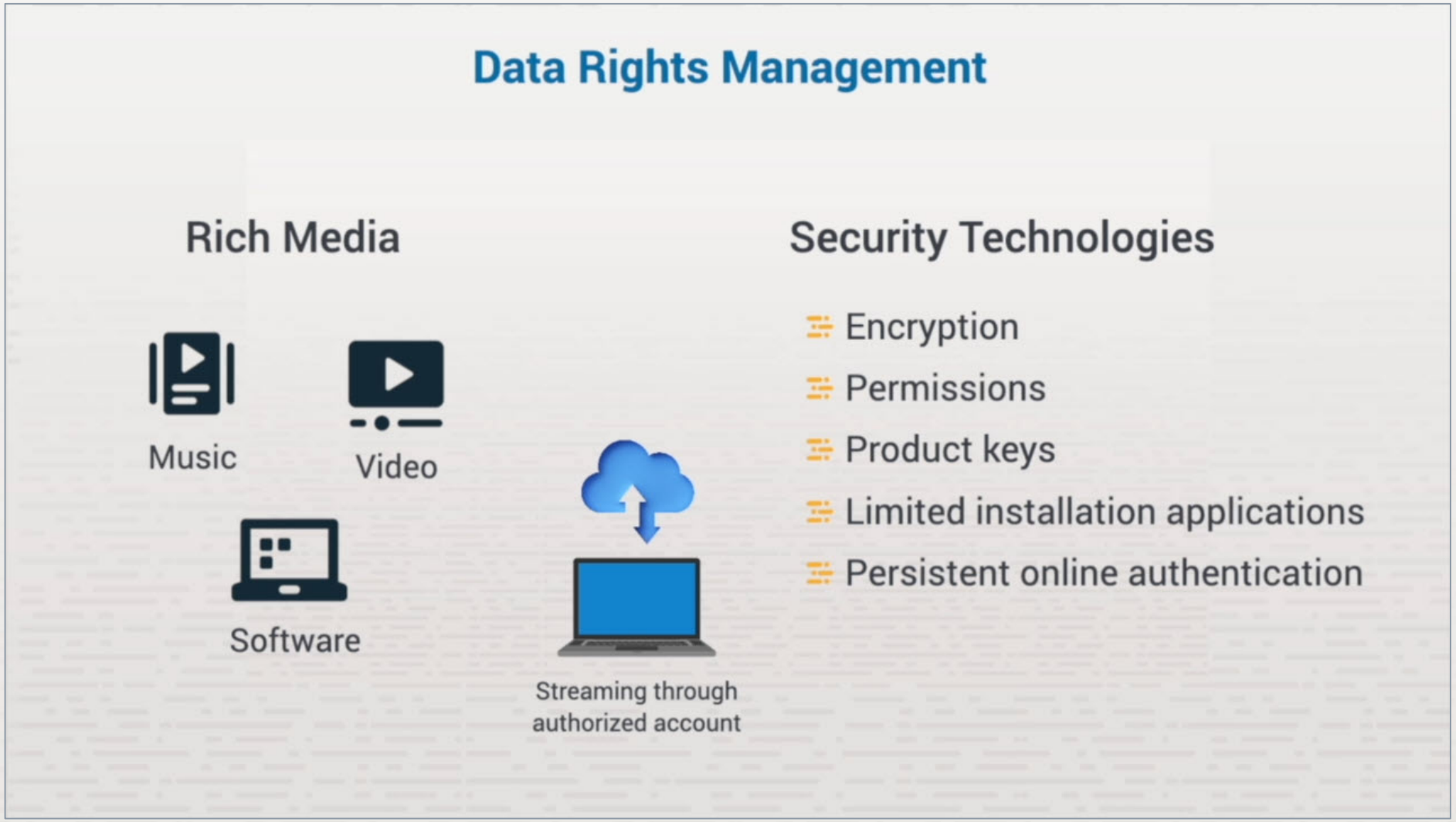
# Data Loss Prevention



# Rights Management

- File level protection
- Security travels with file
- Control access remotely
- Two categories
  - Digital rights management
  - Information rights management

# Data Loss Prevention



# IRM

- Business-to-business files
  - Documents
  - Emails
  - Spreadsheets
  - Financial Data
- Encryption
- Permissions

# Summary

- DLP
- Masking
- Encryption
- Tokenization
- Rights management

# Class Discussion

- What is the purpose of a DLP system?
- How can DLP be implemented?
- Why is endpoint DLP important?
- What is an example of file-level DLP?

10.3 SECURING DATA AND APPLICATIONS

# Web Application Attacks

TESTOUT SECURITY PRO

# Section Skill Overview

- Perform an SQL injection attack.
- Prevent cross-site scripting.
- Exploit SQL on a webpage.

# Key Terms

- Privilege escalation
- Pointer/object dereferencing
- Buffer overflow
- Resource exhaustion
- Memory leak
- Race conditions
- Error handling
- Improper input handling
- Replay attack
- Pass the hash
- API attacks
- SSL stripping
- Driver manipulation

# Key Definitions

- **Privilege escalation:** The exploitation of a misconfiguration, a bug, or design flaw to gain unauthorized access to resources.
- **Pointer/object dereferencing:** An attack that retrieves a value stored in memory that can be exploited through a NULL pointer dereference.
- **Buffer overflow:** An attack that exploits an operating system or an application that does not properly enforce boundaries for inputting data such as the amount of data or the type of data.
- **Resource exhaustion:** An attack that focuses on depleting the resources of a network to create a denial of service to legitimate users.
- **Memory leak:** A leak that happens when dynamic memory is allocated in a program, but no pointers are connected to it causing it to never be returned when requested.
- **Race conditions:** A sequence of events with dependencies that a system is programmed to run in a certain order which can lead to a time-of-check to time-of-use bug vulnerability.

# Key Definitions

- **Error handling:** The procedures in a program that respond to irregular input or conditions.
- **Improper input handling:** The lack of validation, sanitization, filtering, decoding, or encoding of input data.
- **Replay attack:** An attack that happens when network traffic is intercepted by an unauthorized person who then delays or replays the communication to its original receiver, acting as the original sender. The original sender is unaware of this occurrence.
- **Pass the hash:** An attack in which an attacker obtains a hashed password and uses it to gain unauthorized access.
- **API attacks:** A malicious use of an API (application programming interface).
- **SSL stripping:** An attack that focuses on stripping the security from HTTPS - enabled websites.
- **Driver manipulation:** An attack that focuses on device drivers. The attack uses refactoring or shimming.

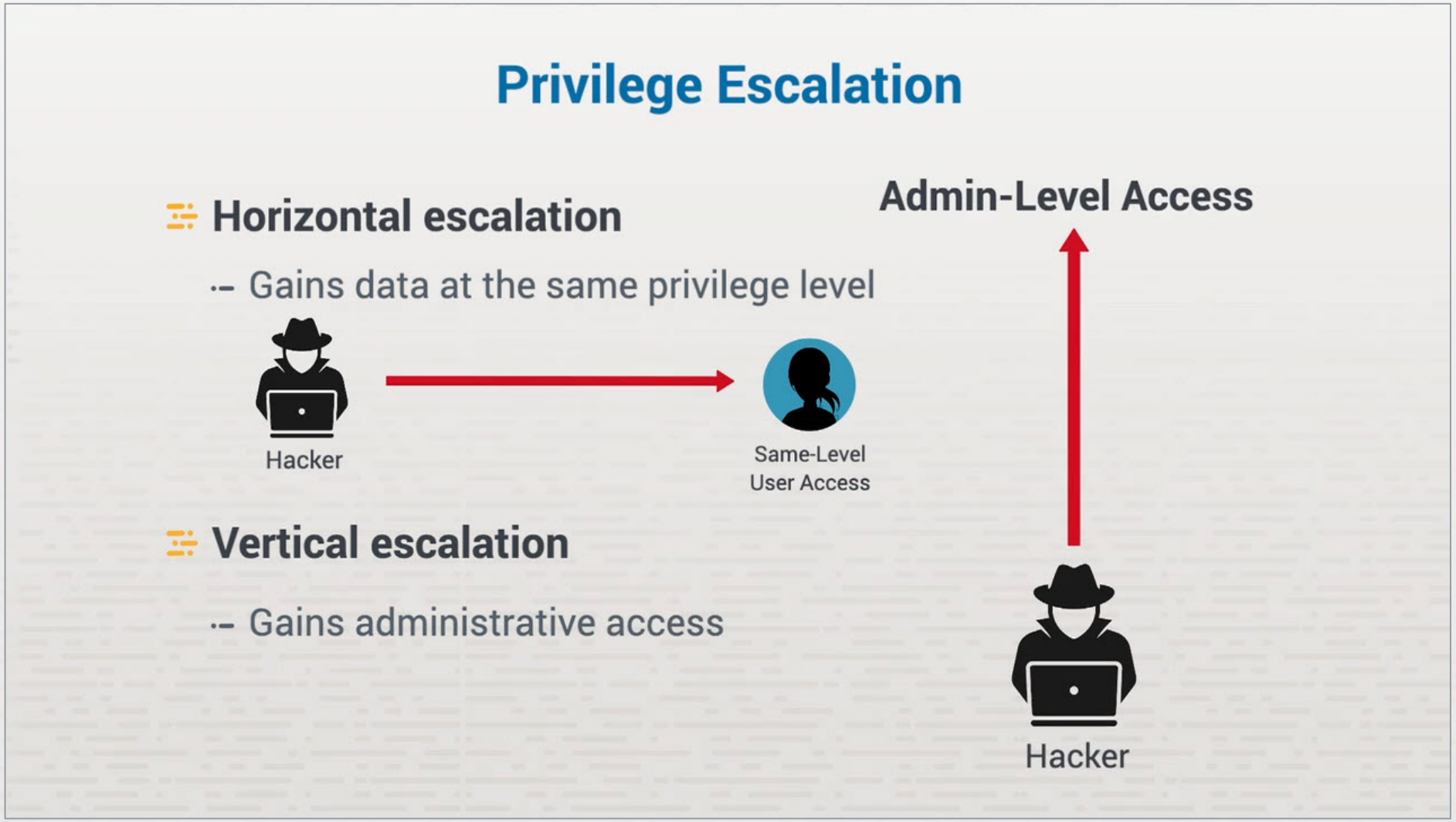
# Web Application Attacks 1

# Web Application Attacks 1

## Privilege Escalation

- **Horizontal escalation**
  - Gains data at the same privilege level

# Web Application Attacks 1



# Web Application Attacks 1

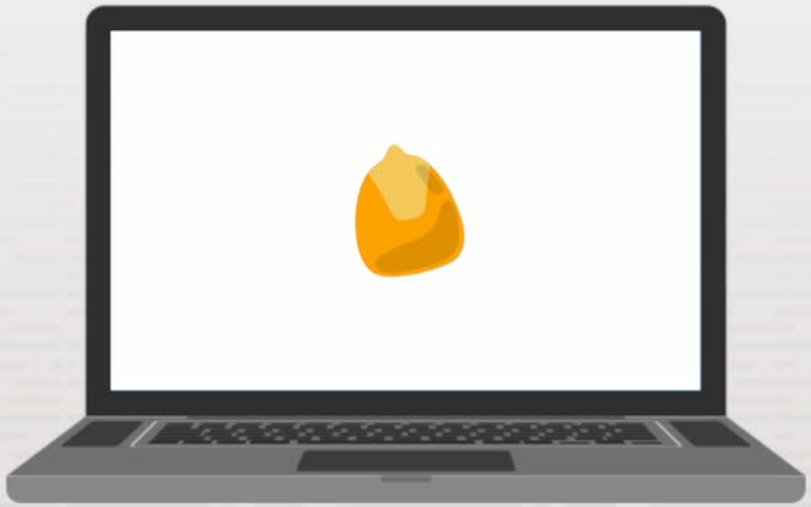
## Pointer/Object Dereference Attack

A pointer stores a memory address.


Dereferencing the pointer

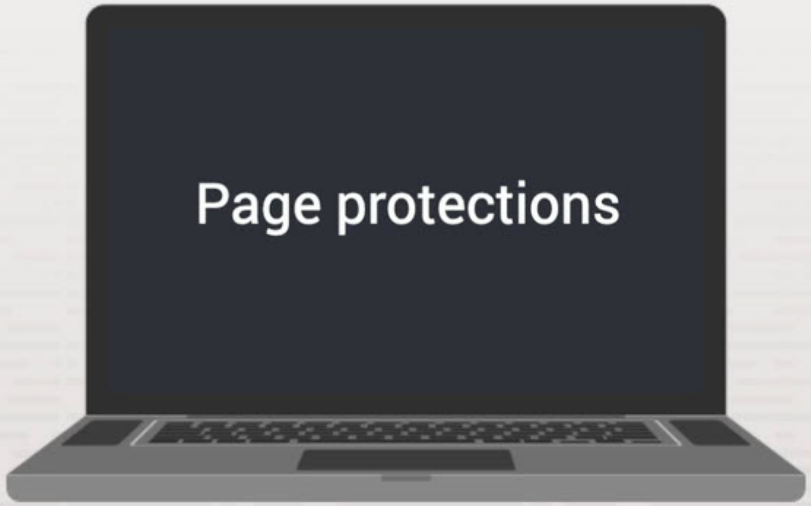
# Web Application Attacks 1

## Pointer/Object Dereference Attack



# Web Application Attacks 1

## Pointer/Object Dereference Attack



# Web Application Attacks 1

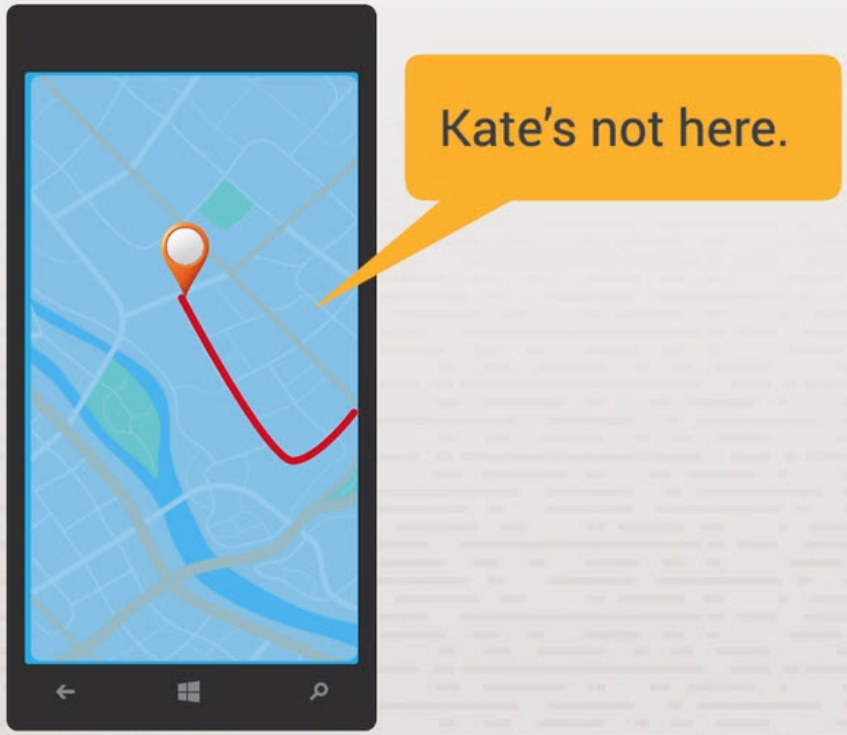
## Pointer/Object Dereference Attack

# Web Application Attacks 1

## Pointer/Object Dereference Attack

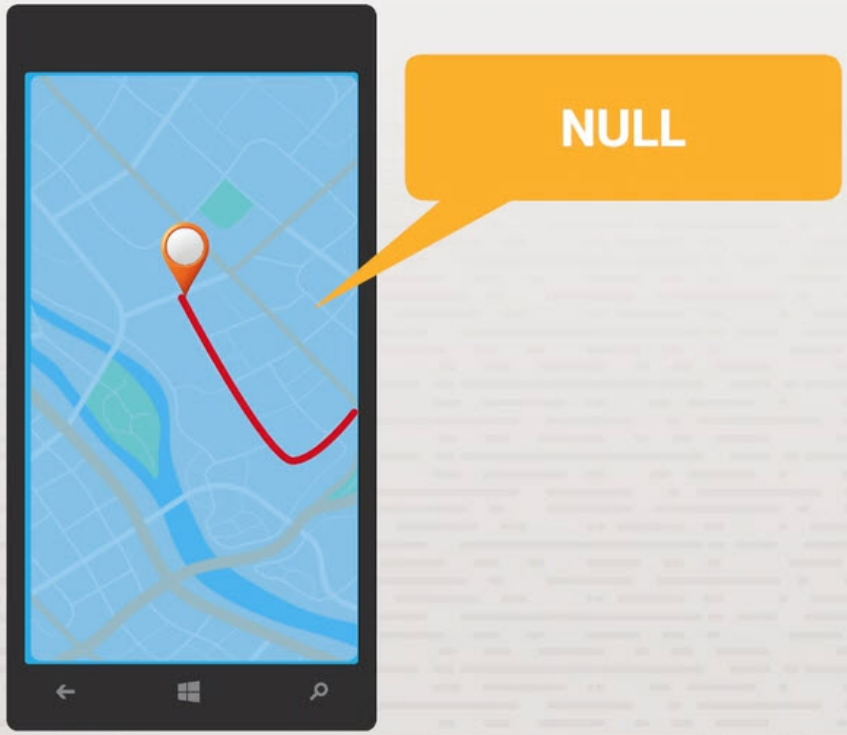
# Web Application Attacks 1

## Pointer/Object Dereference Attack



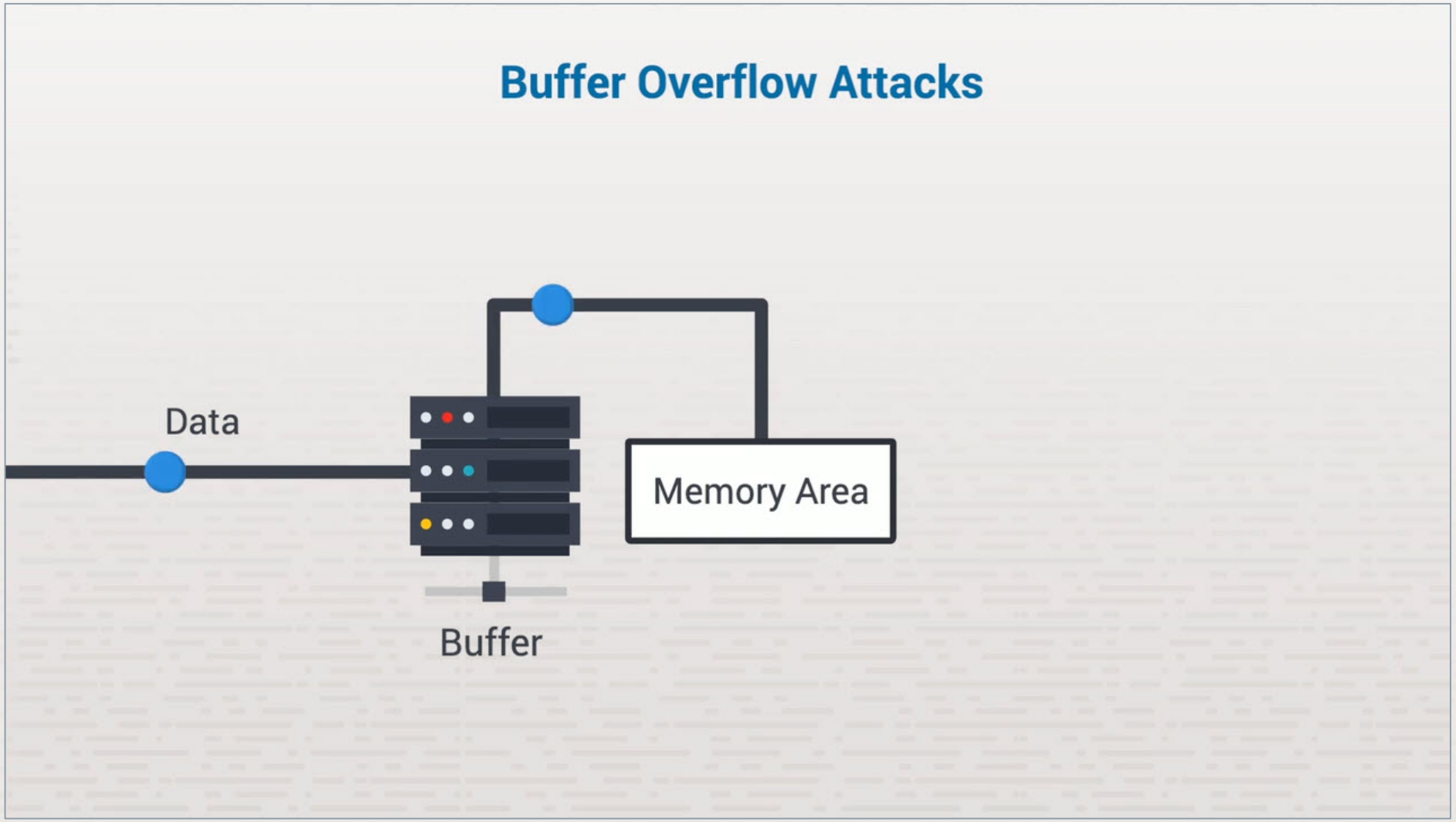
# Web Application Attacks 1

## Pointer/Object Dereference Attack

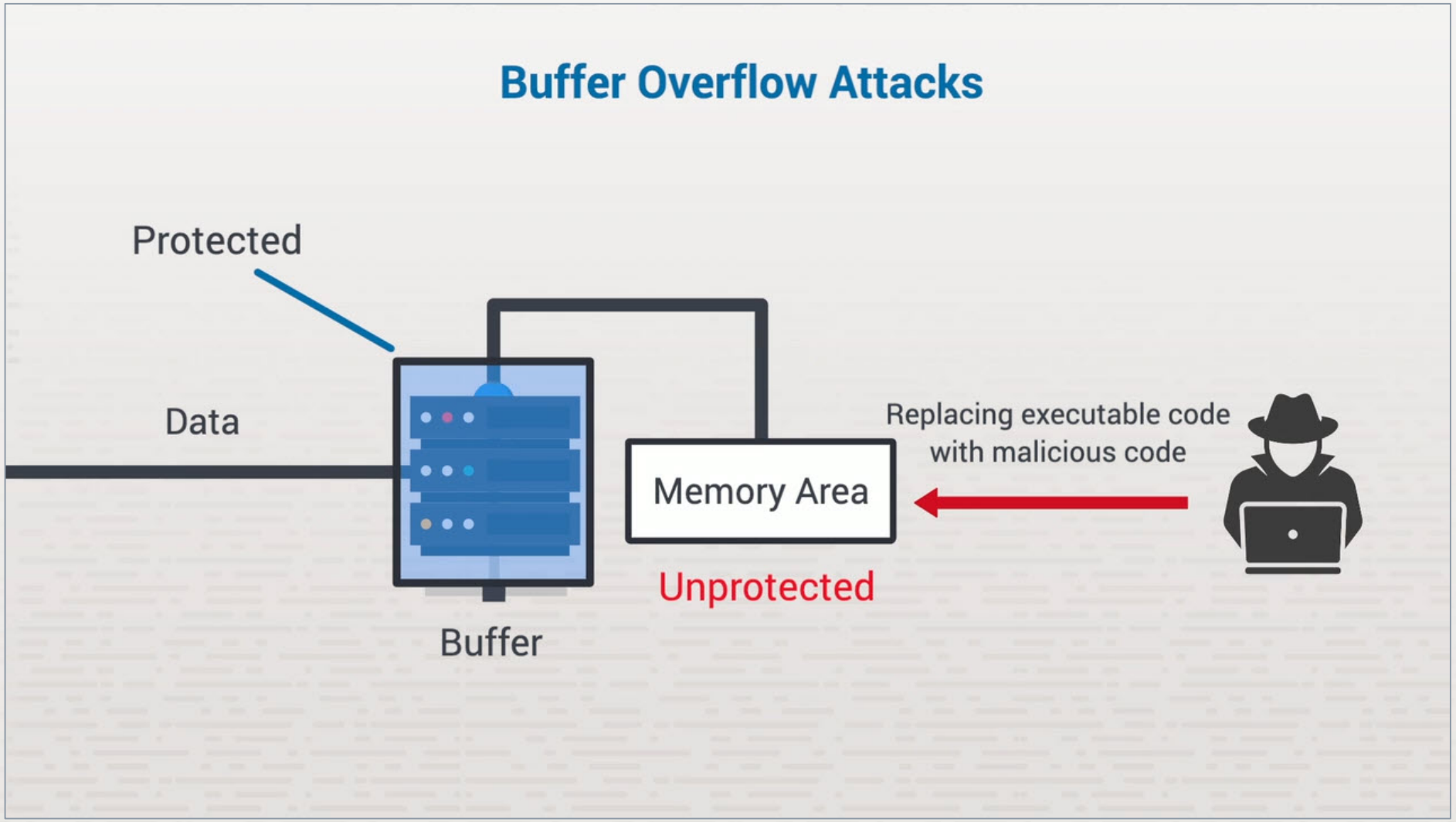


# Web Application Attacks 1

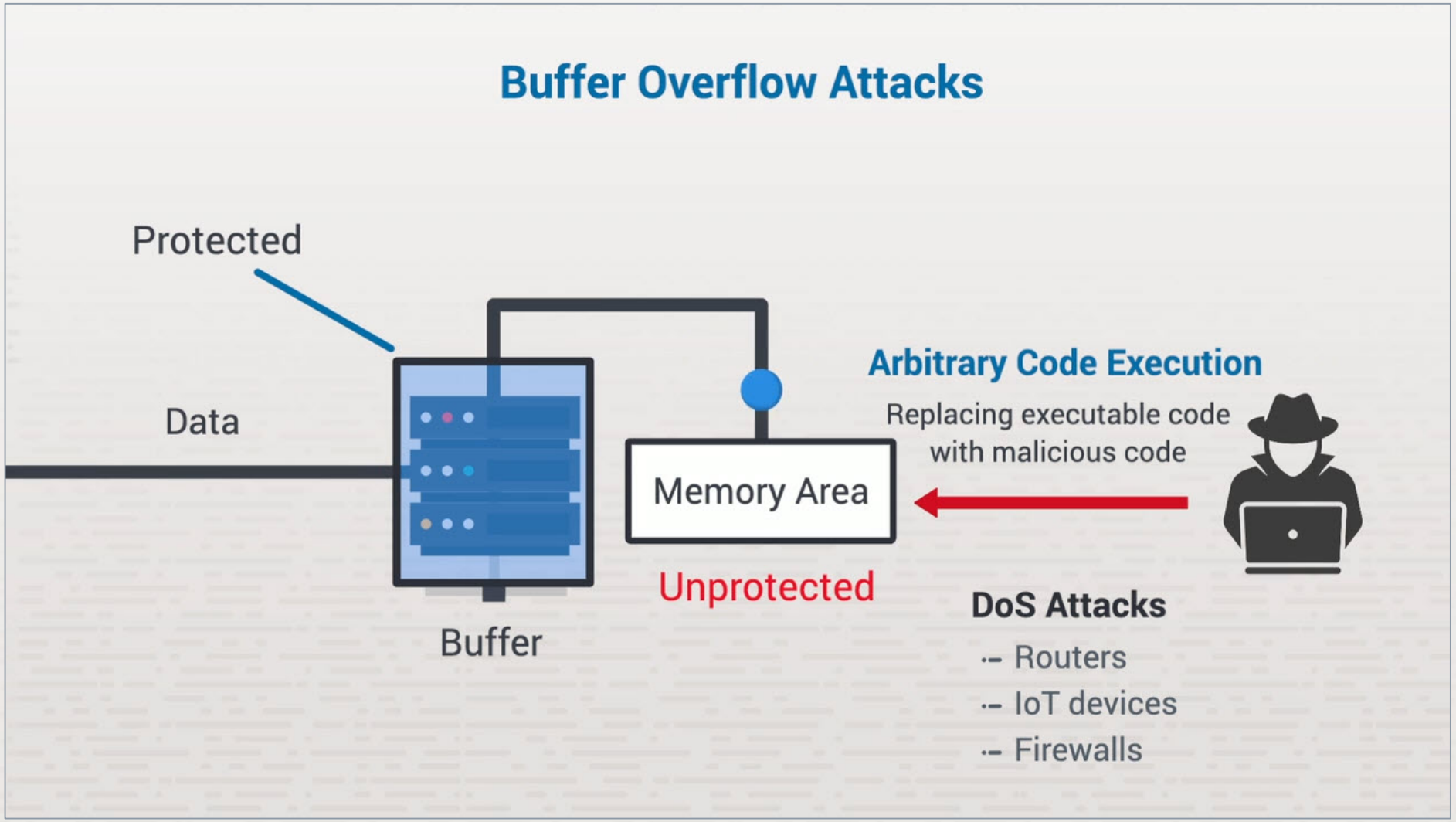
# Web Application Attacks 1



# Web Application Attacks 1



# Web Application Attacks 1



# Web Application Attacks 1

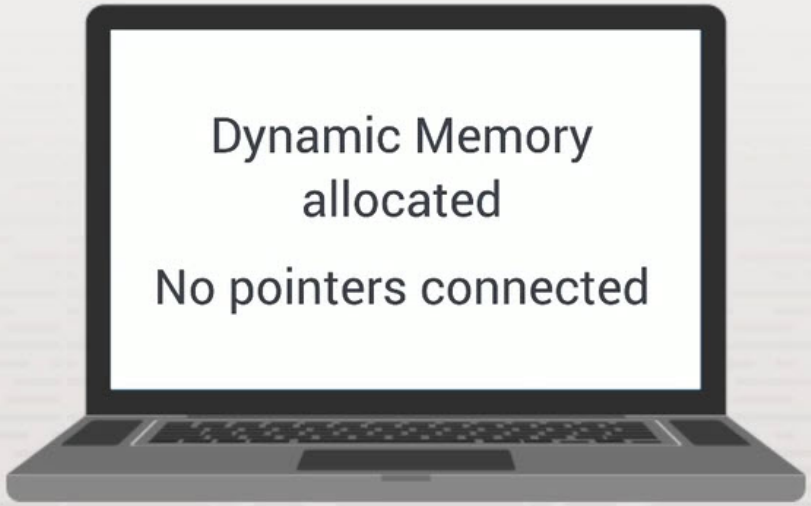
## Resource Exhaustion Attacks

Deplete the resources of a network to create a DoS

- Slow header attacks
  - Set HTTP header timeouts
- Slow POST attacks
  - Through forms, logins, feedback input fields
  - Set maximum body size for forms
  - Set web server with max total transfer time
- Focus on:
  - Memory
  - File system storage
  - Database connection-pool entries
  - CPU
  - Set controls on size of the resource or number

# Web Application Attacks 1

## Memory Leak Attacks

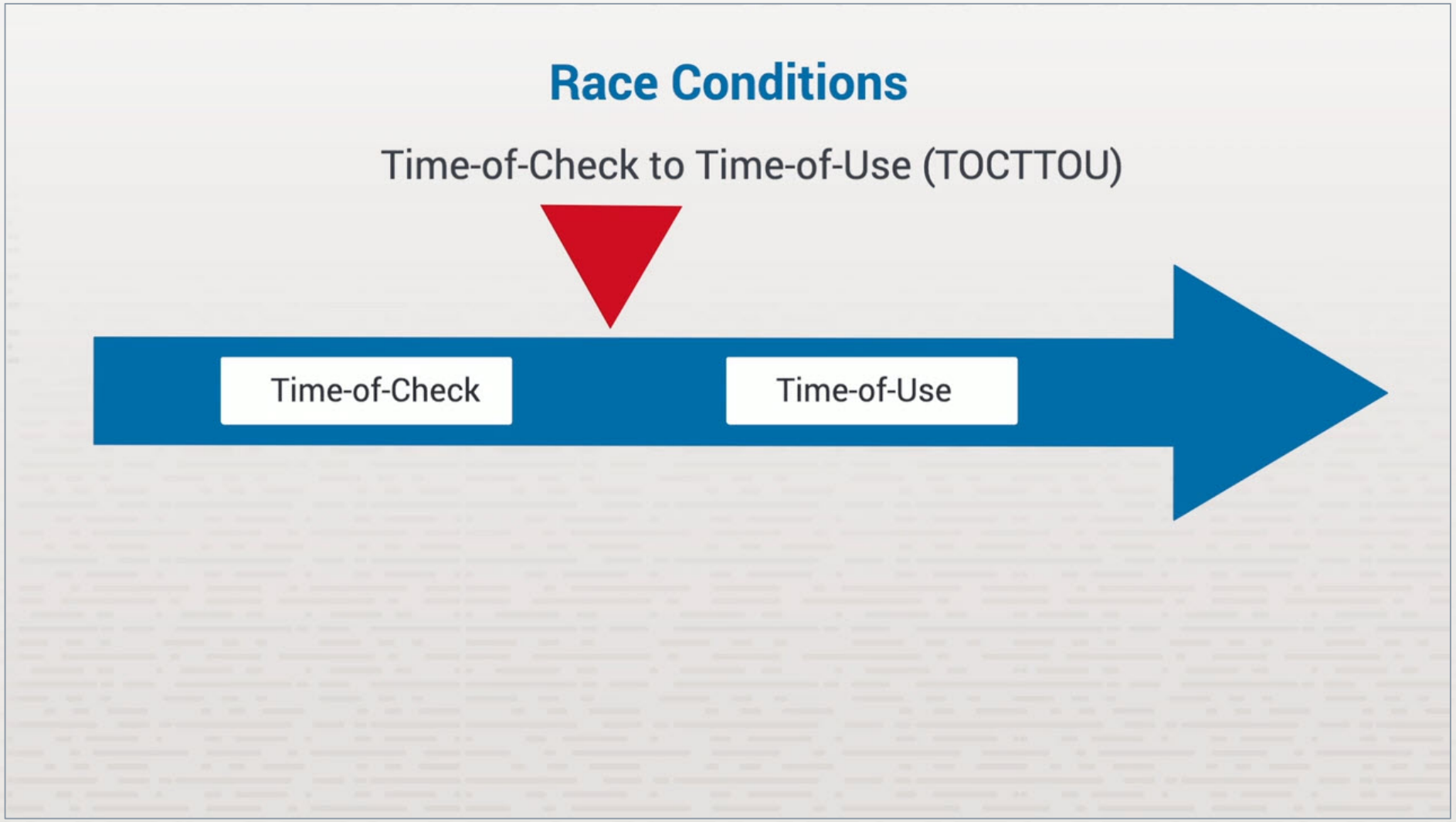


# Web Application Attacks 1

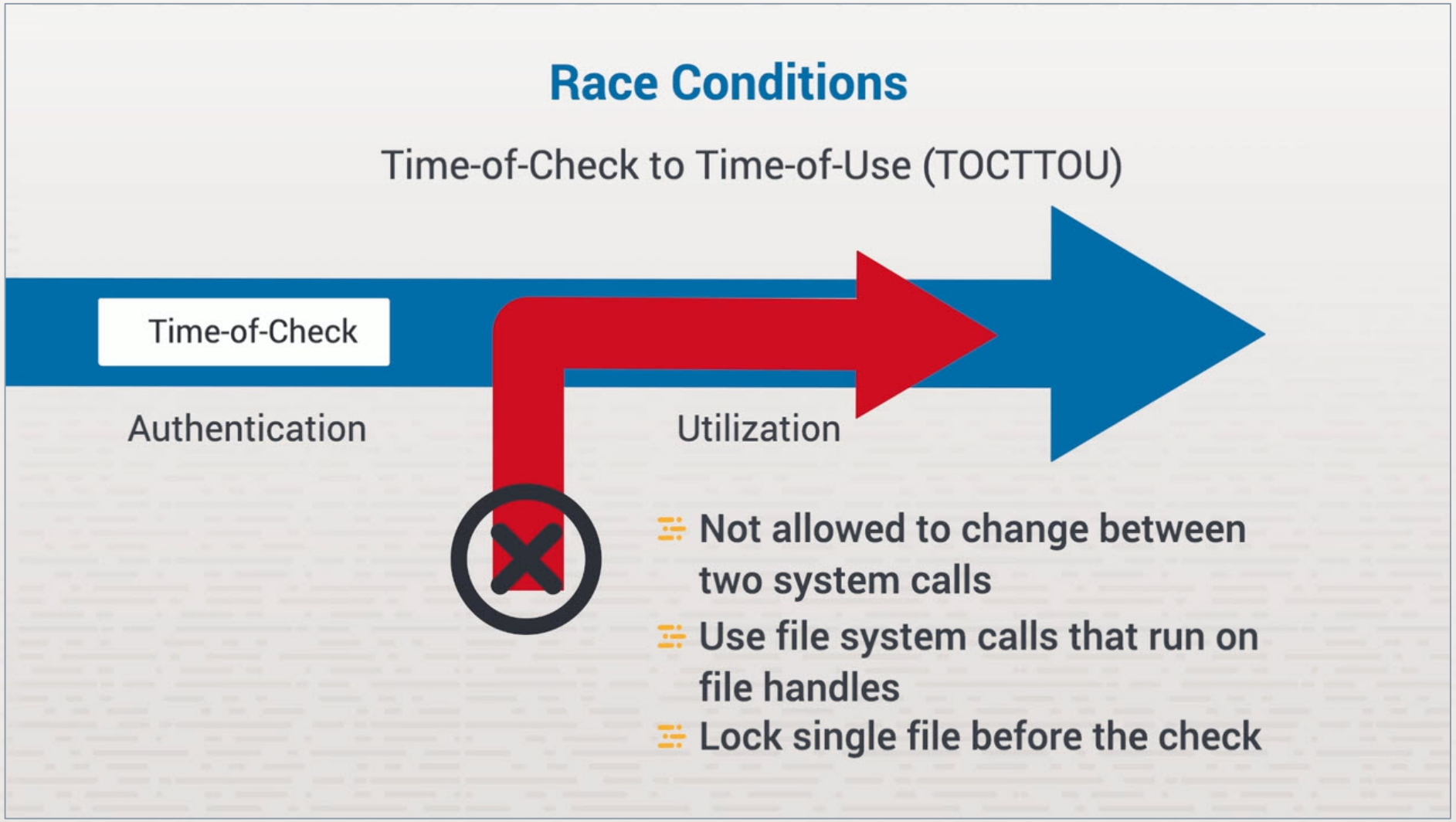
## Memory Leak Attacks


- Resource exhaustion
- DoS
- Exploit areas affected by low-memory conditions

# Web Application Attacks 1



# Web Application Attacks 1



# Summary

- Privilege escalation
- Pointer dereference
- Buffer overflow
- Resource exhaustion
- Memory leaks
- Race conditions

# Web Application Attacks 2

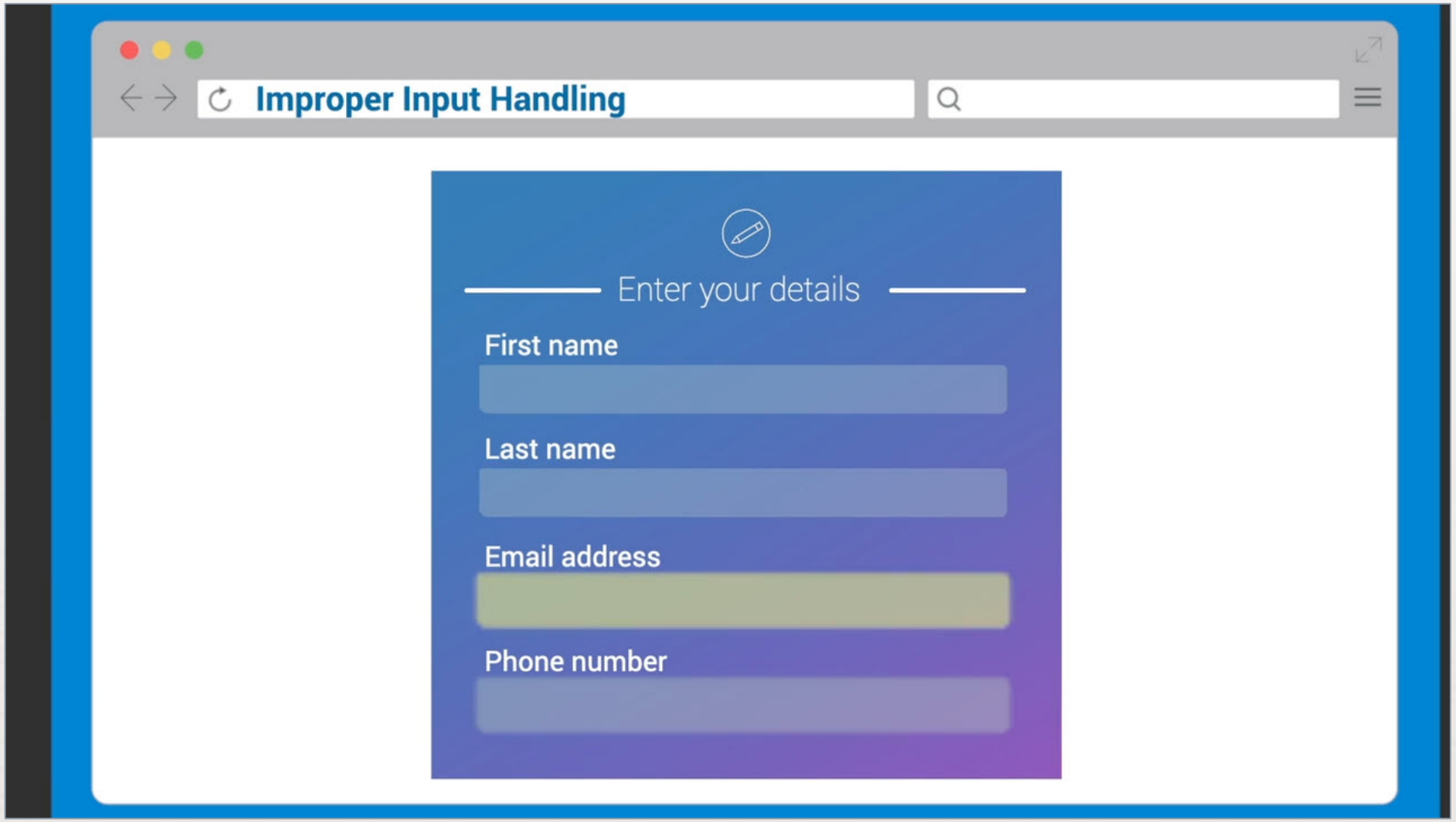
# Error Handling

- Too much information
  - Malformed query
    - Query logic
    - Sensitive data
  - Path transversal weakness attempt
    - Full pathname
- To protect:
  - Minimal information error message
  - Useful only to intended audience

# Improper Input Handling

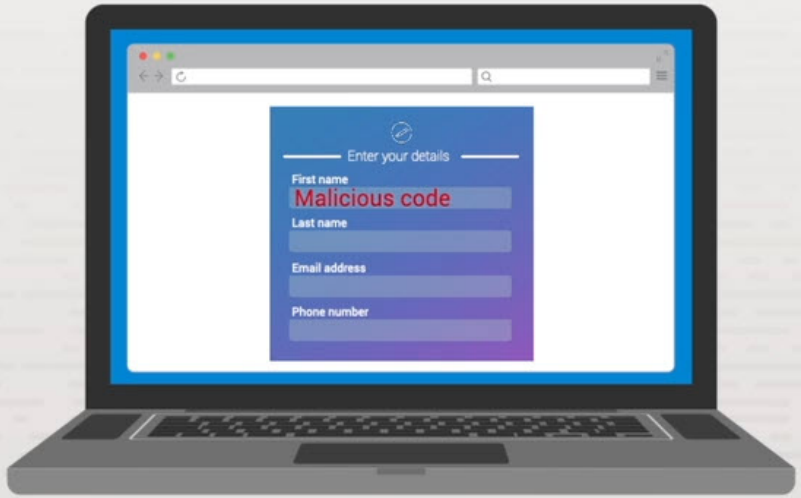
- Most common attack today
- App fails to validate input
- Vulnerable to:
  - Buffer overflows
  - Cross-site scripting
  - Directory transversal
  - NULL byte injections
  - SQL injections
  - Uncontrolled format strings
  - DoS attacks
  - OS commanding

# Web Application Attacks 2



# Web Application Attacks 2

## Improper Input Handling

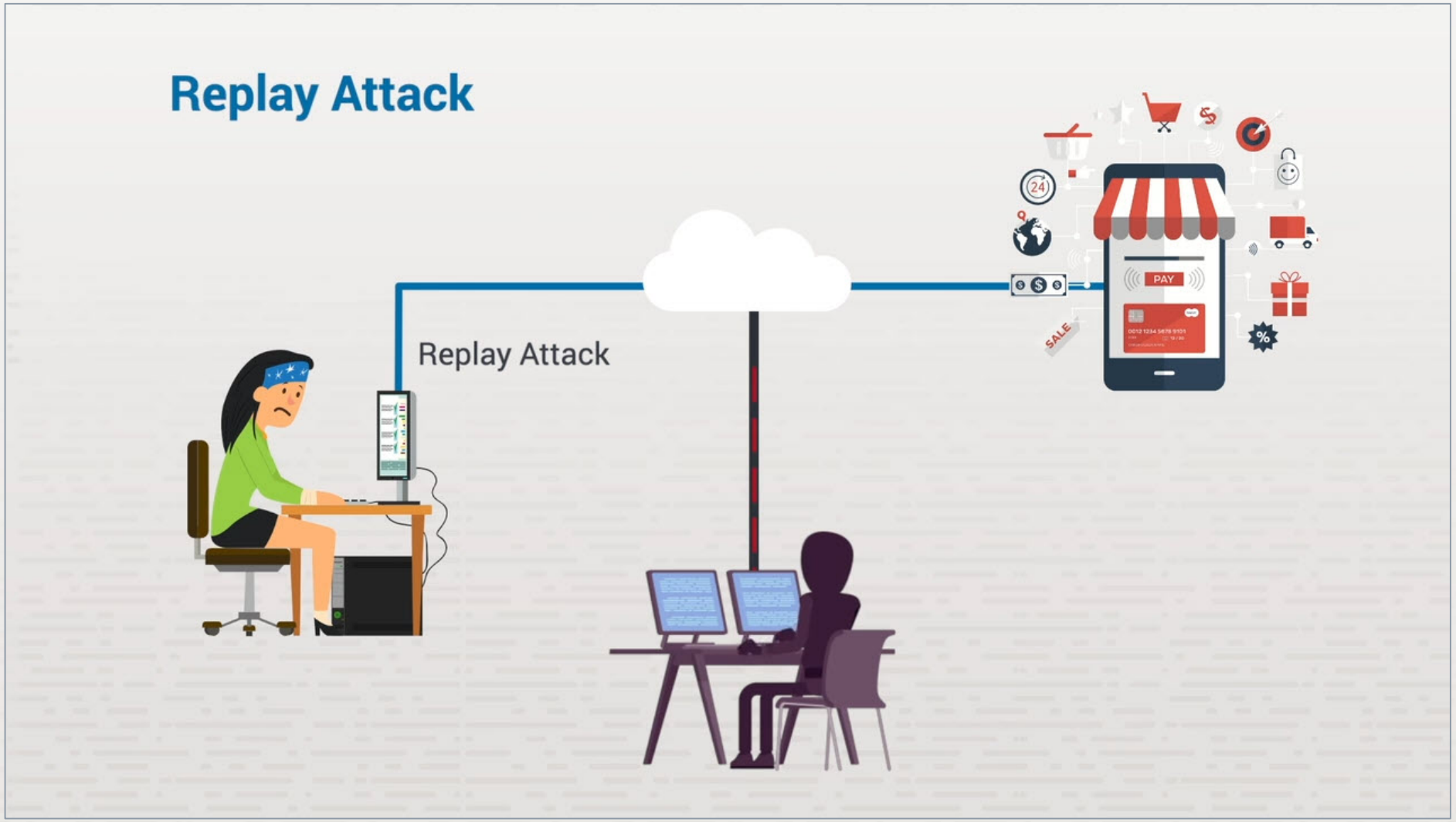


# Web Application Attacks 2

## Replay Attack

- Session replay attack
- Traffic intercepted – delayed or replayed
- Original sender unaware
- Man-in-the-middle attack

# Web Application Attacks 2



# Replay Attack Prevention

- Strong digital signatures
  - Timestamps
- Session keys
  - Time - and process-bound
- Sequence numbers

# API

- Communicates between:
  - Systems within
  - Other organizations
  - Customers
  - Websites
- Openly published
  - May use internally
  - Interface exploits

# API Protection

- Rate limiting
- Security logs
- Access logs
- SQL injections
- Error notifications

# Web Application Attacks 2

## Secure Sockets Layer (SSL) Stripping

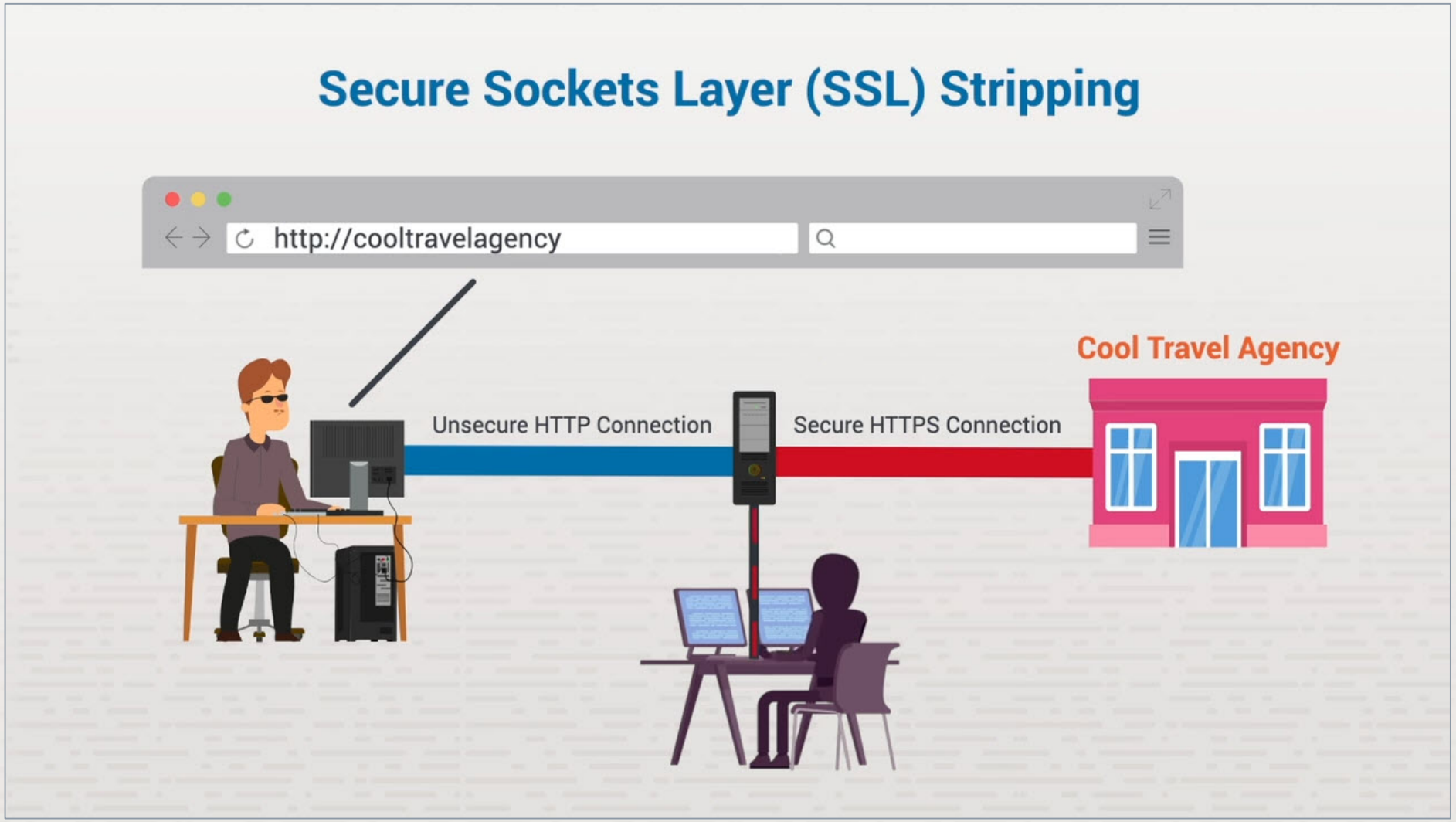
Strips the security from HTTPS-enabled website

- **Intercepting initial request**
- **Attacker makes secure connection with intended server**
- **Attacker makes unsecure HTTP connection with the user**

# Web Application Attacks 2

## Secure Sockets Layer (SSL) Stripping

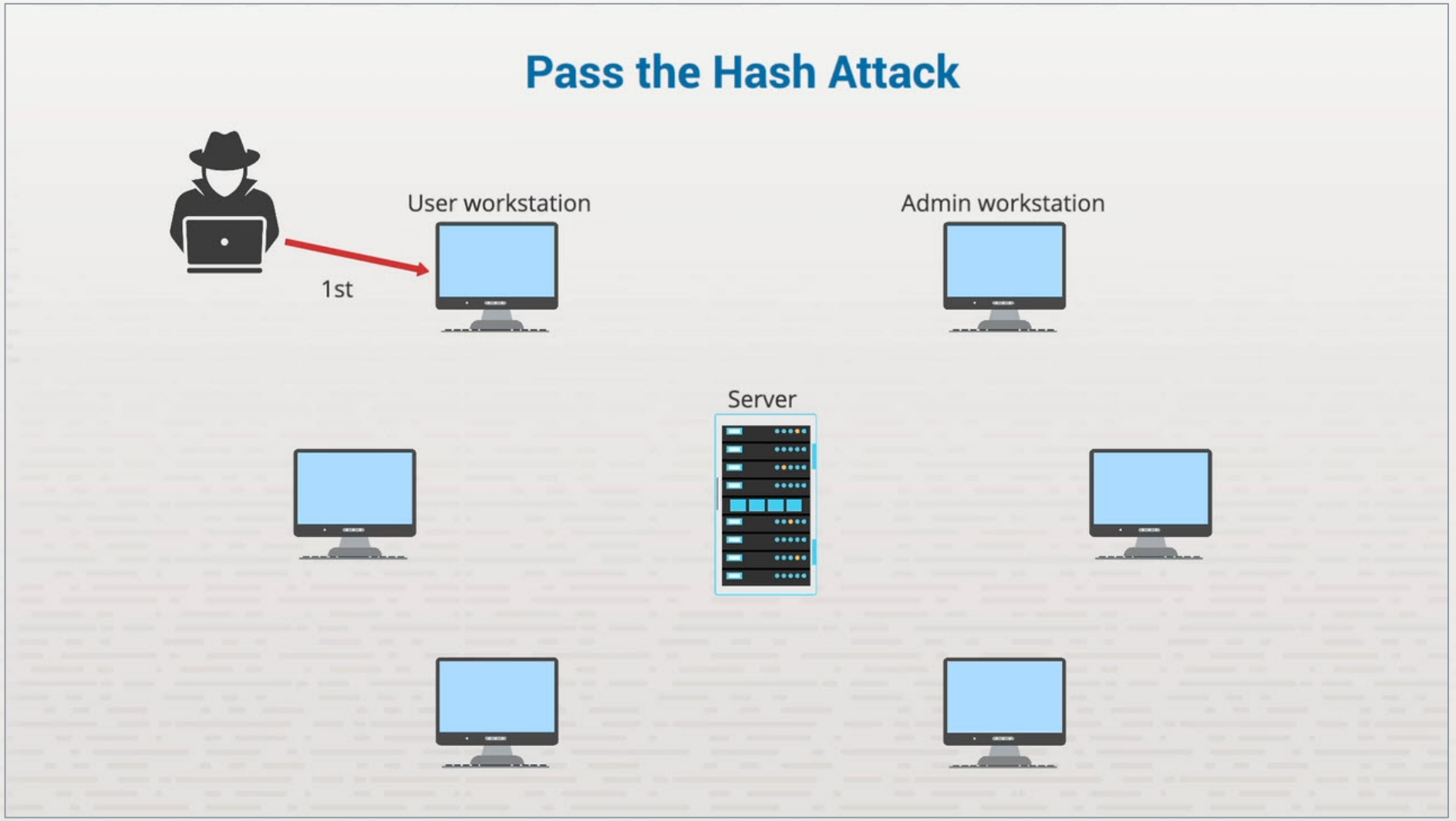
# Web Application Attacks 2



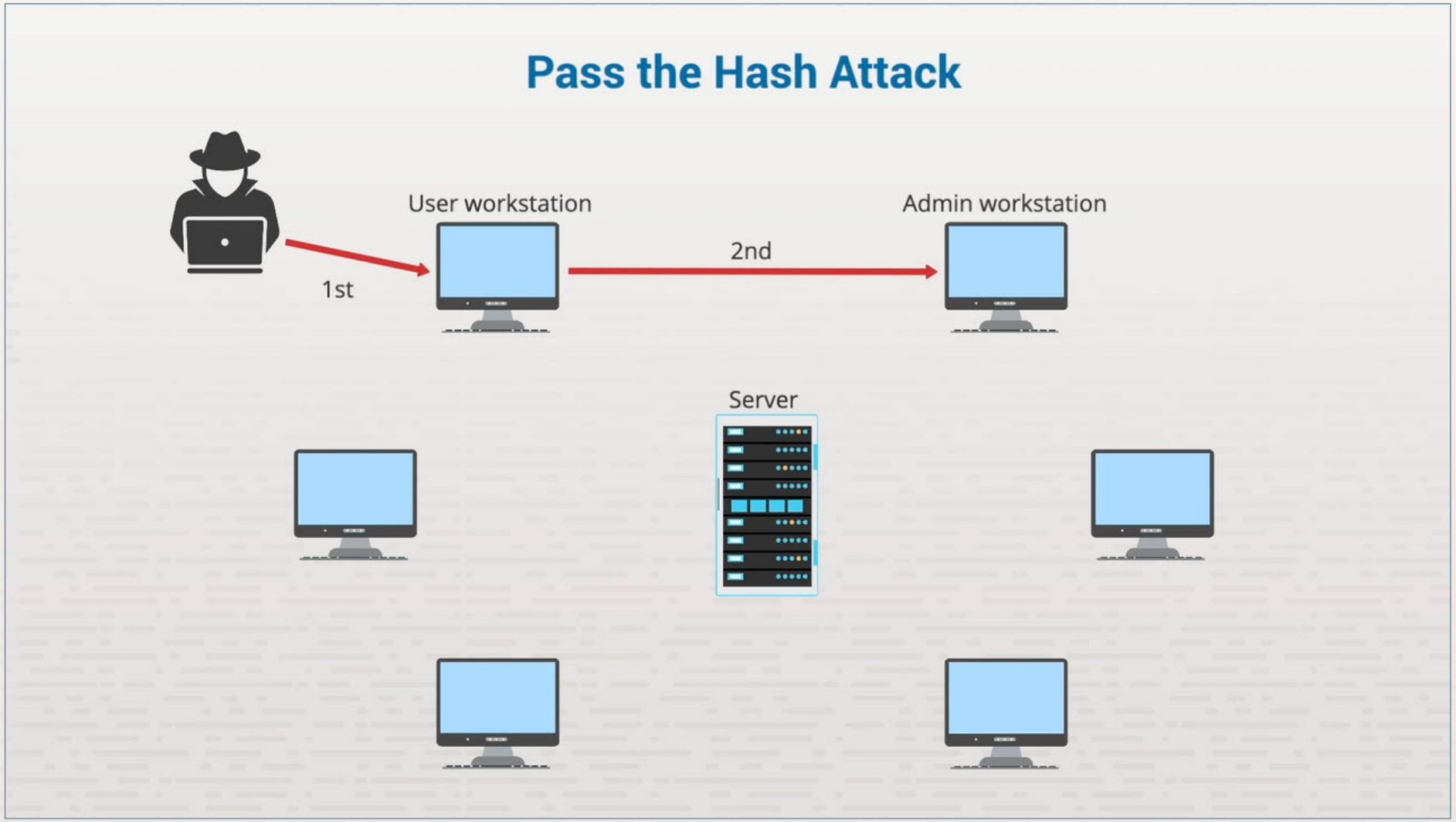
# Driver Manipulation

- Shimming
  - Additional code for compatibility
- Code refactoring
  - Rewriting internal process of code to protect against:
    - Frequent backups
    - Antivirus software
    - Malicious URLs

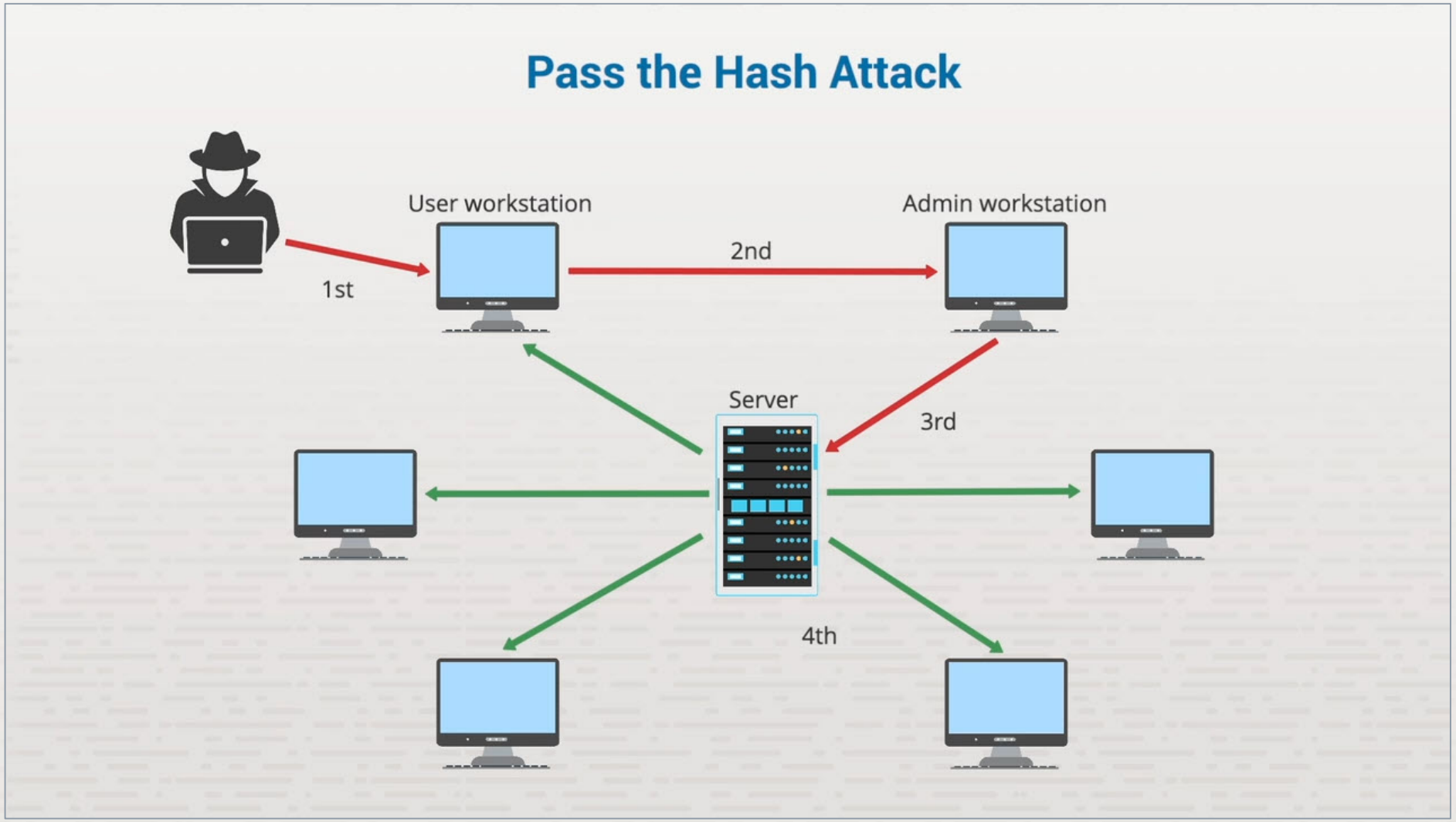
# Web Application Attacks 2



# Web Application Attacks 2



# Web Application Attacks 2



# Pass the Hash

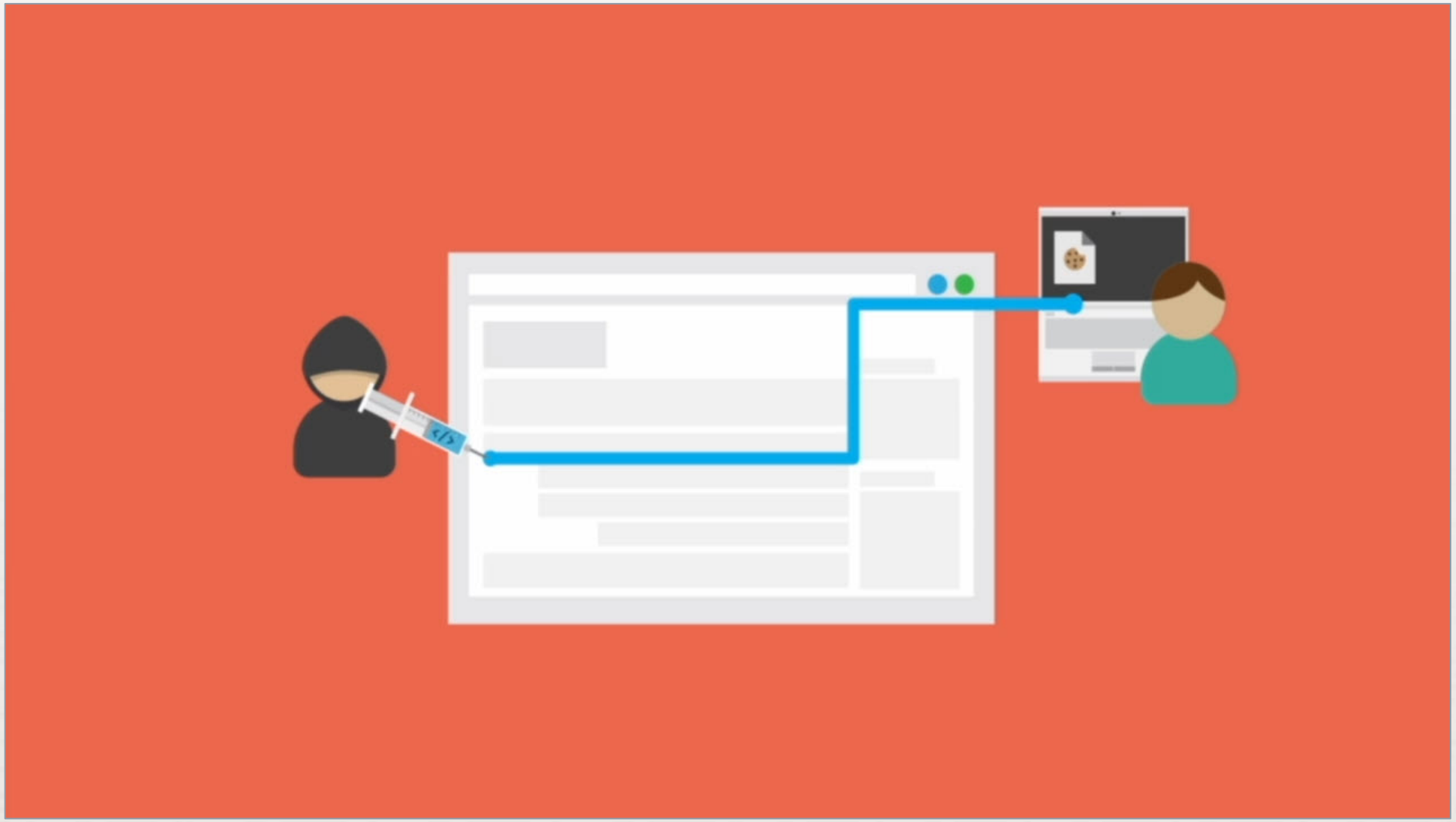
- Prevent access to others' workstations
- Disable Remote Desktop connections
- Standard user accounts for admins

# Summary

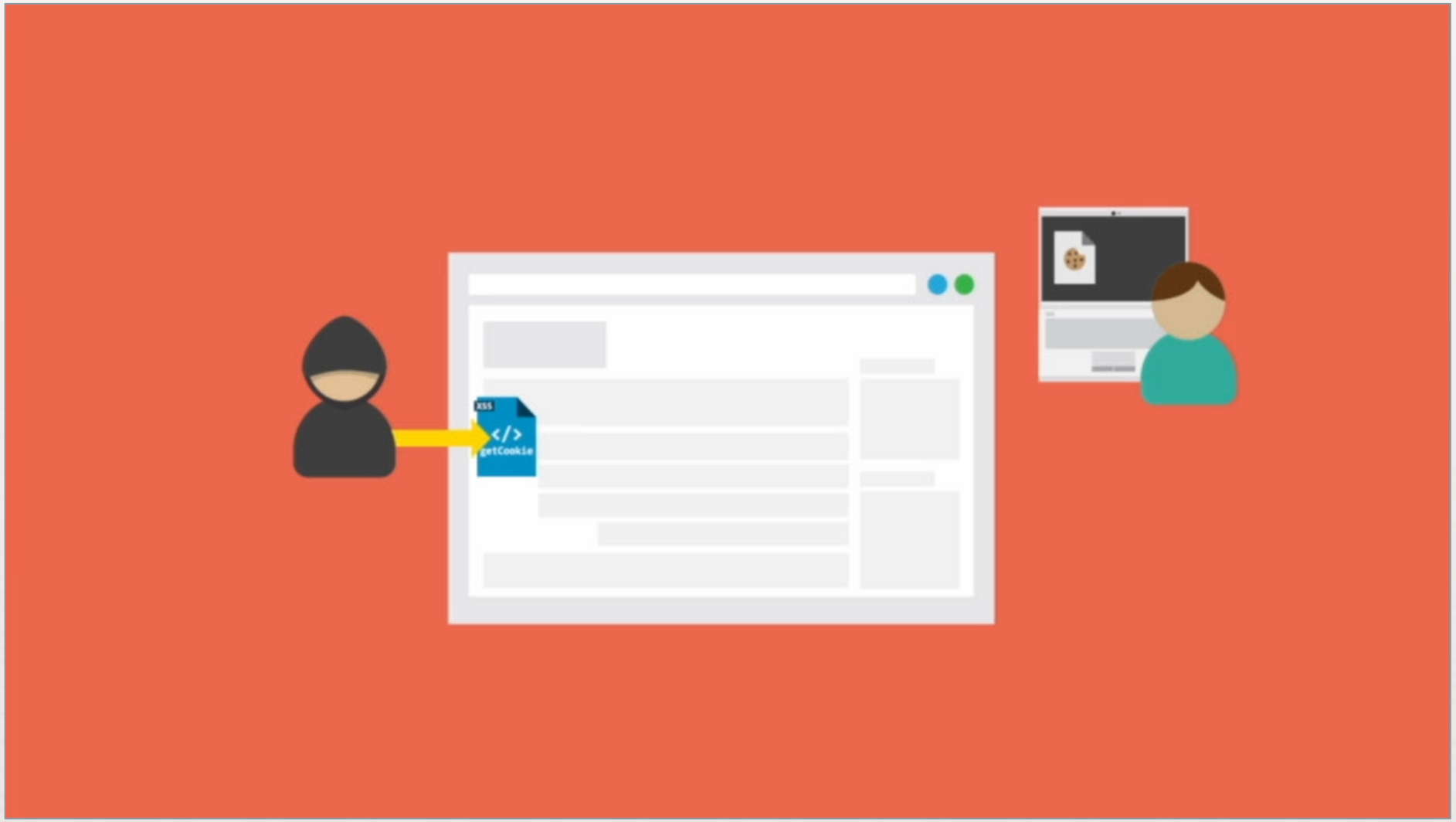
- Error handling
- Improper input handling
- Replay sessions
- API
- SSL stripping
- Driver manipulation
- Pass-the-hash

# XSS and CSRF Attacks

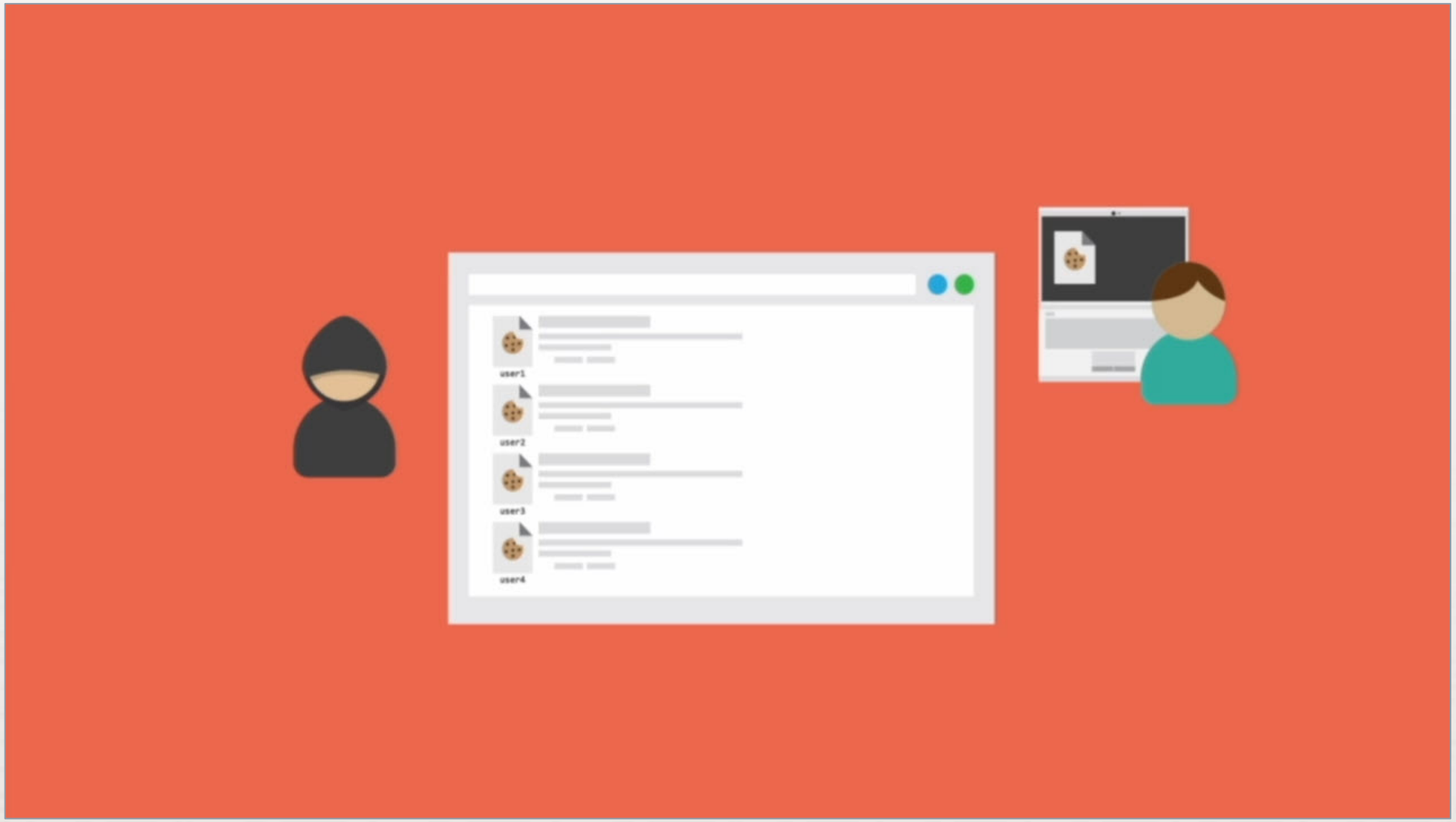
# XSS and CSRF Attacks



# XSS and CSRF Attacks



# XSS and CSRF Attacks



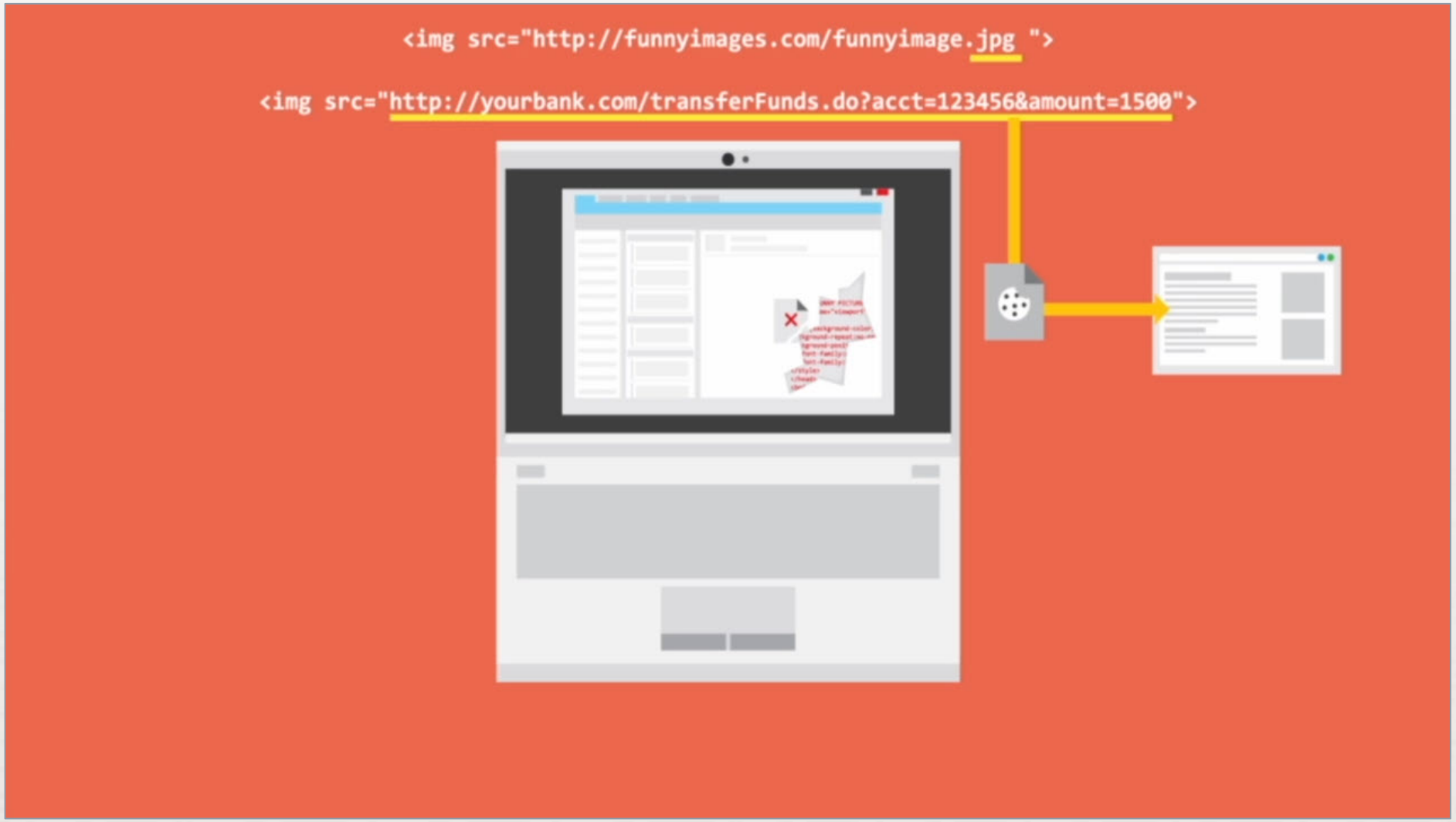
# XSS and CSRF Attacks

# XSS and CSRF Attacks

# XSS and CSRF Attacks

# XSS and CSRF Attacks

# XSS and CSRF Attacks



# CSRF Win Conditions

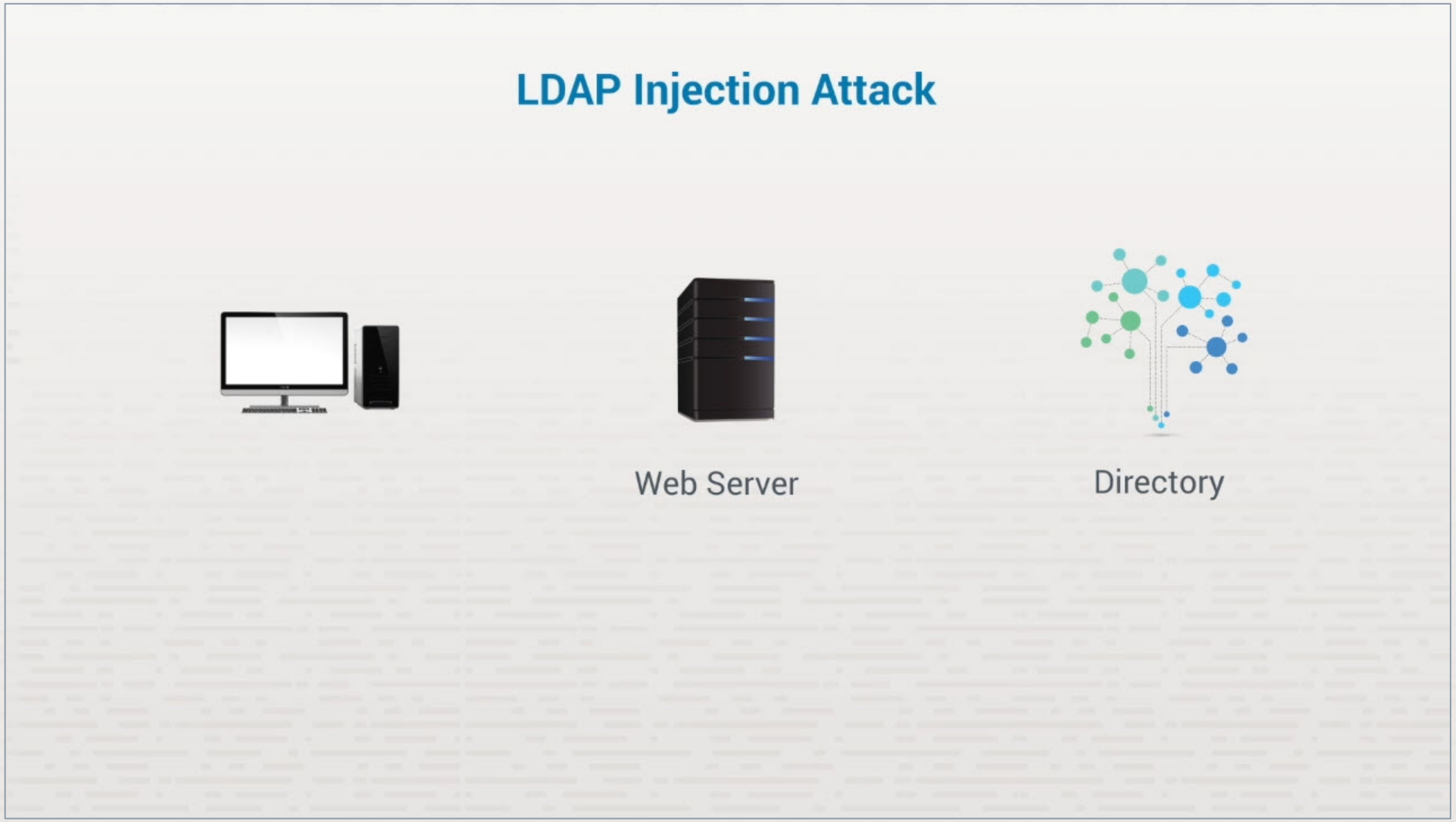
- Target site doesn't check referring header
- Attacker must understand form syntax
- Key Pll information must be known
- User authentication must be possible and transparent

# Preventing CSRF Attacks

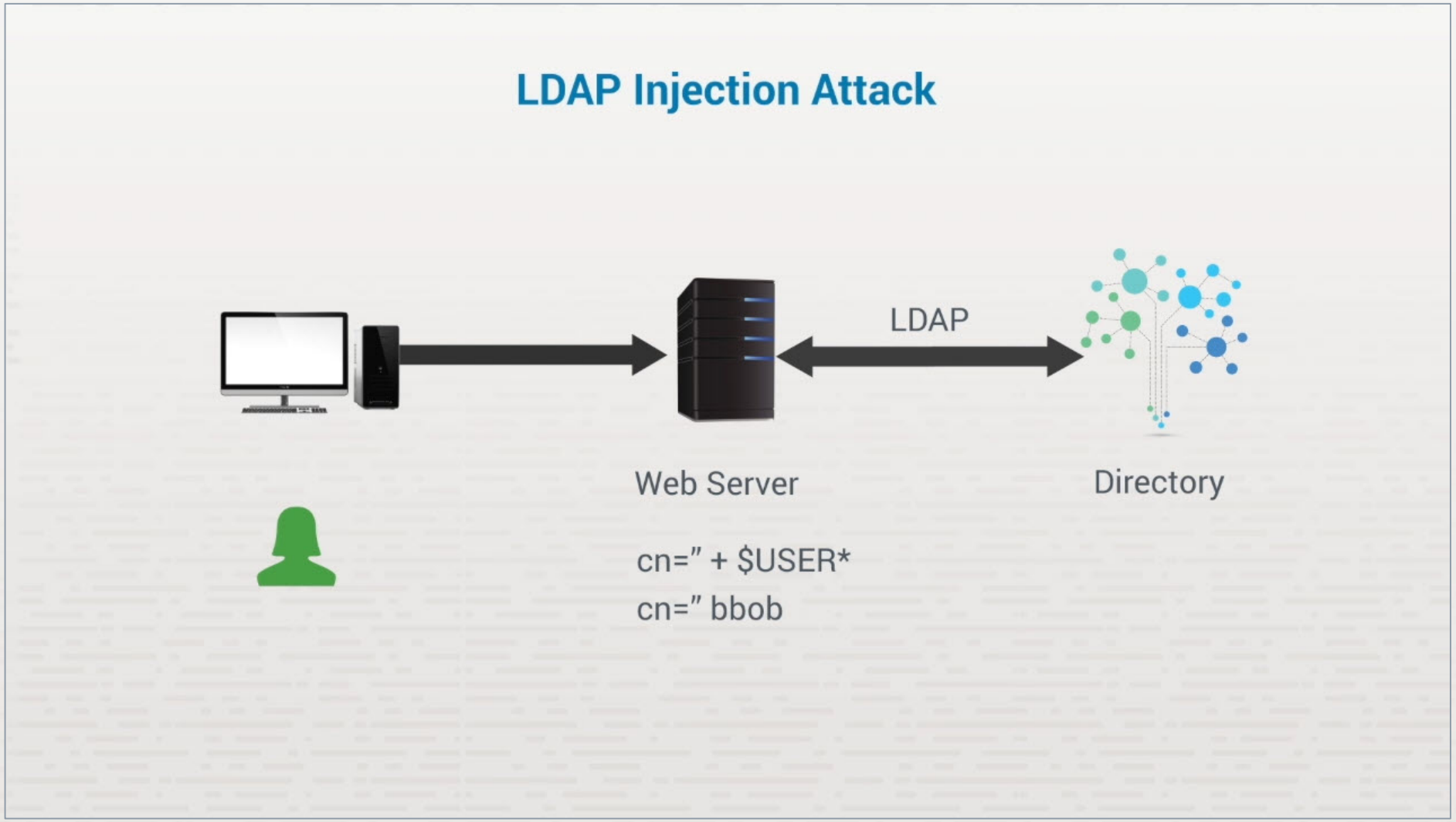
- Implement secure development practices
  - Paired tokens
  - Idle timeouts
- Train end users
  - Always log out of sites
  - Close unneeded browser tabs/windows
  - Don't use the Remember Me feature
  - Never load images from unknown sources

# Injection Attacks

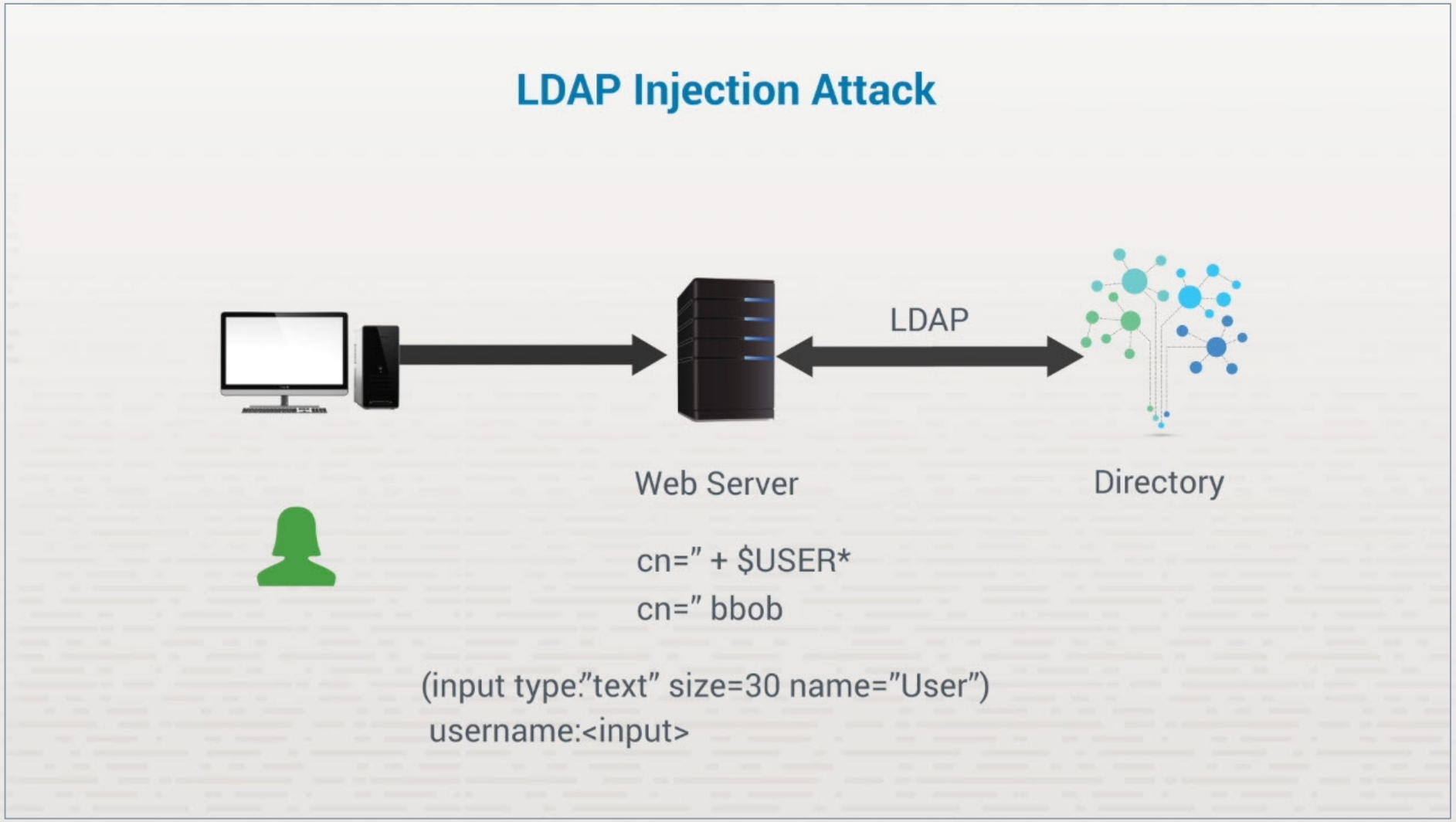
# Injection Attacks



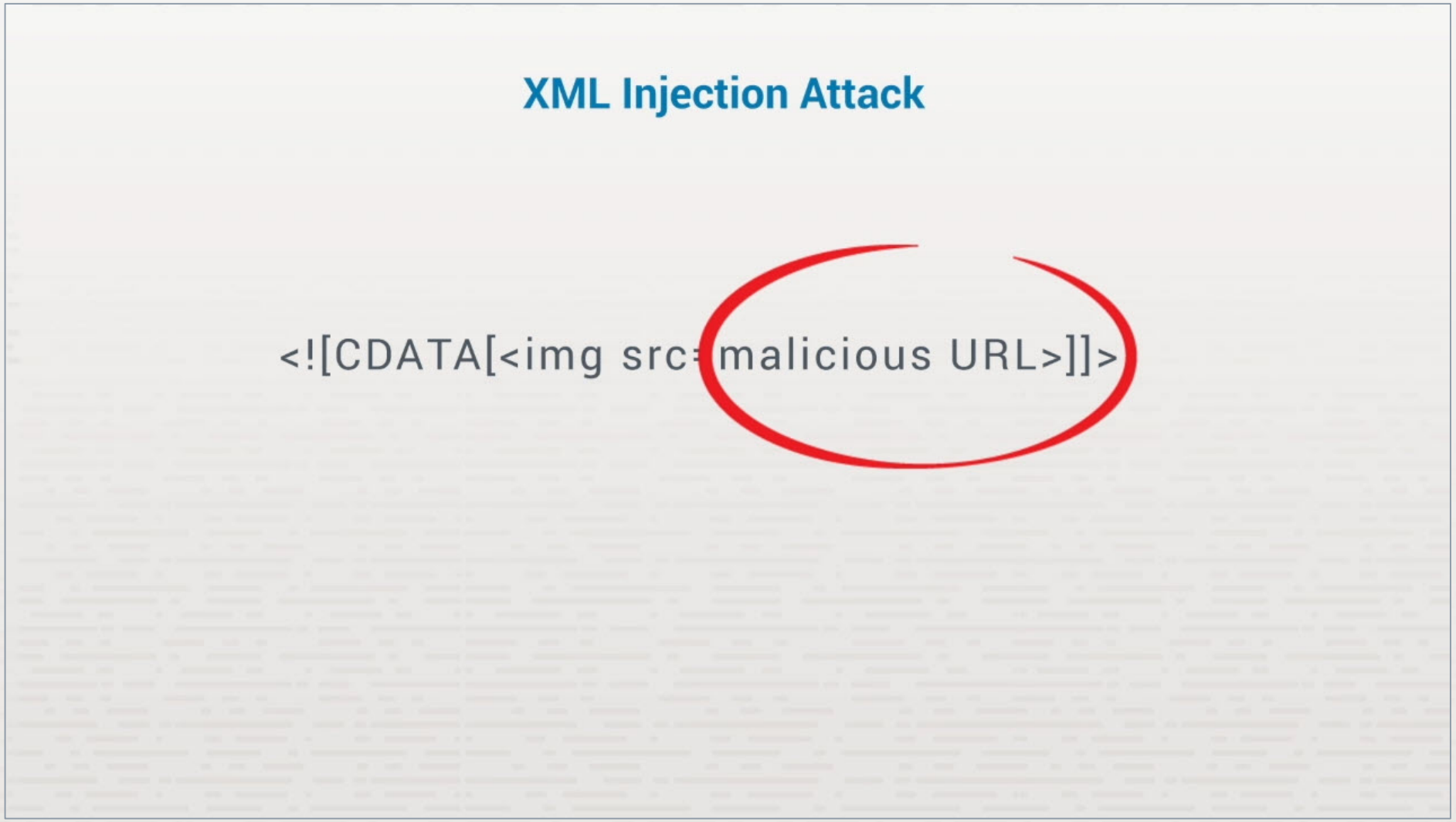
# Injection Attacks



# Injection Attacks



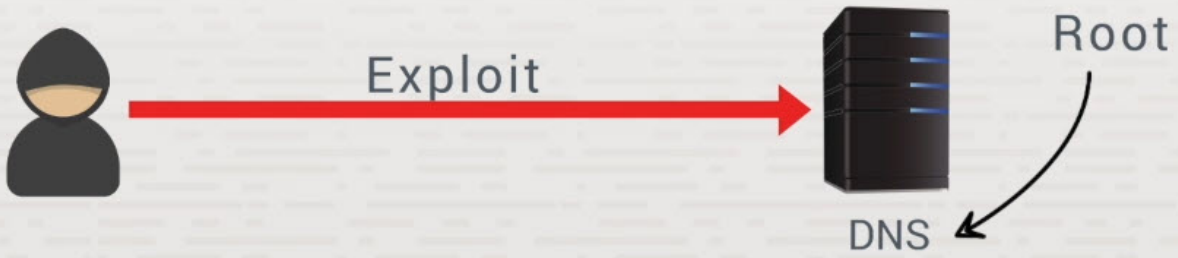
# Injection Attacks



# Injection Attacks

## Command Injection

- Vulnerable application runs commands on behalf of attacker
- Key : runs with privileges of the application

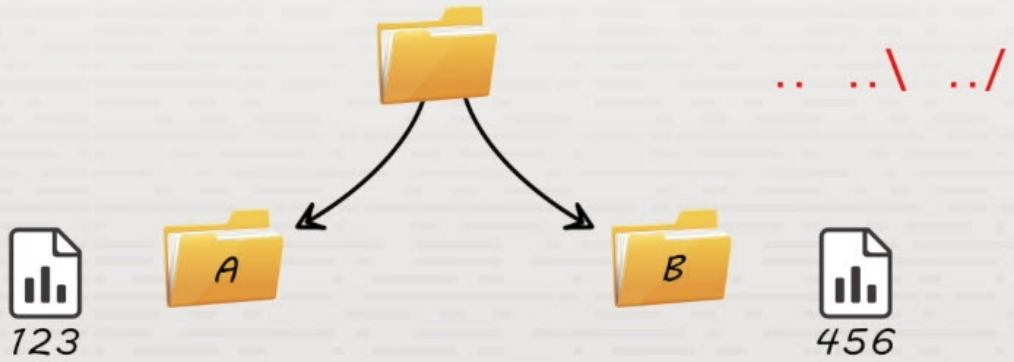


# Injection Attacks

## Directory Traversal

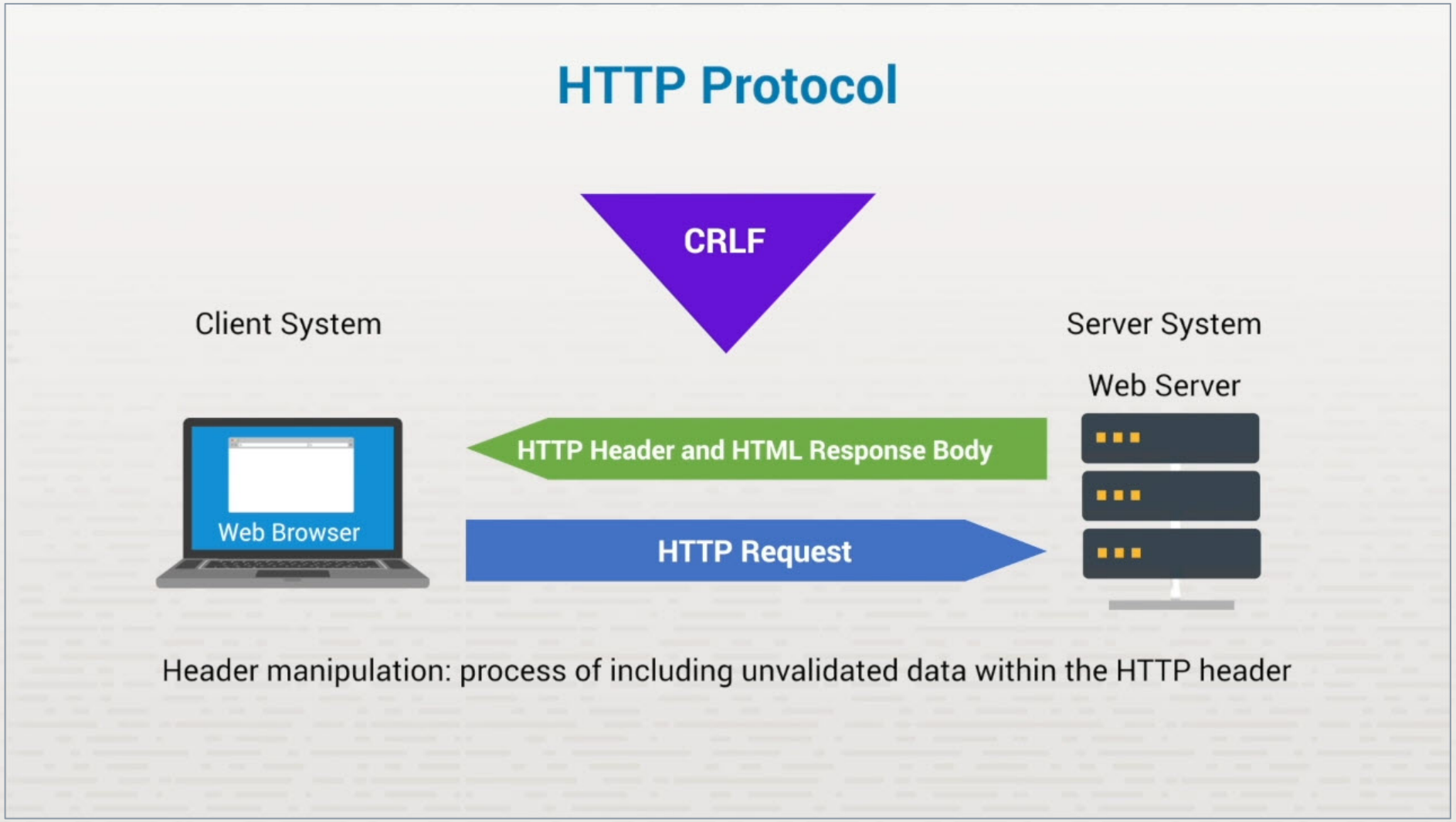
Sample PHP code snippet:

%2e%2e%2f



# Header Manipulation

# Header Manipulation

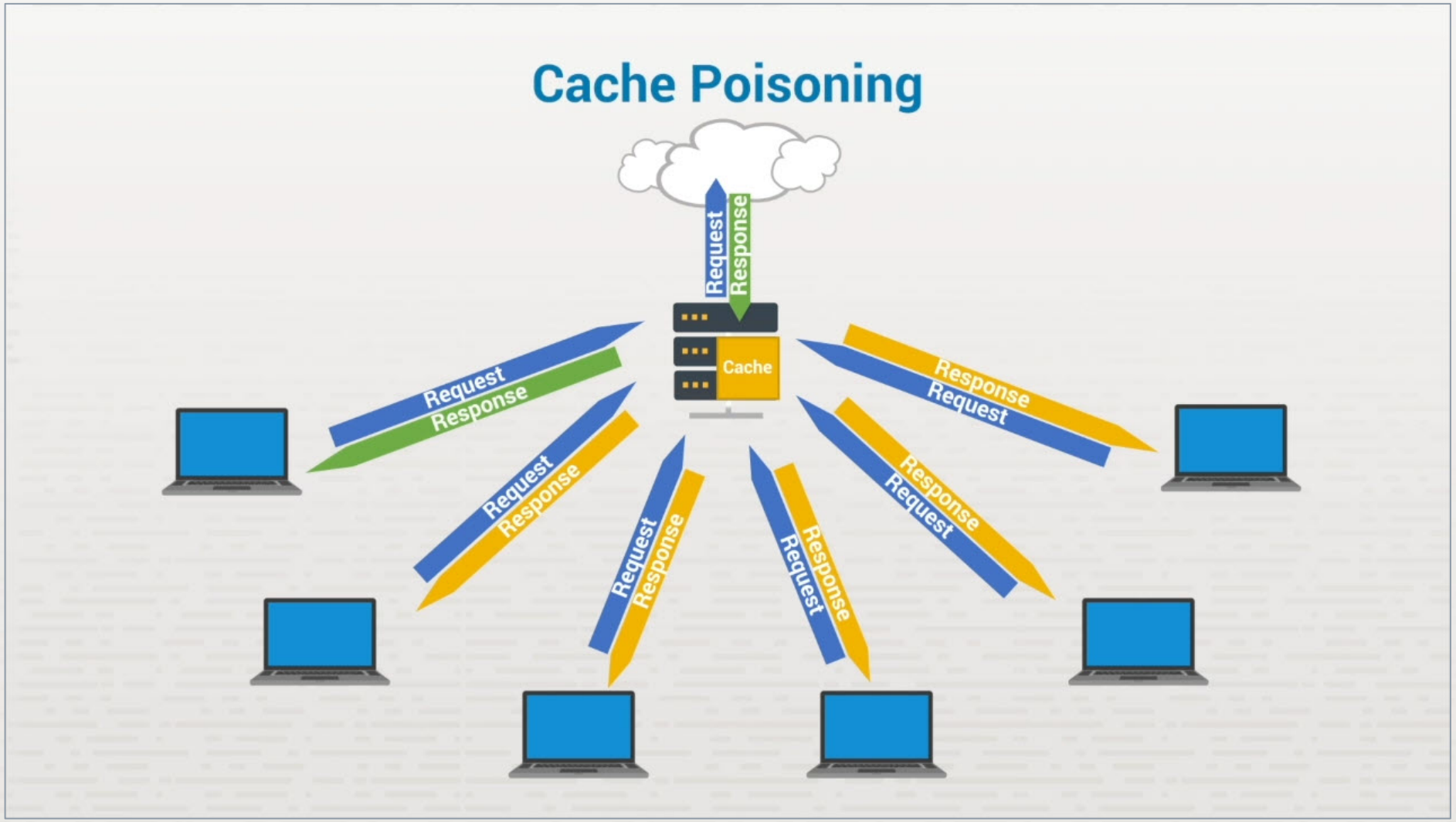


# Header Manipulation

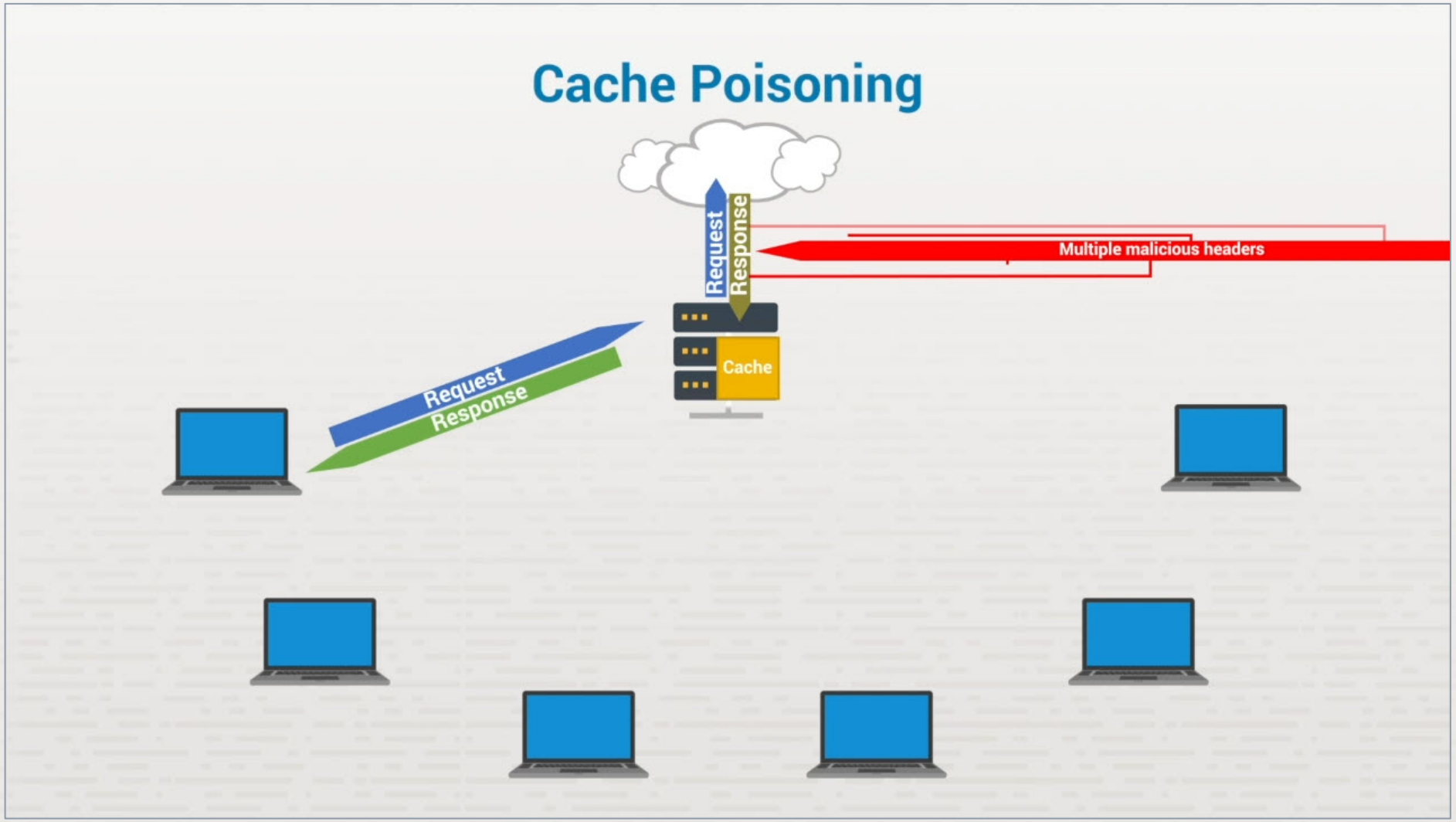
## HTTP Response Splitting

- Adds CR (\r) & LF (\n) into the header
- Example:
  - Index.html\r\nHTTP\LI 200 OK\r\nMALICIOUS CODE

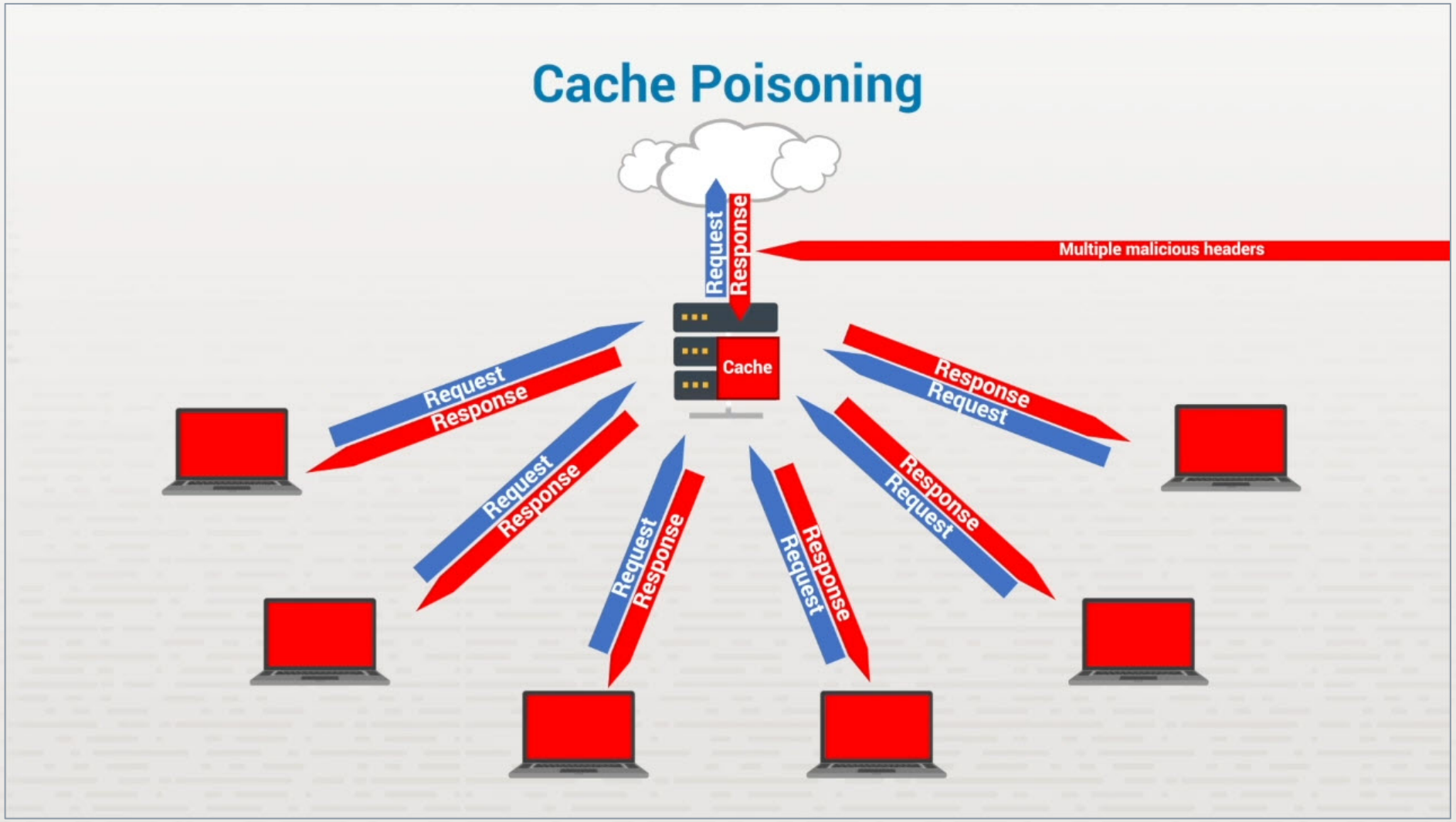
# Header Manipulation



# Header Manipulation



# Header Manipulation



# Zero Day Attacks

# Zero Day Attacks

- Web browser
  - Everyone has one
- Email attachments

# Zero Day Application Attacks

## Unknown Vulnerability Management Process

- **Analyze**
  - Attack vectors
- **Test**
  - Fuzz testing
- **Report**
- **Mitigate**
  - Develop a fix

# Zero Day Defense

- Automatic updates
- Network firewall
- Host firewall
- Use IDS or IPS

# Summary

- Zero day attacks
- Find vulnerabilities
- Defense tactics

# Client-Side Attacks

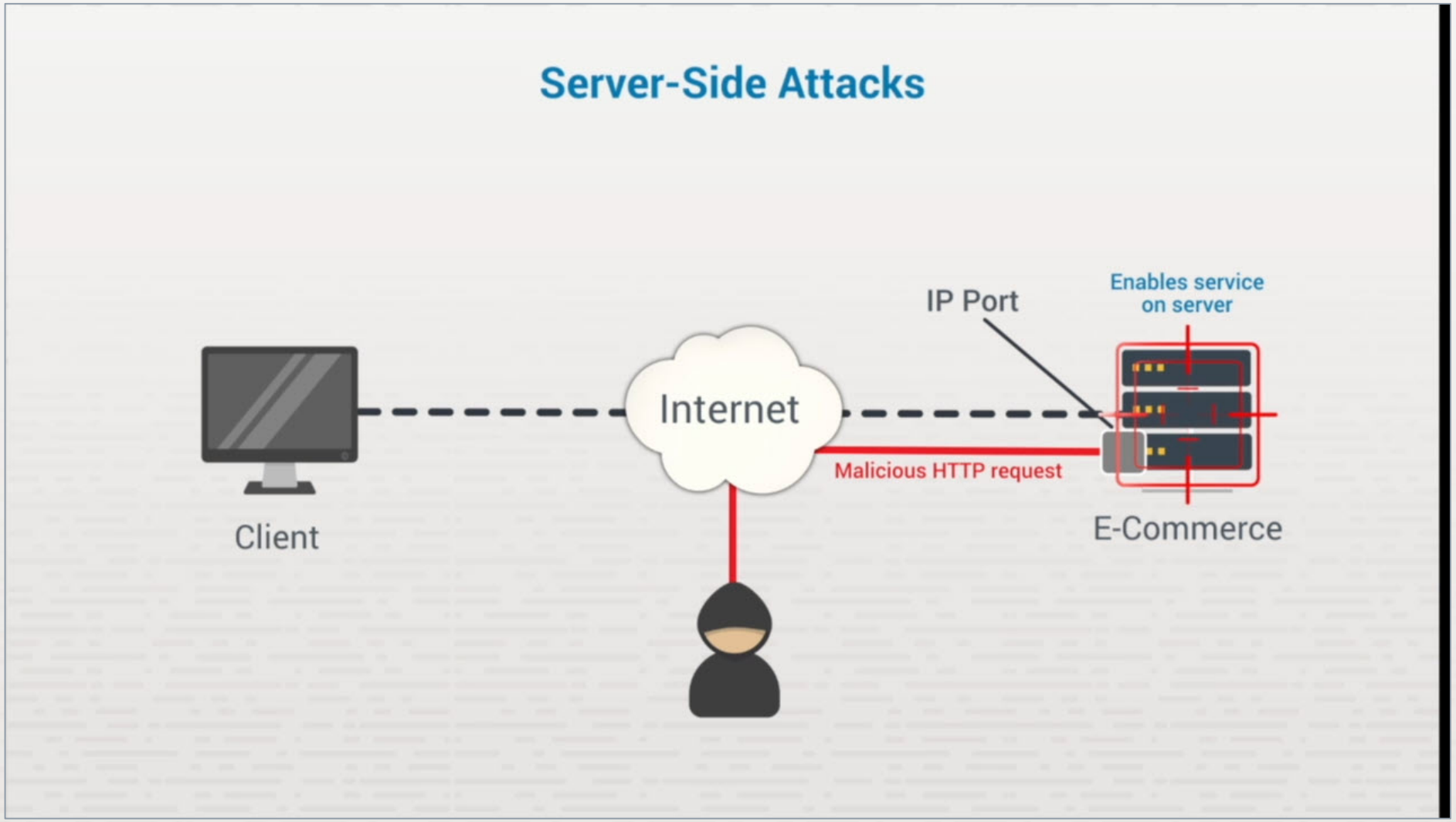
# Servers

- Assets
  - Databases
  - Financial information
  - More

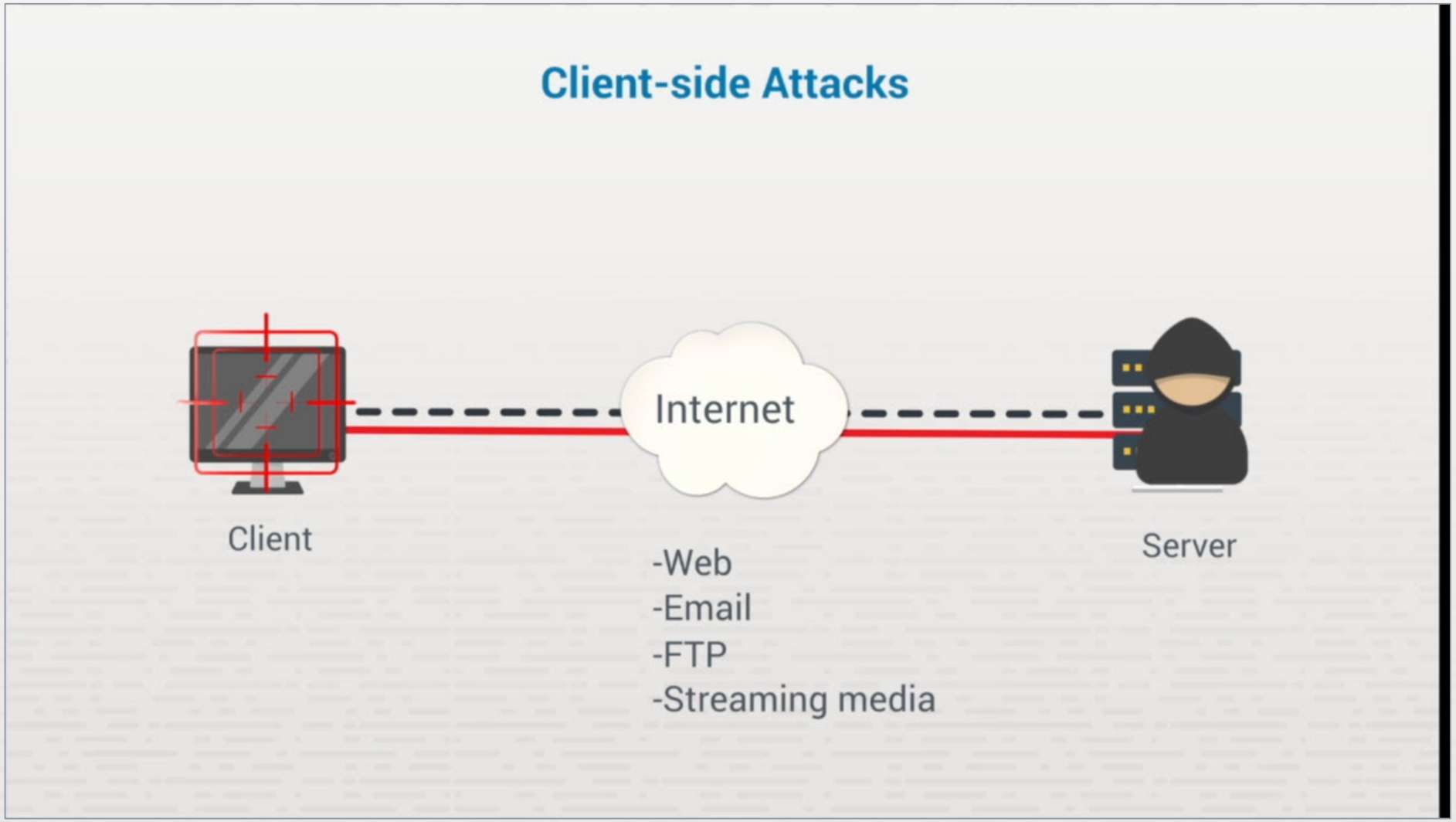
# Clients

- Fewer protections
- Personal information
- Privilege escalation

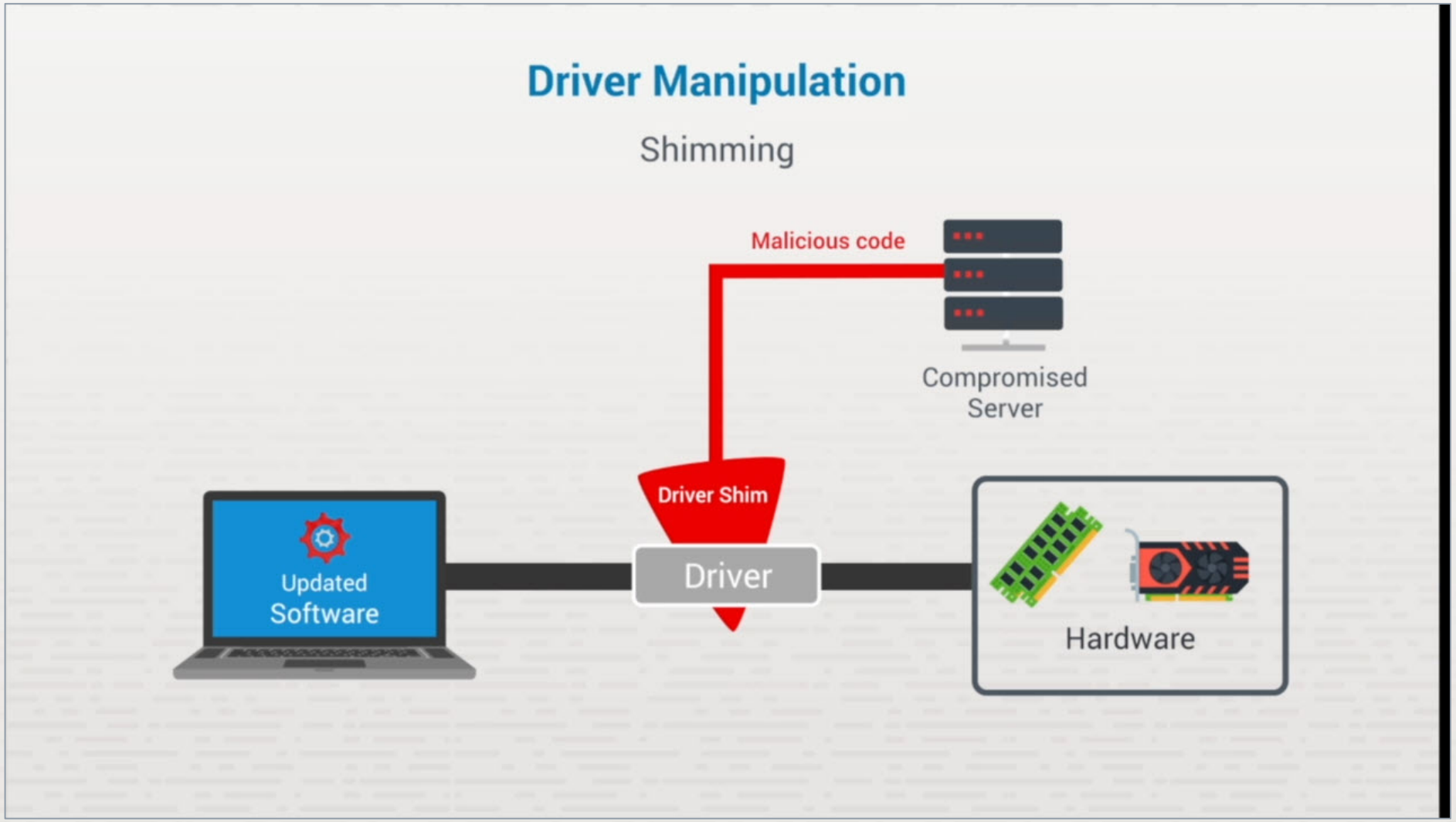
# Client-Side Attacks



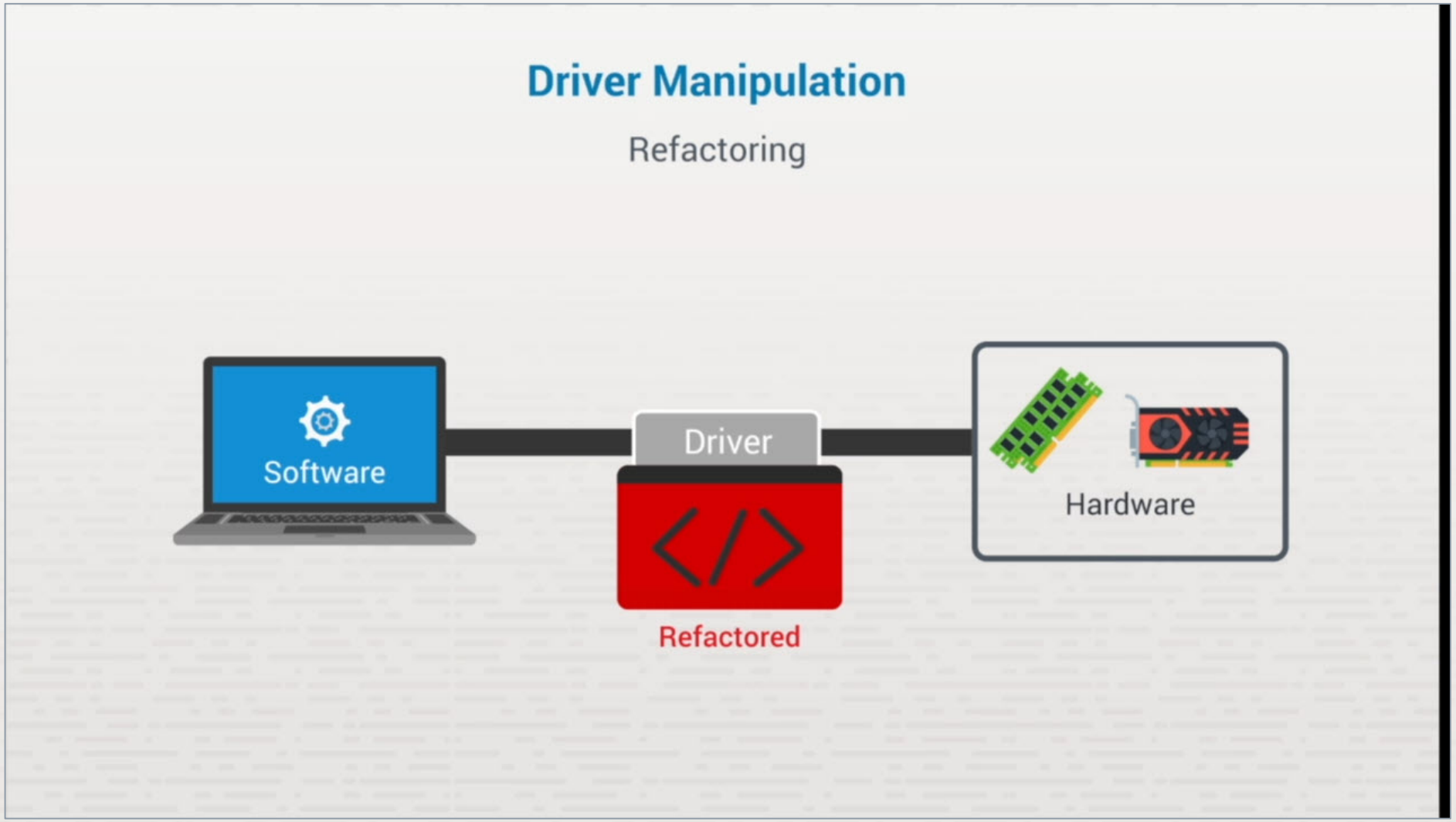
# Client-Side Attacks



# Client-Side Attacks



# Client-Side Attacks



# Client-Side Attack Mitigation

- Host-based firewall
- Network firewall
- Intrusion detection system (IDS)
- Block malicious URLs
- Back up often

# Summary

- Client-side vs server-side
- Forms of attack
- Driver manipulation
- Mitigation

# Web Browser Threats

# Web Browser Threats

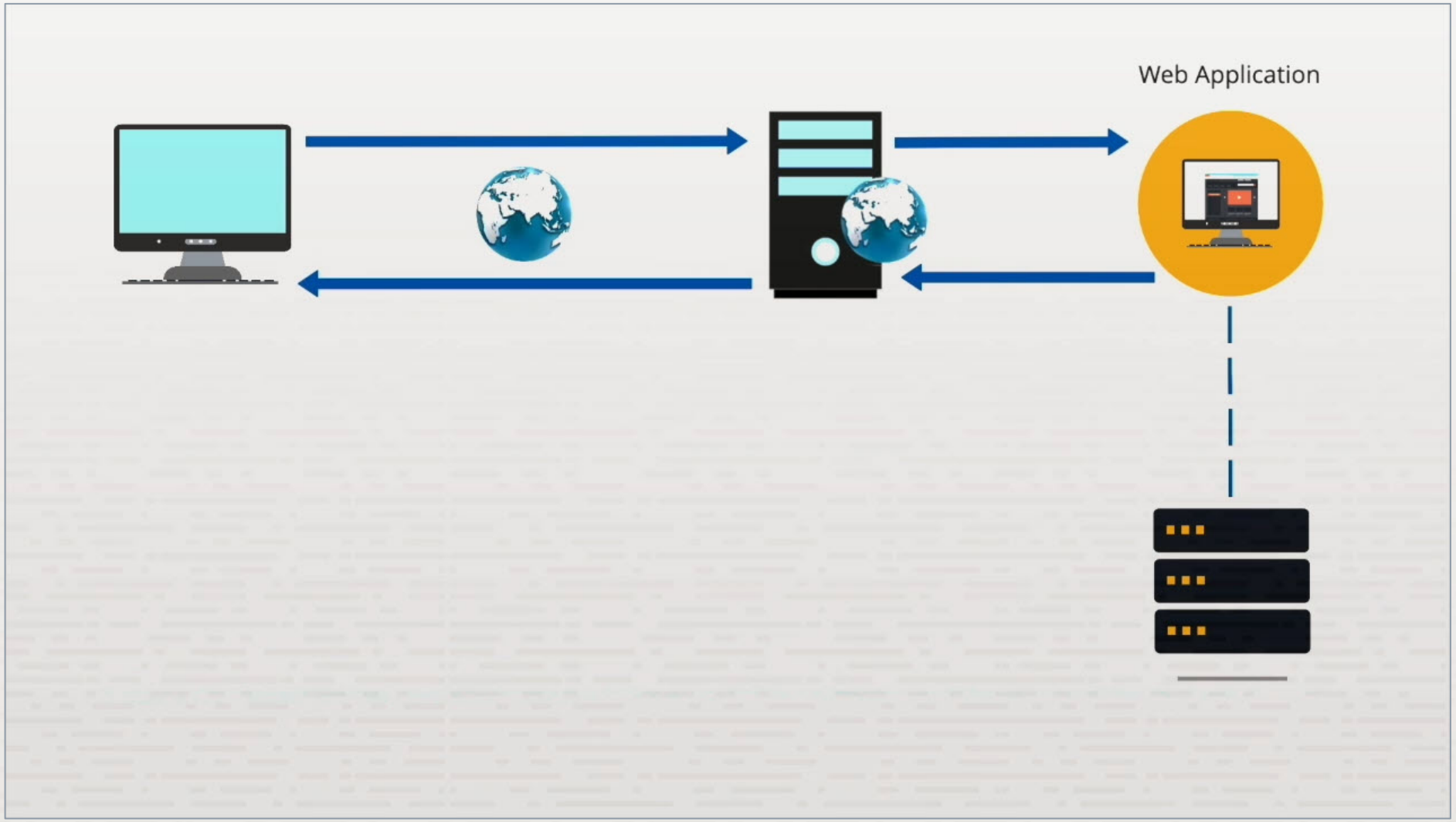
- Weak configuration settings
- XSS
- Third-party plug-ins/add-ons
- Injection flaws (SQL, XML, LDAP)
- Password vaults
- Outdated browser version
- Third-party cookies

# Threat Mitigation

- Antivirus/anti -malware
- Ad blockers
- Disable Javascript
- Implement DNSSEC
- Use HTTPS
- Verify certificate information

# SQL Injections

# SQL Injections



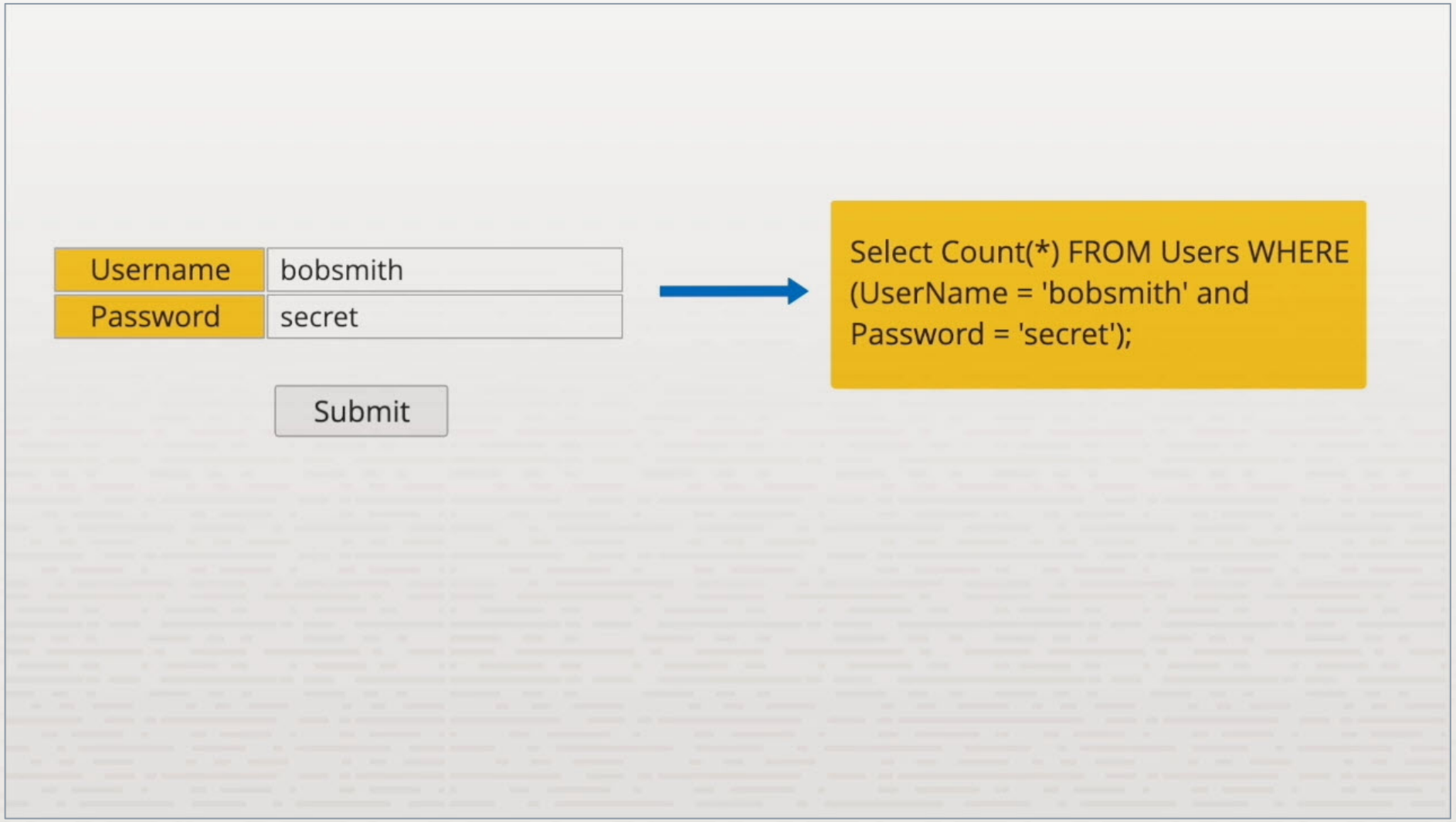
# Database Types

- Relational database
- Distributed database
- Object-oriented database

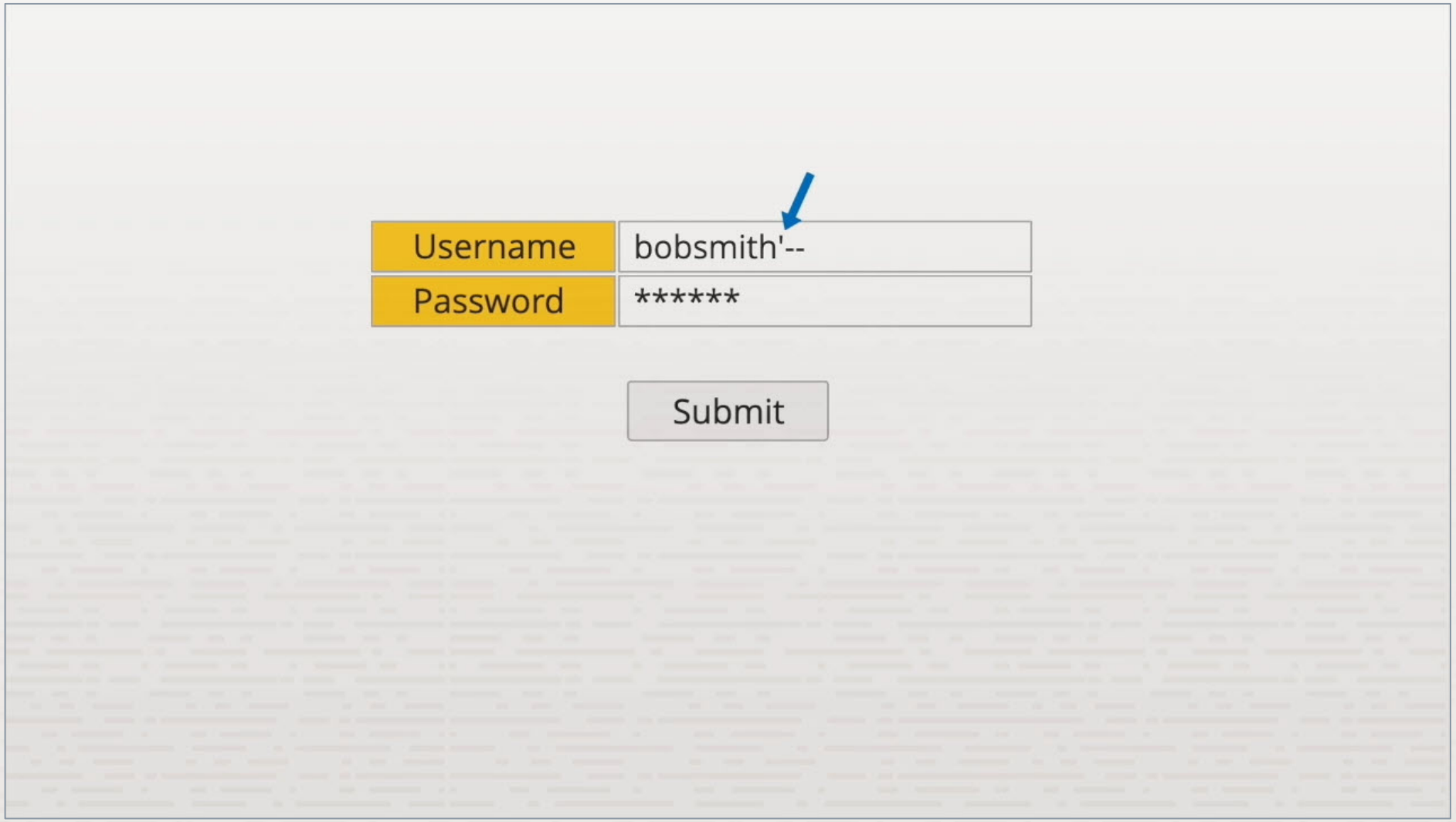
# SQL Injections

- Target flaws in web applications
- Exploit input vulnerabilities
- Use existing code

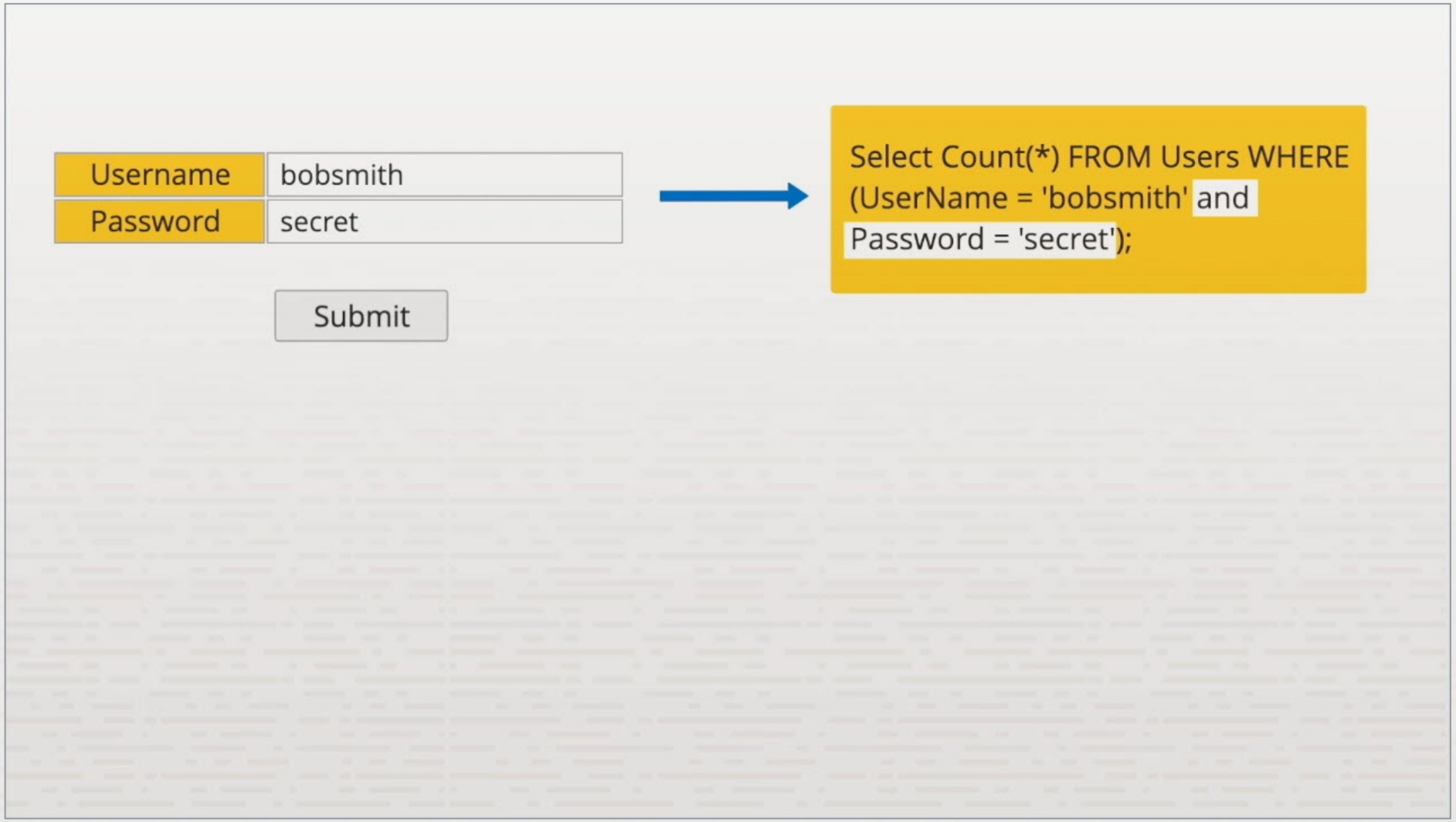
# SQL Injections



# SQL Injections



# SQL Injections



# SQL Injection Types

- Authentication bypass
- Information disclosure
- Compromised data integrity
- Remote code execution

# In-Class Practice

Do the following labs:

- 10.3.10 Clear the Browser Cache
- 10.3.15 Perform an SQL Injection Attack

# Class Discussion

- What are the common forms of web application attacks?
- How do you mitigate replay attacks?
- What are some methods to prevent driver manipulation?
- How does SSL stripping work?

10.4 SECURING DATA AND APPLICATIONS

# Application Development and Security

TESTOUT SECURITY PRO

# Section Skill Overview

- Harden applications on Linux.
- Implement Data Execution Preventions (DEP).
- Implement application whitelisting with AppLocker.

# Key Terms

- Normalization
- Stored procedures
- Code obfuscation
- Code reuse
- Dead code
- Memory management
- Third-party libraries
- Software Development Kits (SDKs)
- Data exposure
- Fuzz testing
- Code signing

# Key Definitions

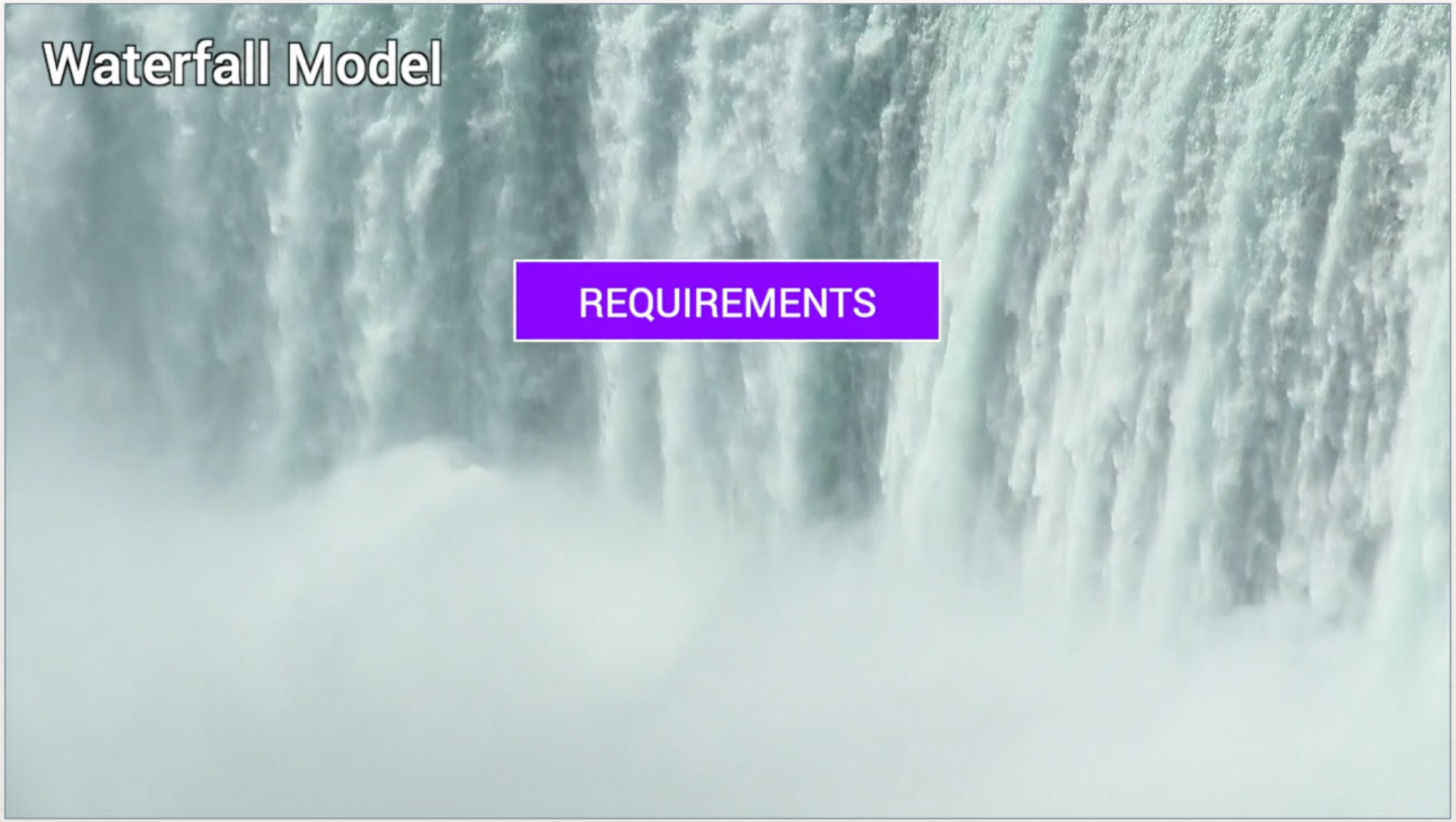
- **Normalization:** Data reorganized in a relational database to eliminate redundancy by having all data stored in one place and storing all related items together.
- **Stored procedures:** One or more database statements stored as a group in a database's data dictionary, which when called, executes all the statements in the collection.
- **Code obfuscation:** The deliberate act of creating source or machine code that is difficult for humans to understand. In other words, the code is camouflaged.
- **Code reuse:** Using the same code multiple times.
- **Dead code:** Code that is non-executable at run-time, or source code in a program that is executed but is not used in any other computation.

# Key Definitions

- **Memory management:** A resource management process applied to computer memory. It allows your computer system to assign portions of memory, called blocks, to various running programs to optimize overall system performance.
- **Third-party libraries:** A library where the code is not maintained in-house.
- **Software Development Kits (SDKs):** A set of software development tools that can be installed as one unit.
- **Data exposure:** Unintended exposure of personal and confidential data.
- **Fuzz testing:** A software testing technique that exposes security problems by providing invalid, unexpected, or random data to the inputs of an application.
- **Code signing:** The process of digitally signing (encrypting) executables and scripts to confirm the software author and guarantee that the code has not been altered or corrupted since it was signed.

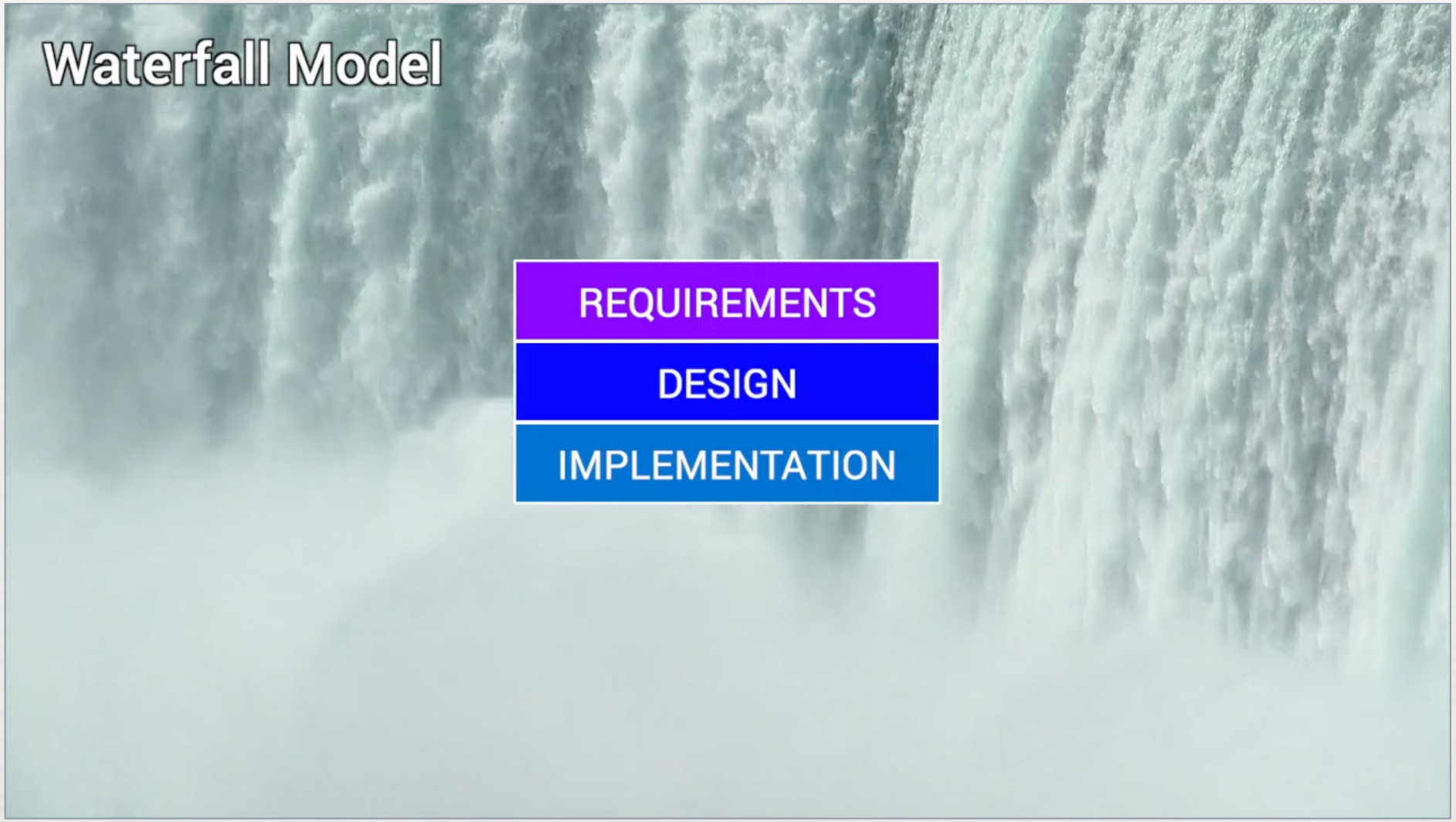
# Development Life Cycle

# Development Life Cycle

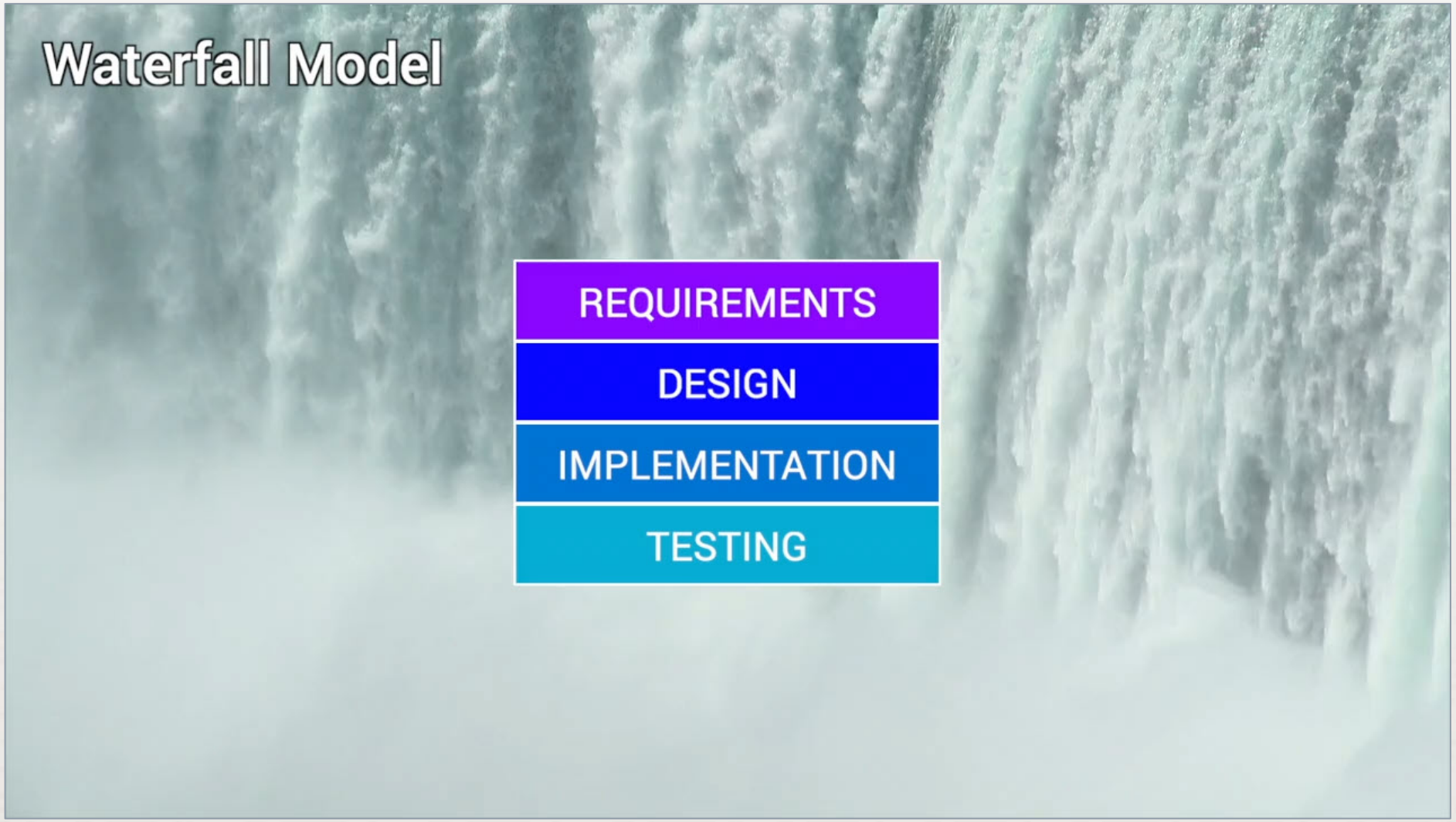


# Development Life Cycle

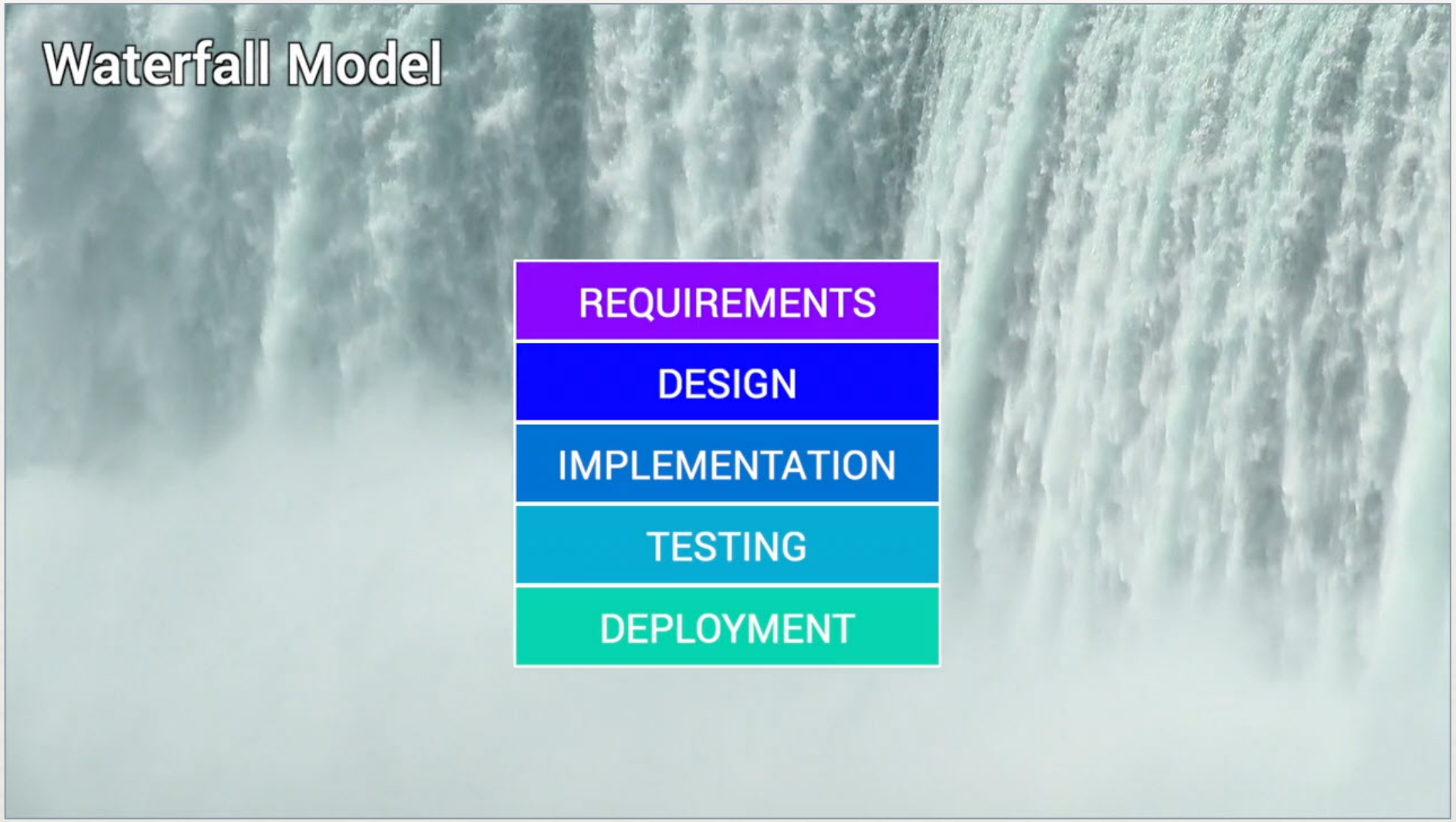
# Development Life Cycle



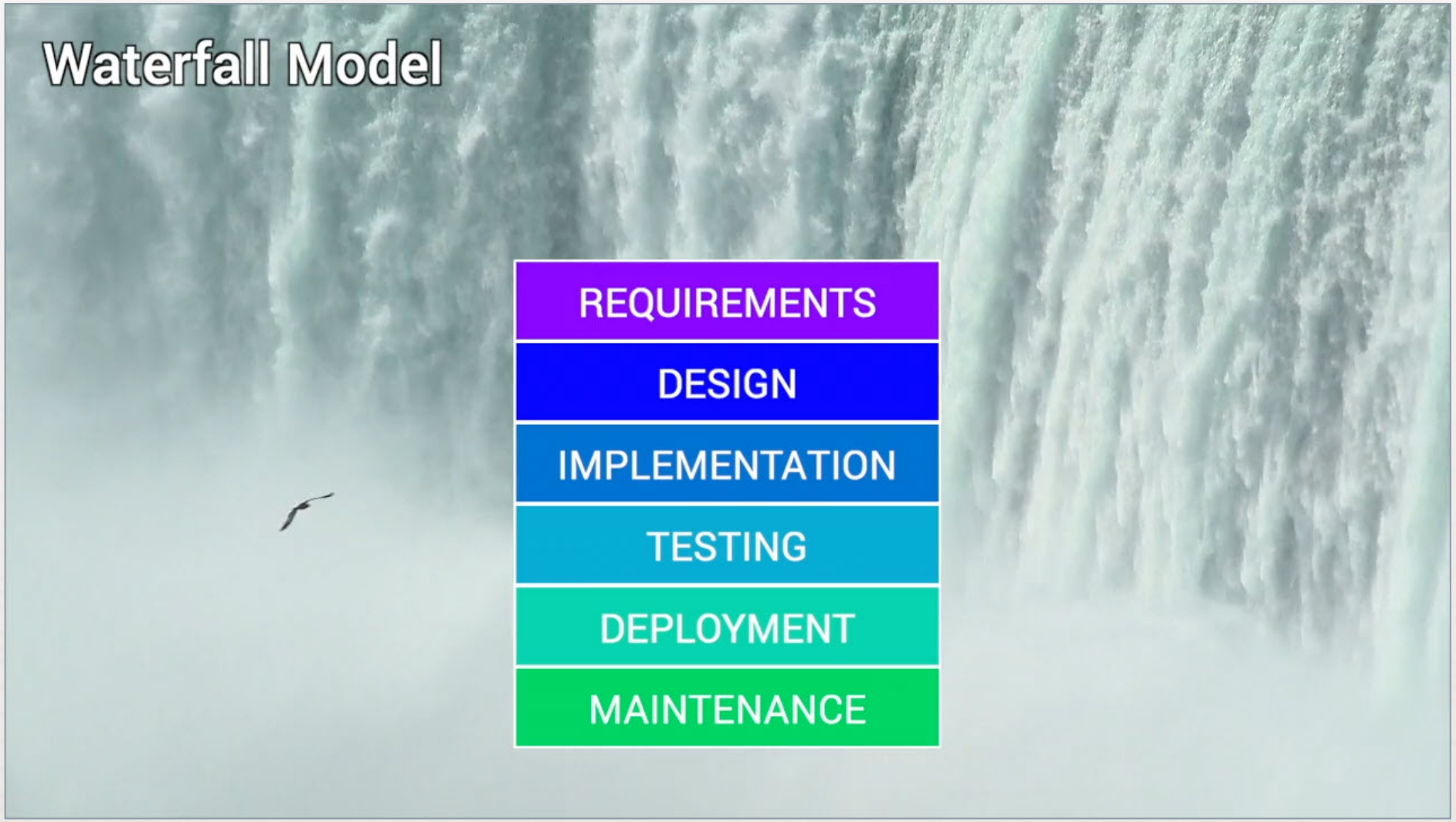
# Development Life Cycle



# Development Life Cycle



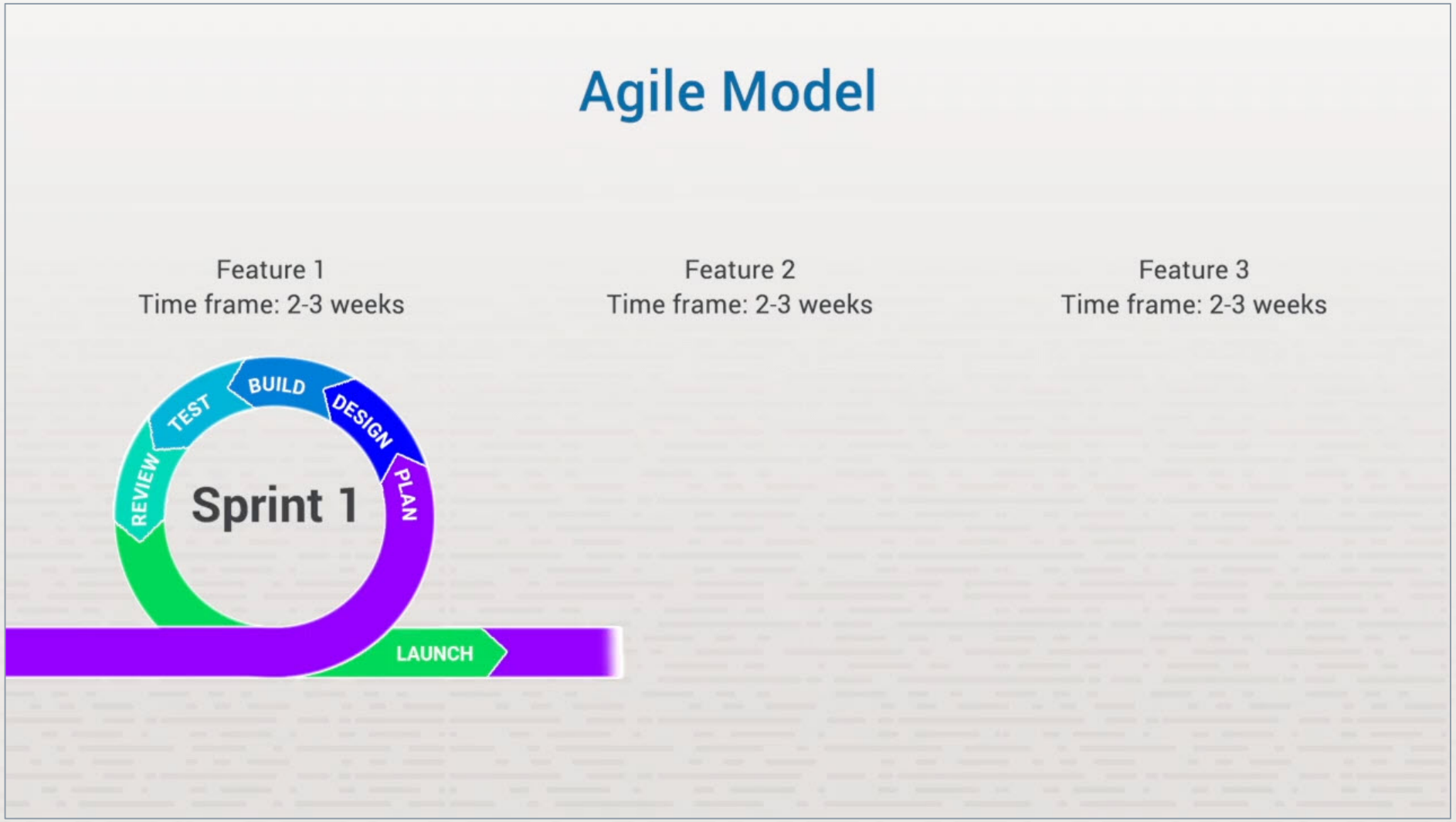
# Development Life Cycle



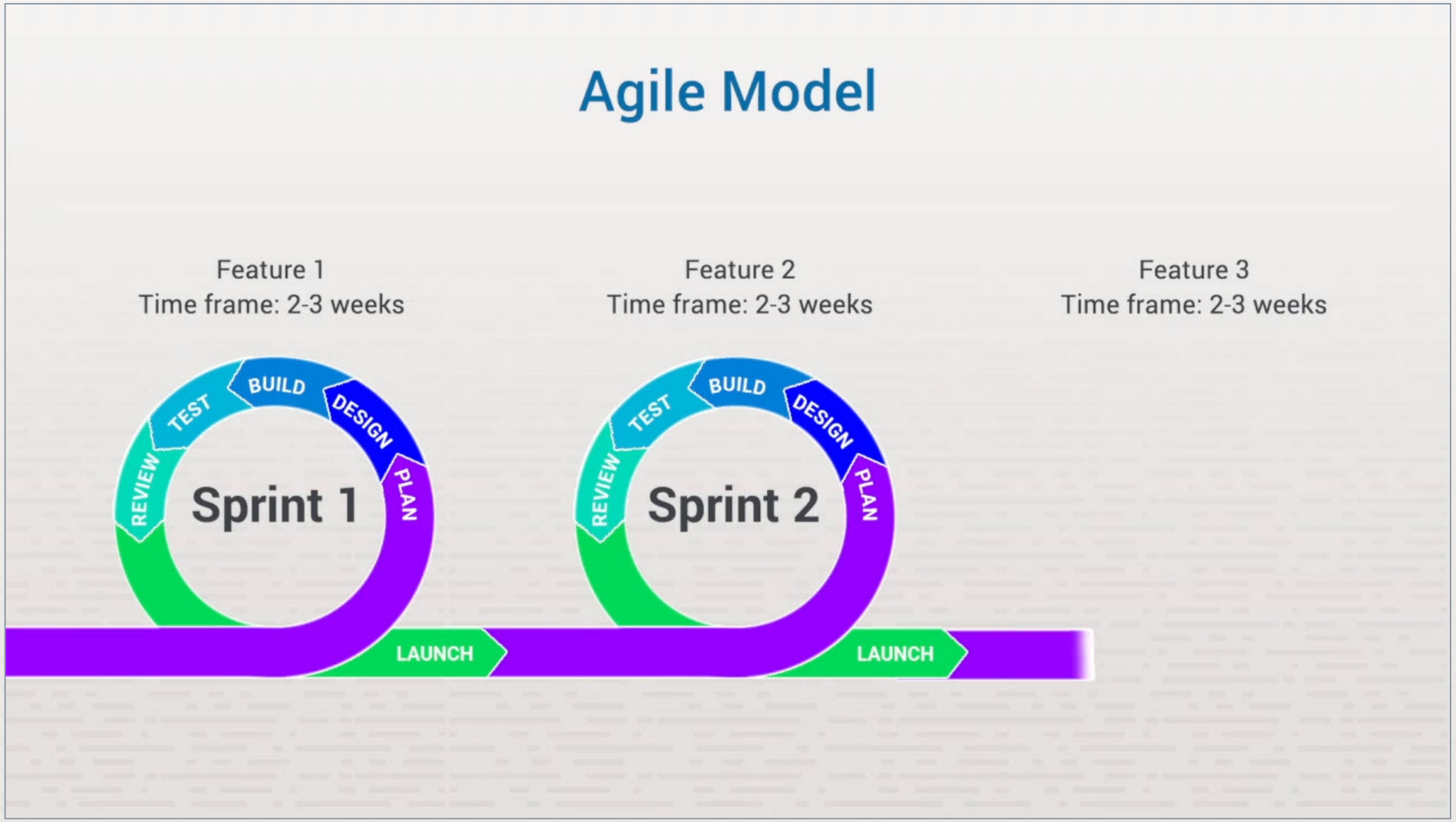
# Agile Model

- Continuously changing
- Never-ending versions
- Bug fixes
- Enhancements

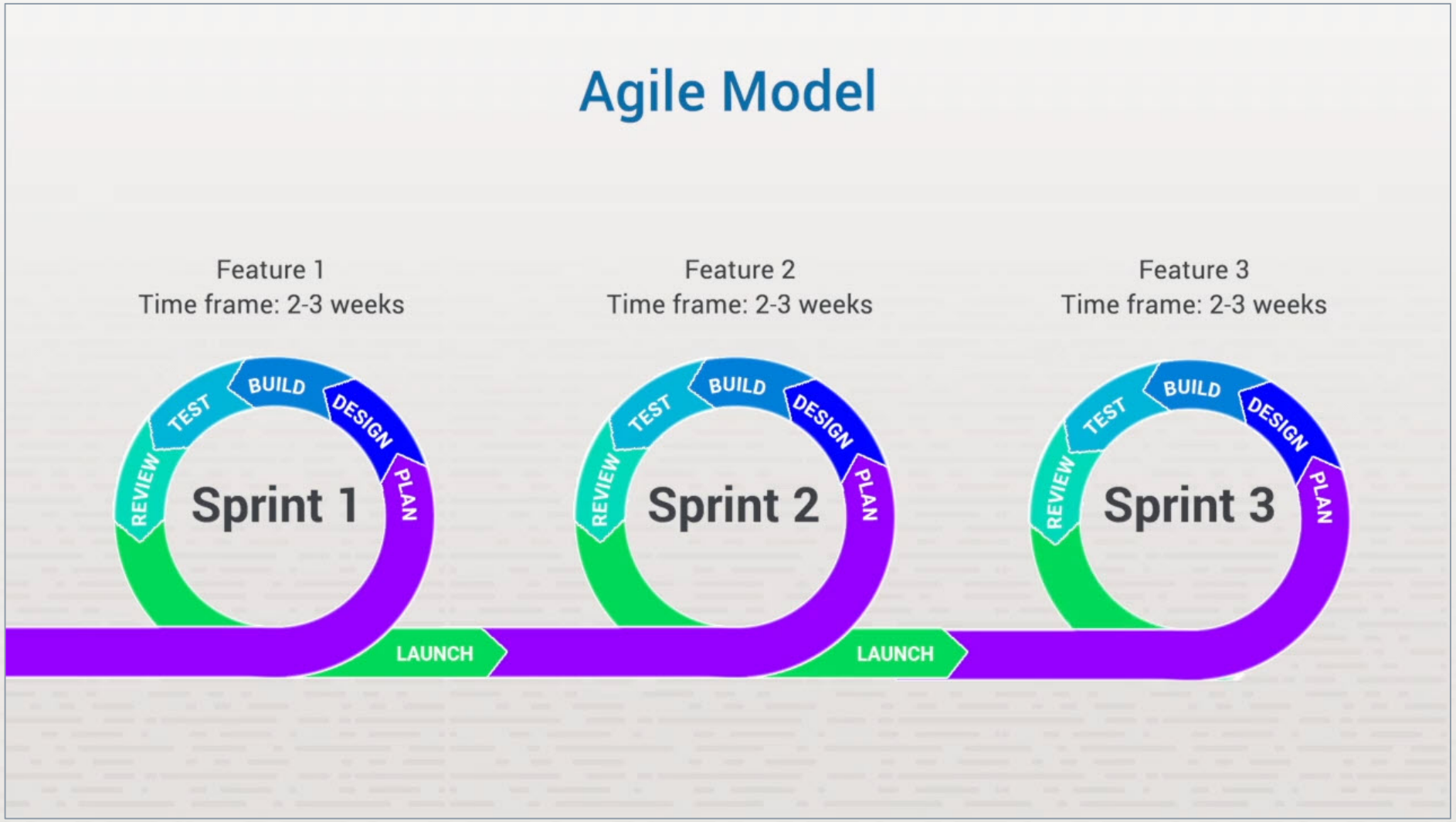
# Development Life Cycle



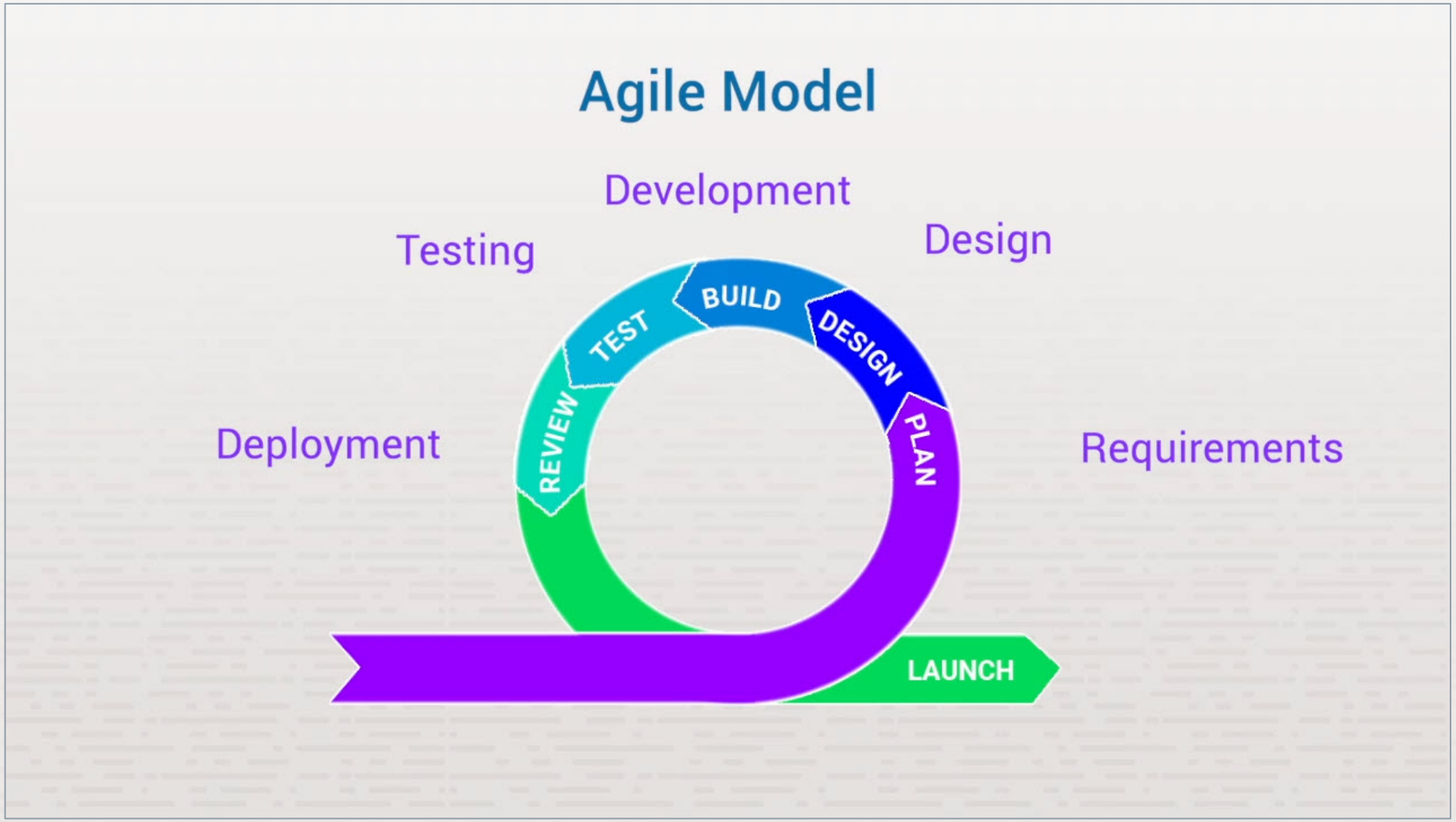
# Development Life Cycle



# Development Life Cycle



# Development Life Cycle



# Coding Errors

- Compile
  - During build/compilation stage
  - Prevents application from running
- Runtime
  - During execution
  - Software cannot resolve

# Development Life Cycle

## Application Security Testing

| Static Application Security Testing (SAST) | | |
|---|---|---|
| White box testing | | |
| Analyzes source code, binaries, byte code during compilation | | |
| Identifies exact cause of coding problem | | |
| Is language specific | | |
| Can run continuously | | |
| Can be widely applied | | |
| High percent of false positives | | |
| Limited in types of vulnerability detection | | |

# Development Life Cycle

## Application Security Testing

| Static Application Security Testing (SAST) | Dynamic Application Security Testing (DAST) | |
|---|---|---|
| White box testing | Black box testing | |
| Analyzes source code, binaries, byte code during compilation | Scans during runtime | |
| Identifies exact cause of coding problem | Tests from outside | |
| Is language specific | Is not language specific | |
| Can run continuously | Fewer false positive rates | |
| Can be widely applied | Hard to automate | |
| High percent of false positives | Can't pinpoint cause of flaw | |
| Limited in types of vulnerability detection | Can take a week | |

# Development Life Cycle

## Application Security Testing

| Static Application Security Testing (SAST) | Dynamic Application Security Testing (DAST) | Interactive Application Security Testing (IAST) |
|---|---|---|
| White box testing | Black box testing | **Passive IAST** |
| Analyzes source code, binaries, byte code during compilation | Scans during runtime | Built into SAST |
| Identifies exact cause of coding problem | Tests from outside | Use source code scanners |
| Is language specific | Is not language specific | Works in runtime |
| Can run continuously | Fewer false positive rates | **Active IAST** |
| Can be widely applied | Hard to automate | Access interpreters and compilers |
| High percent of false positives | Can't pinpoint cause of flaw | Identifies precise problem in runtime |
| Limited in types of vulnerability detection | Can take a week | Speeds up testing time |

# Summary

- Waterfall development model
- Agile development model
- Coding errors
- Application security testing
  - Static
  - Dynamic
  - Interactive

# Automation and Scripting

# Automation

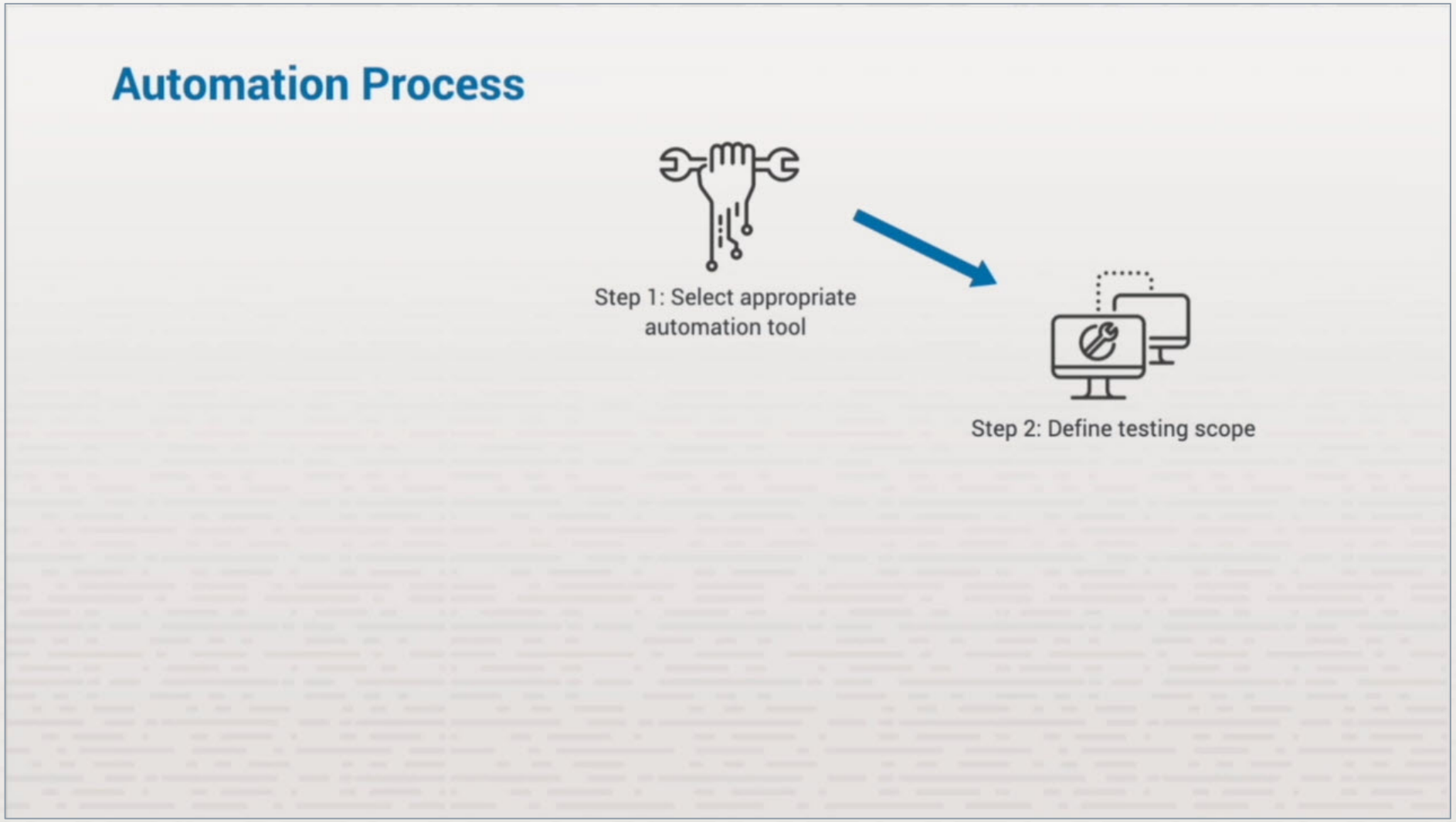
- Use automated testing
  - Repeated tests
  - Time-consuming tests
  - Complex test
- Use manual testing
  - New tests
  - Frequently changing tests

# Automation and Scripting

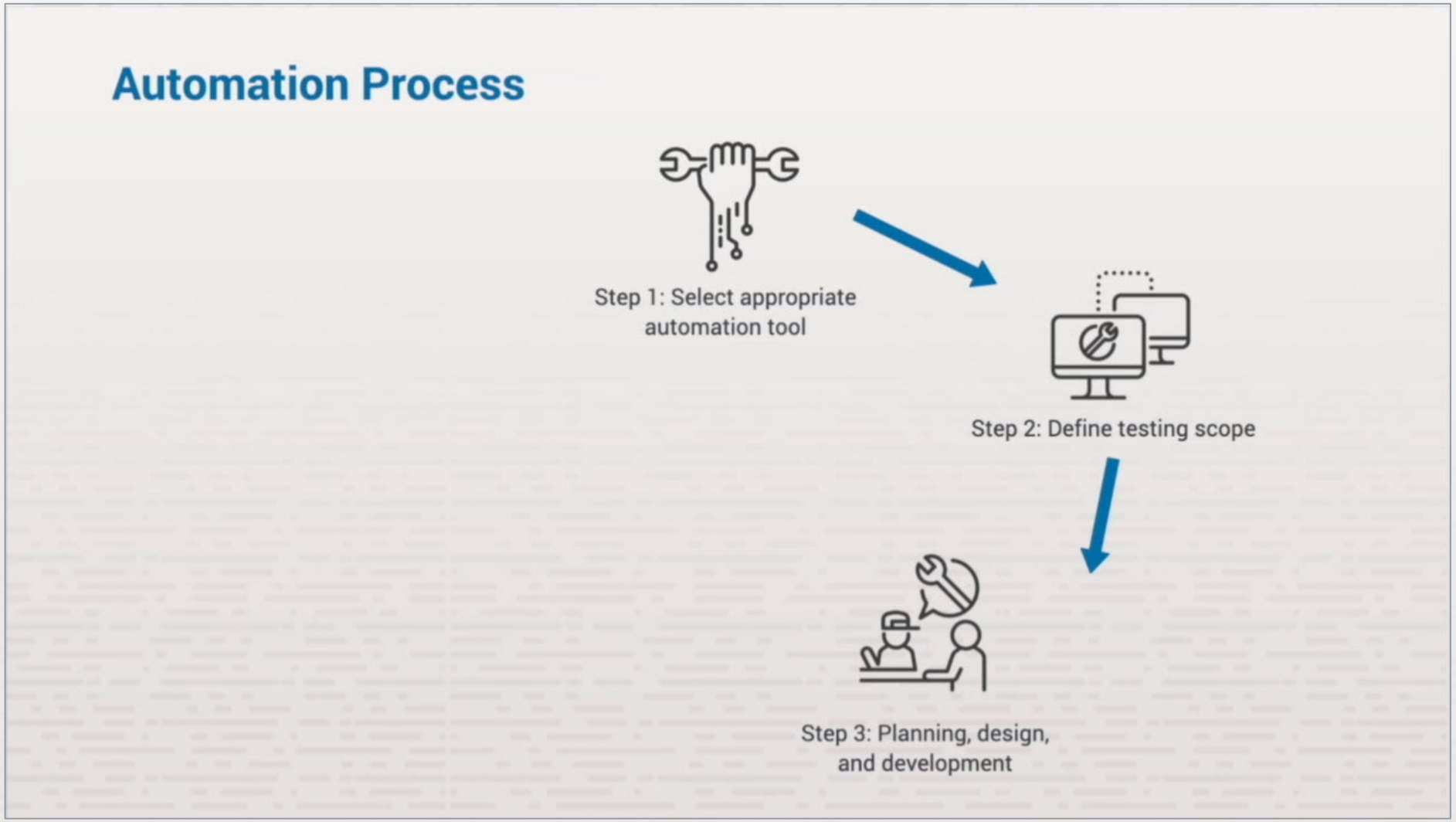
## Automation Process

Step 1: Select appropriate automation tool

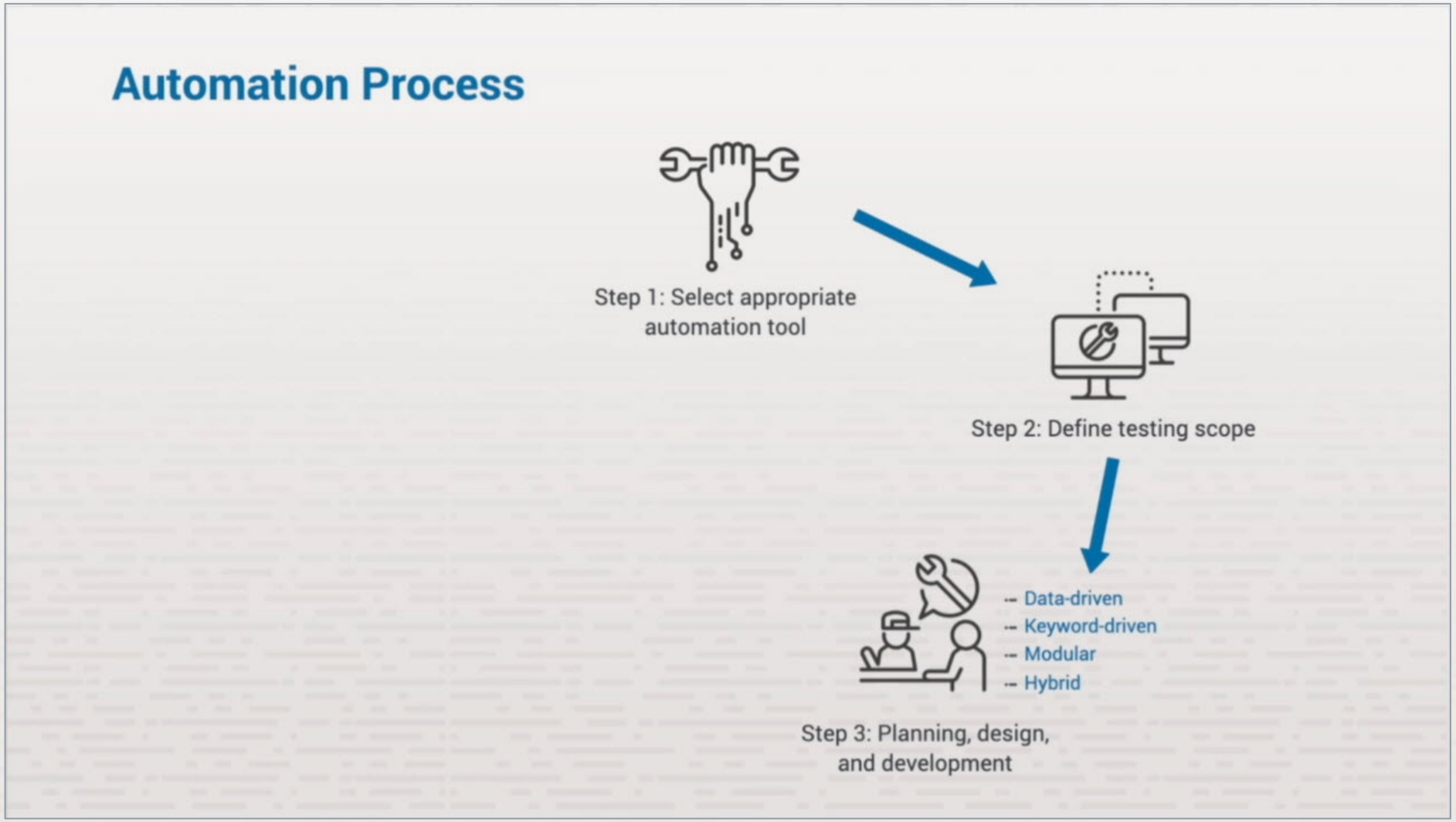
# Automation and Scripting



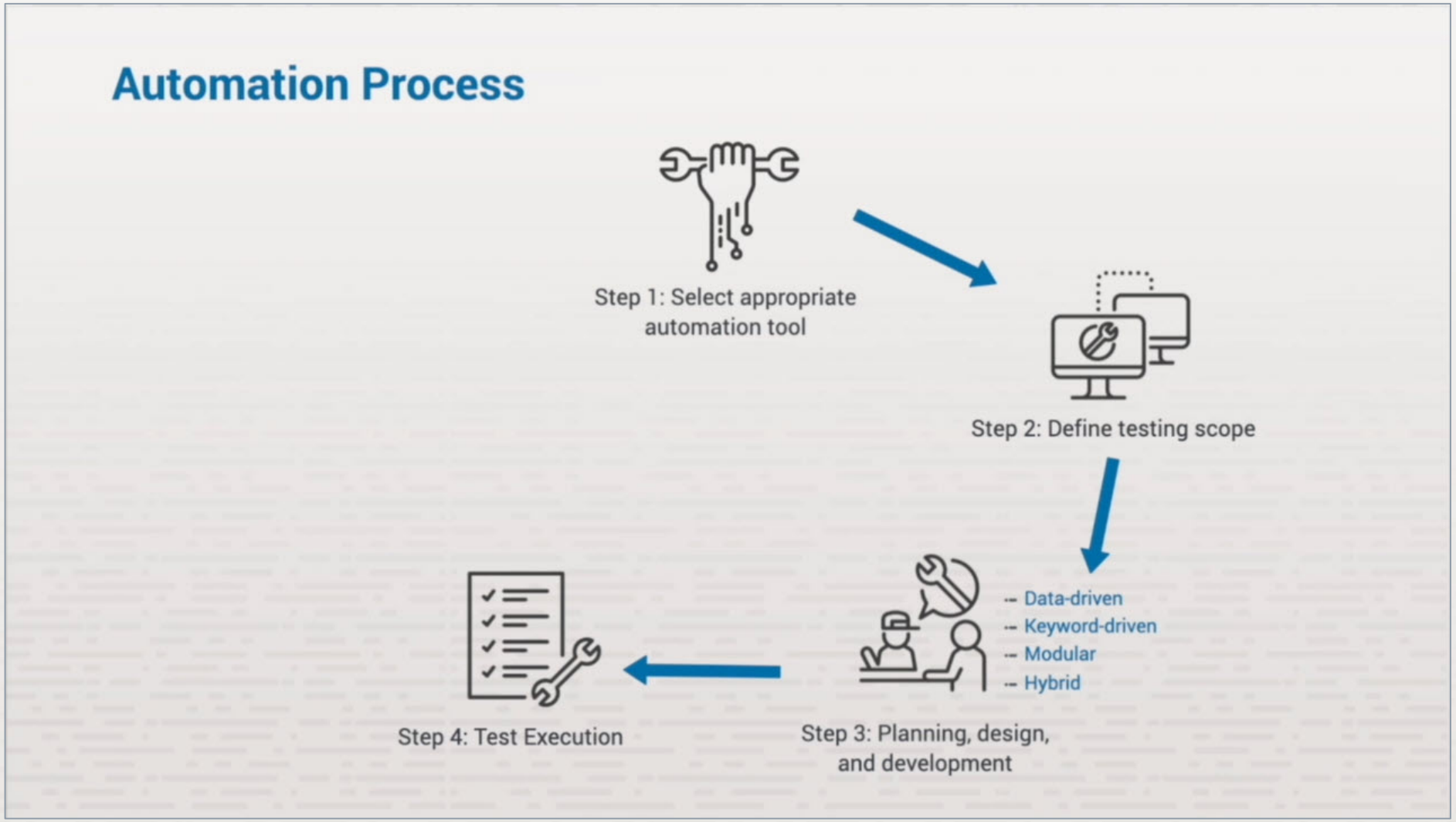
# Automation and Scripting



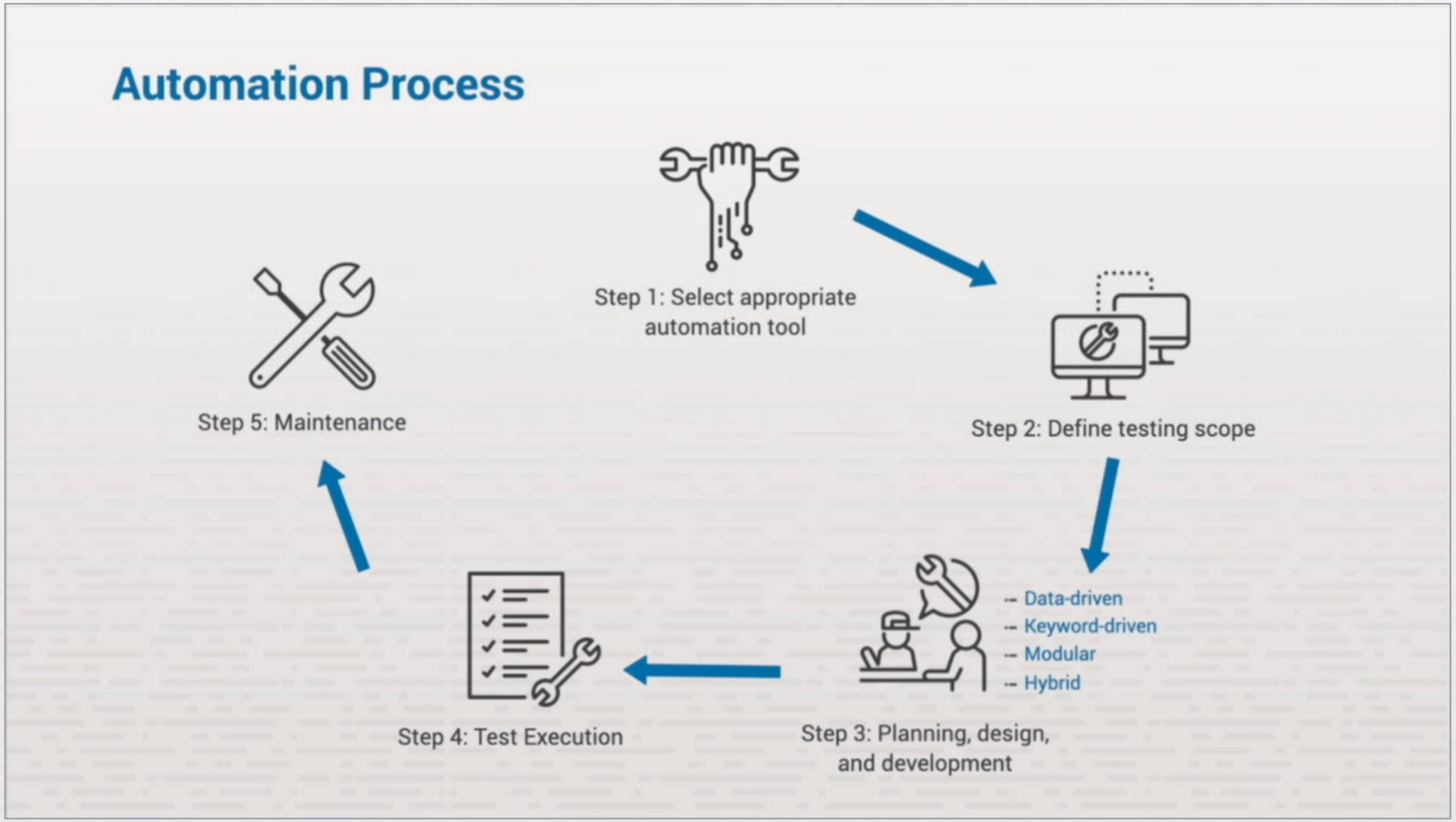
# Automation and Scripting



# Automation and Scripting



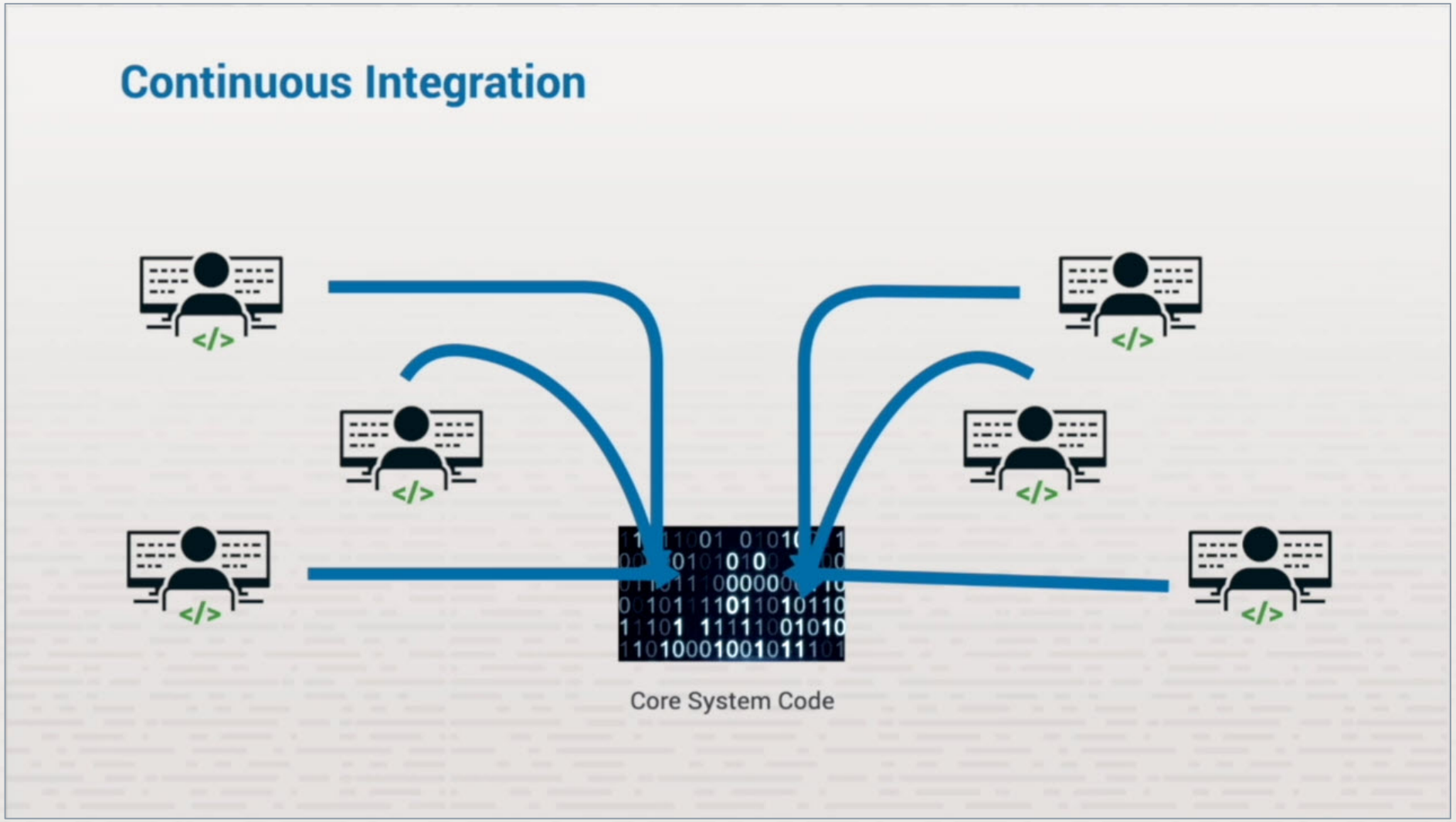
# Automation and Scripting



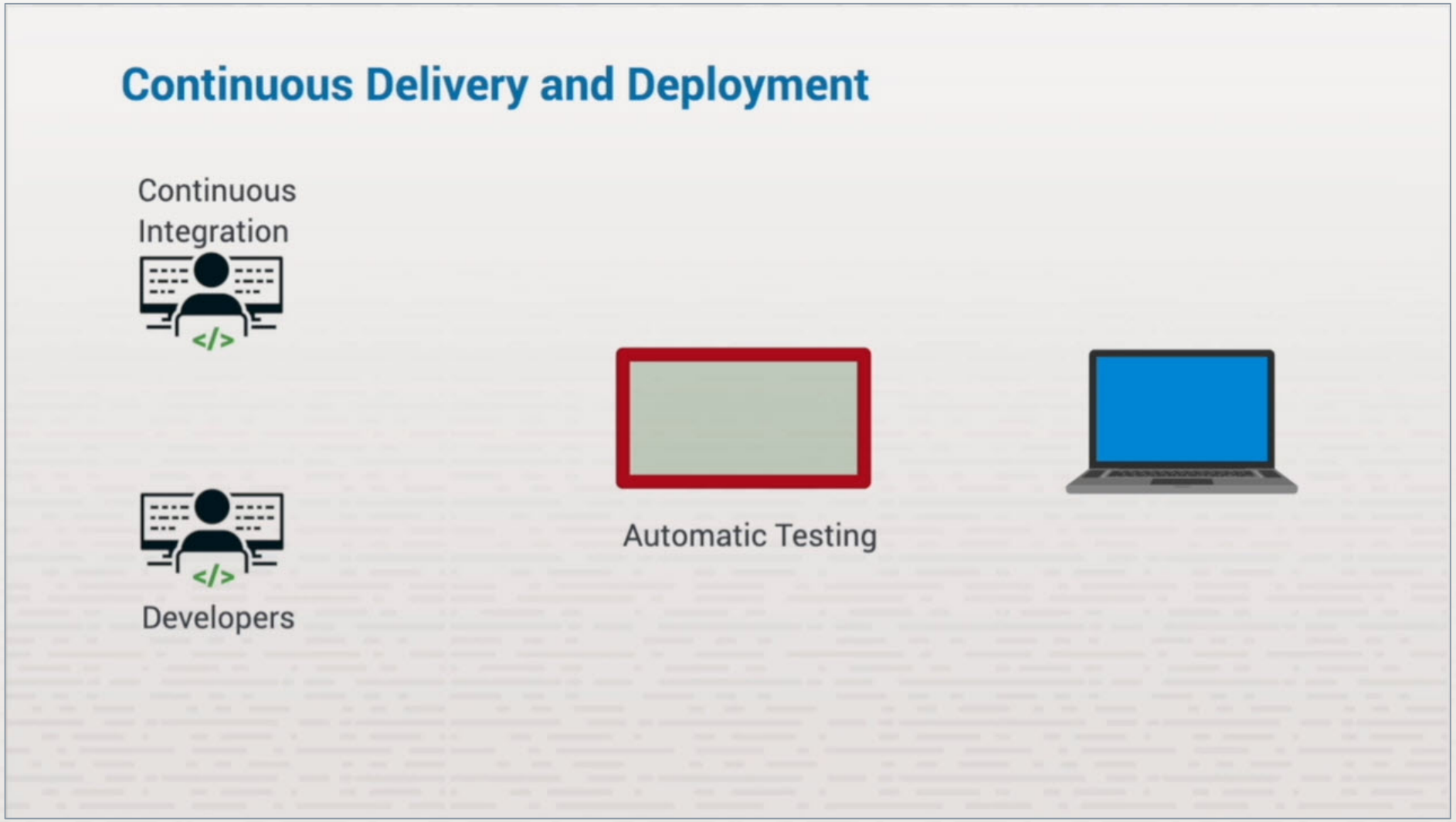
# Automated Courses of Action

- Set of automated actions
- Part of development stage
- Types
  - End-to-end
  - Atomic

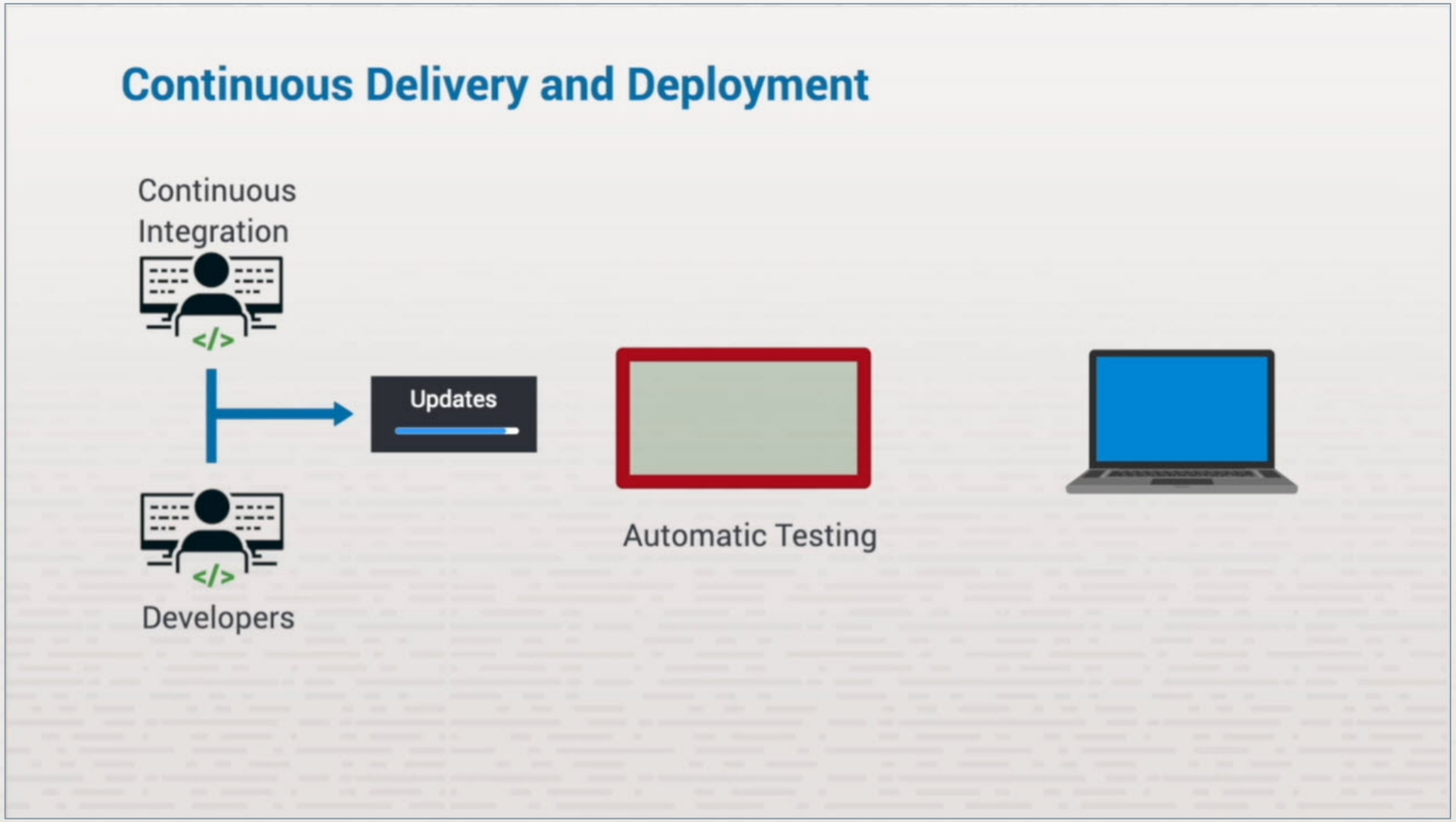
# Automation and Scripting



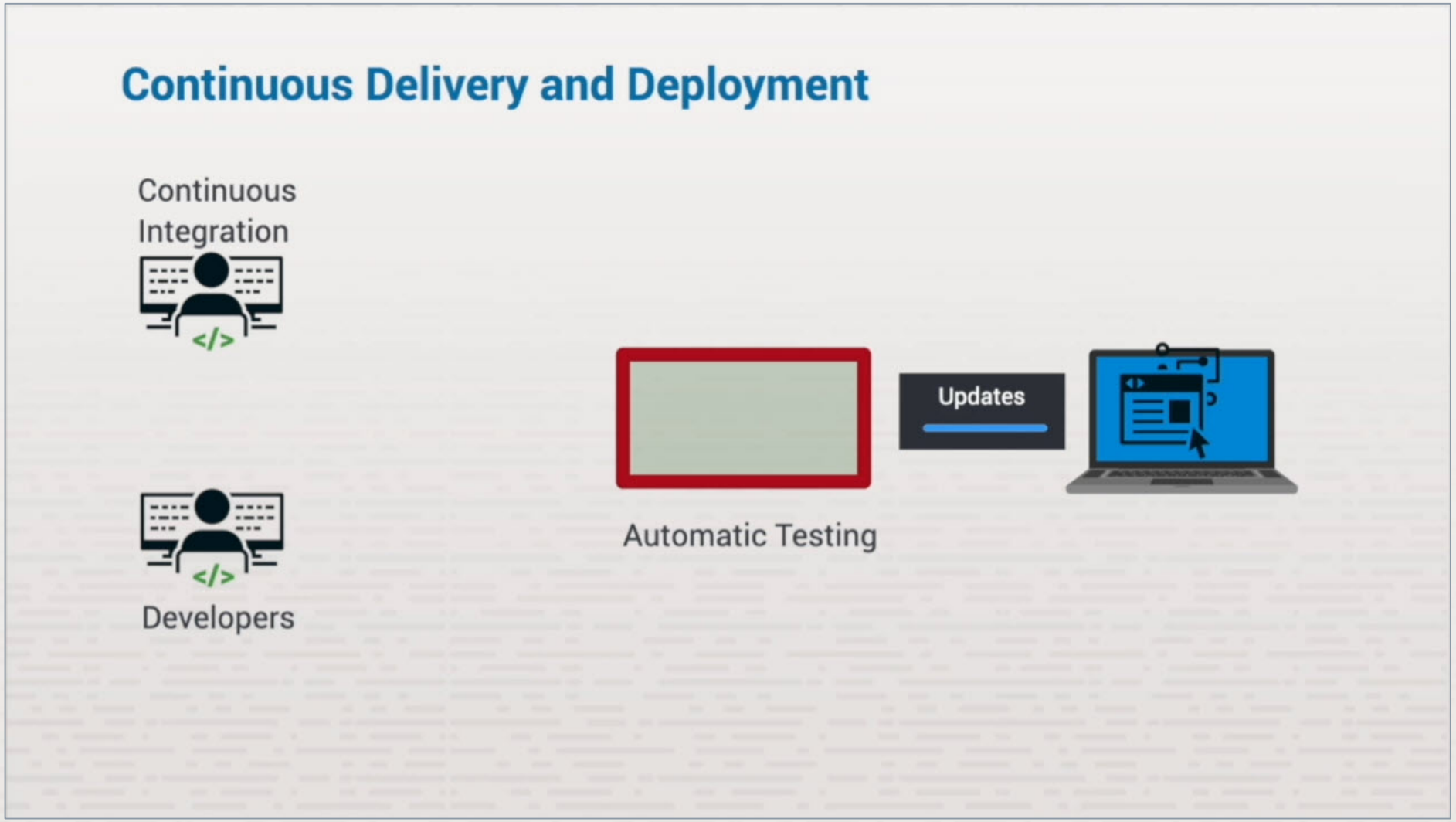
# Automation and Scripting



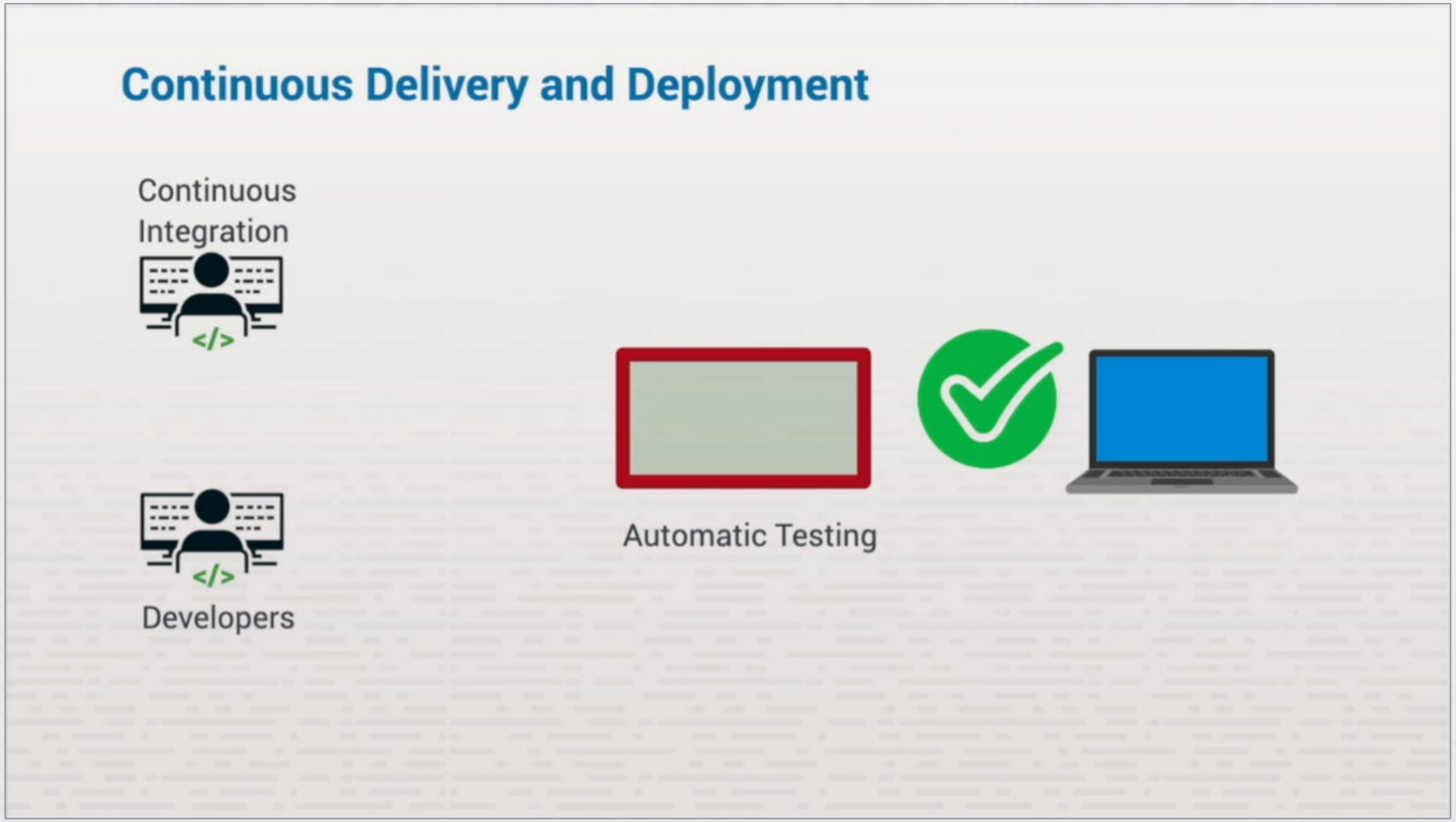
# Automation and Scripting



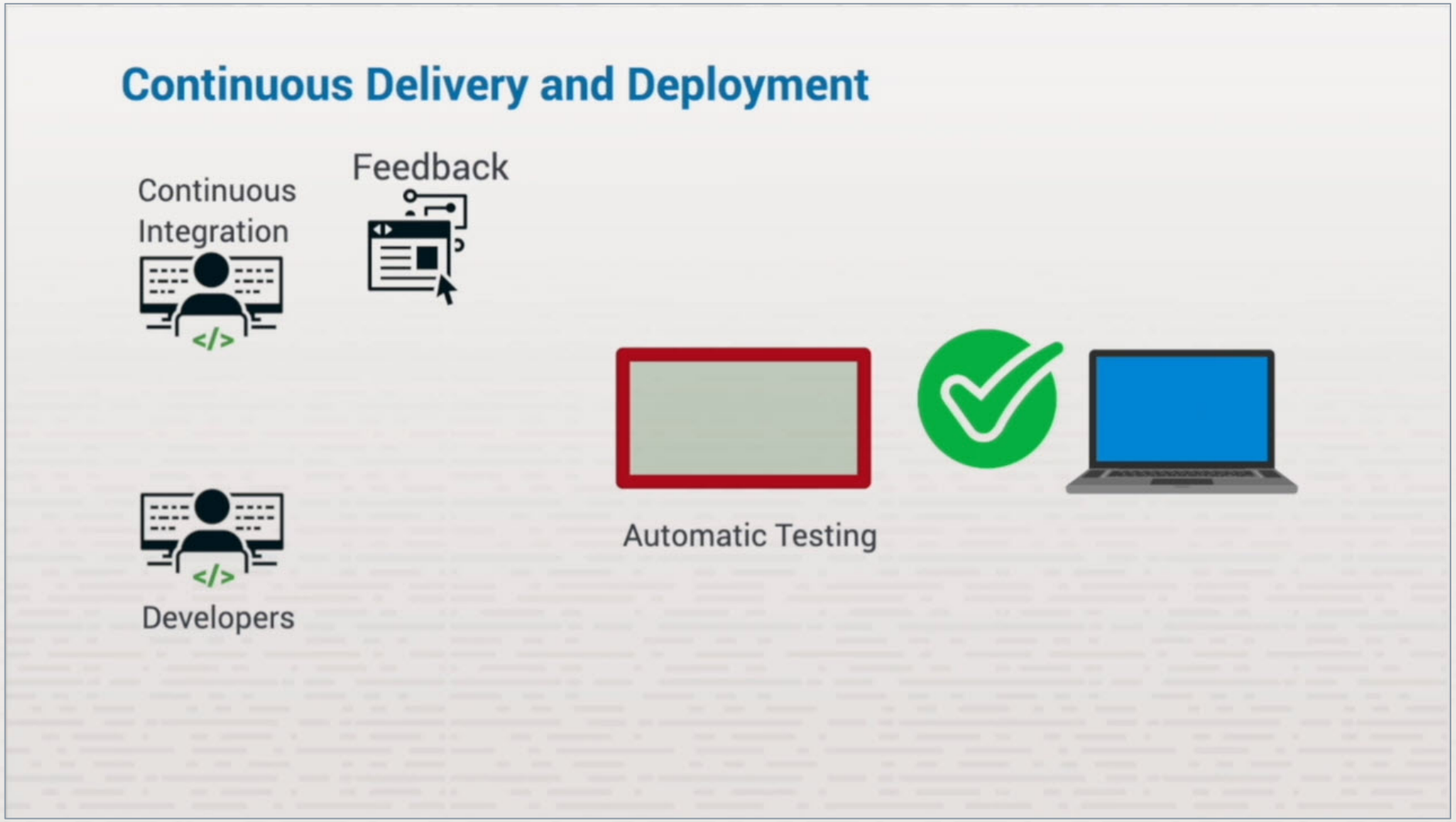
# Automation and Scripting



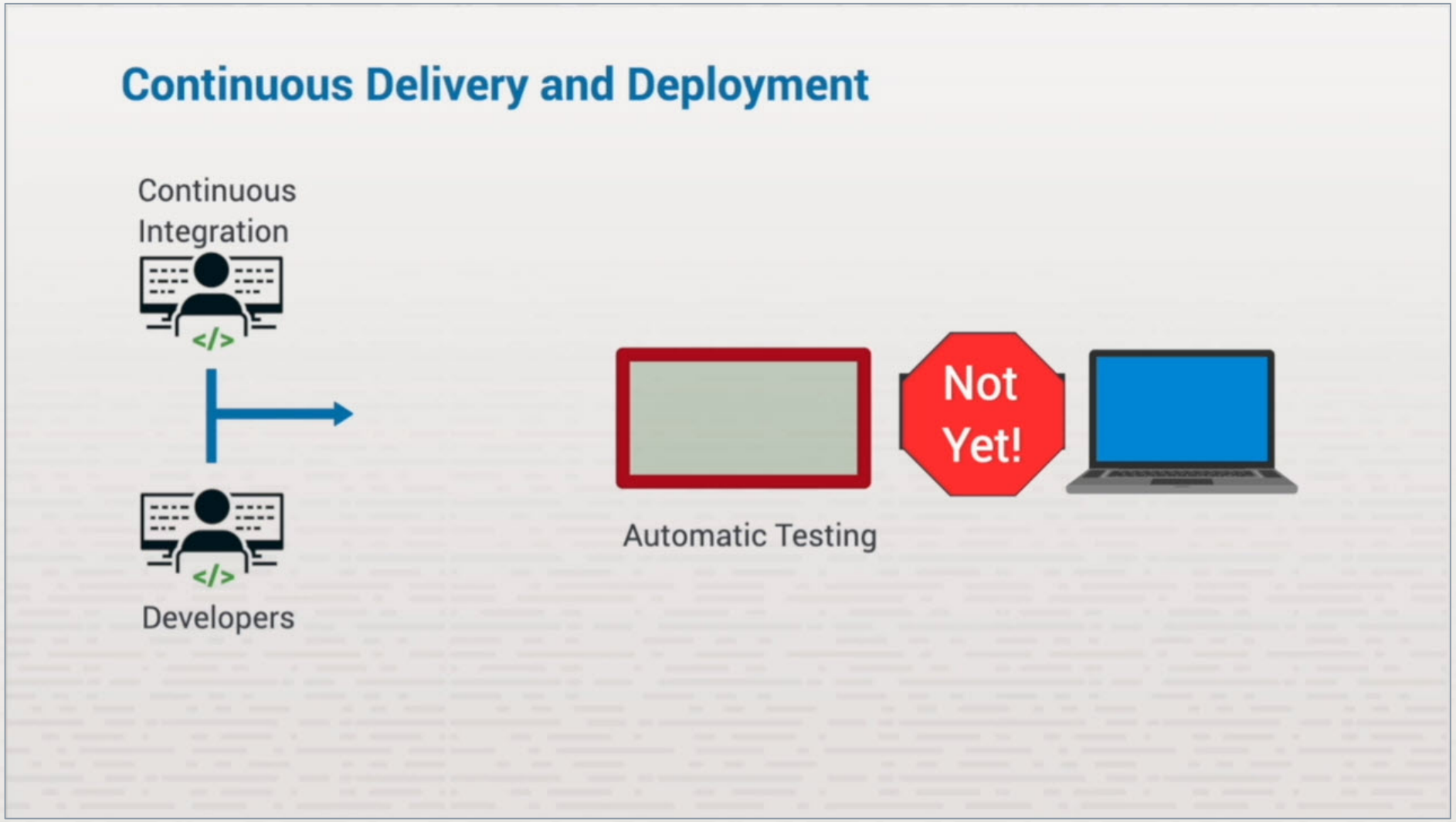
# Automation and Scripting



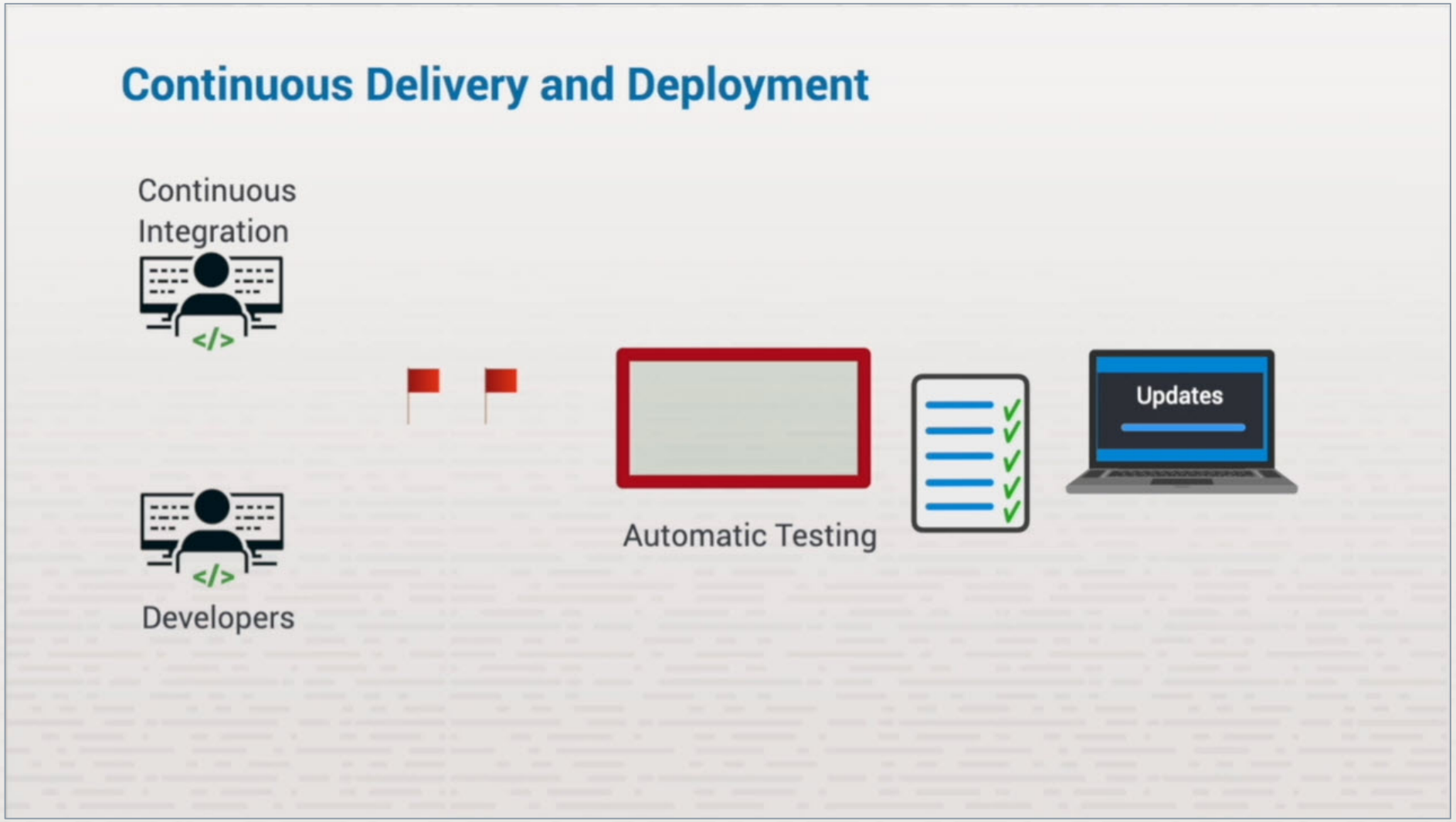
# Automation and Scripting



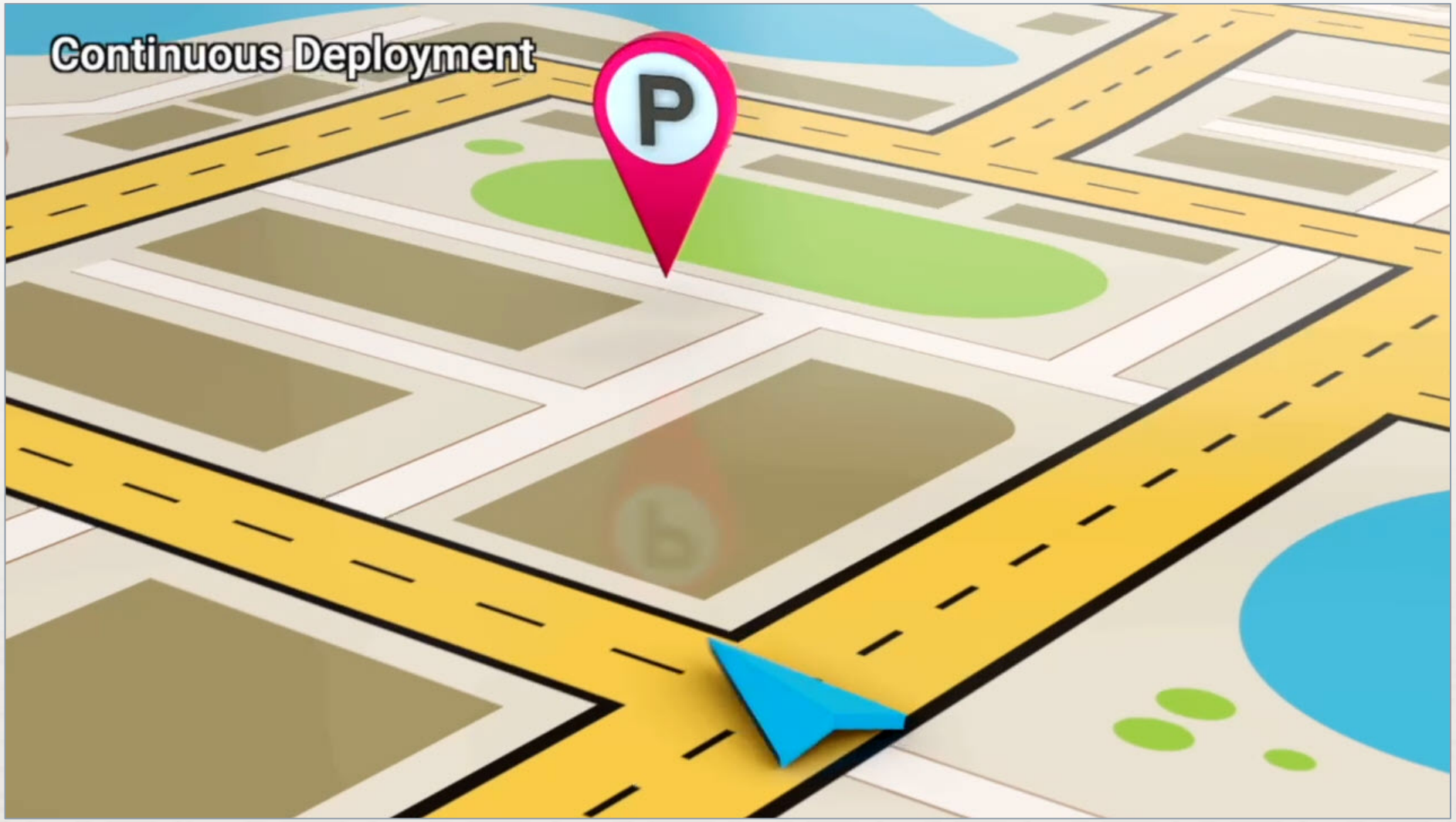
# Automation and Scripting



# Automation and Scripting



# Automation and Scripting



# Continuous Validation

- Creates reports
  - Requirements
  - Functionality
  - Vulnerabilities
- Provides assessment data

# Continuous Monitoring

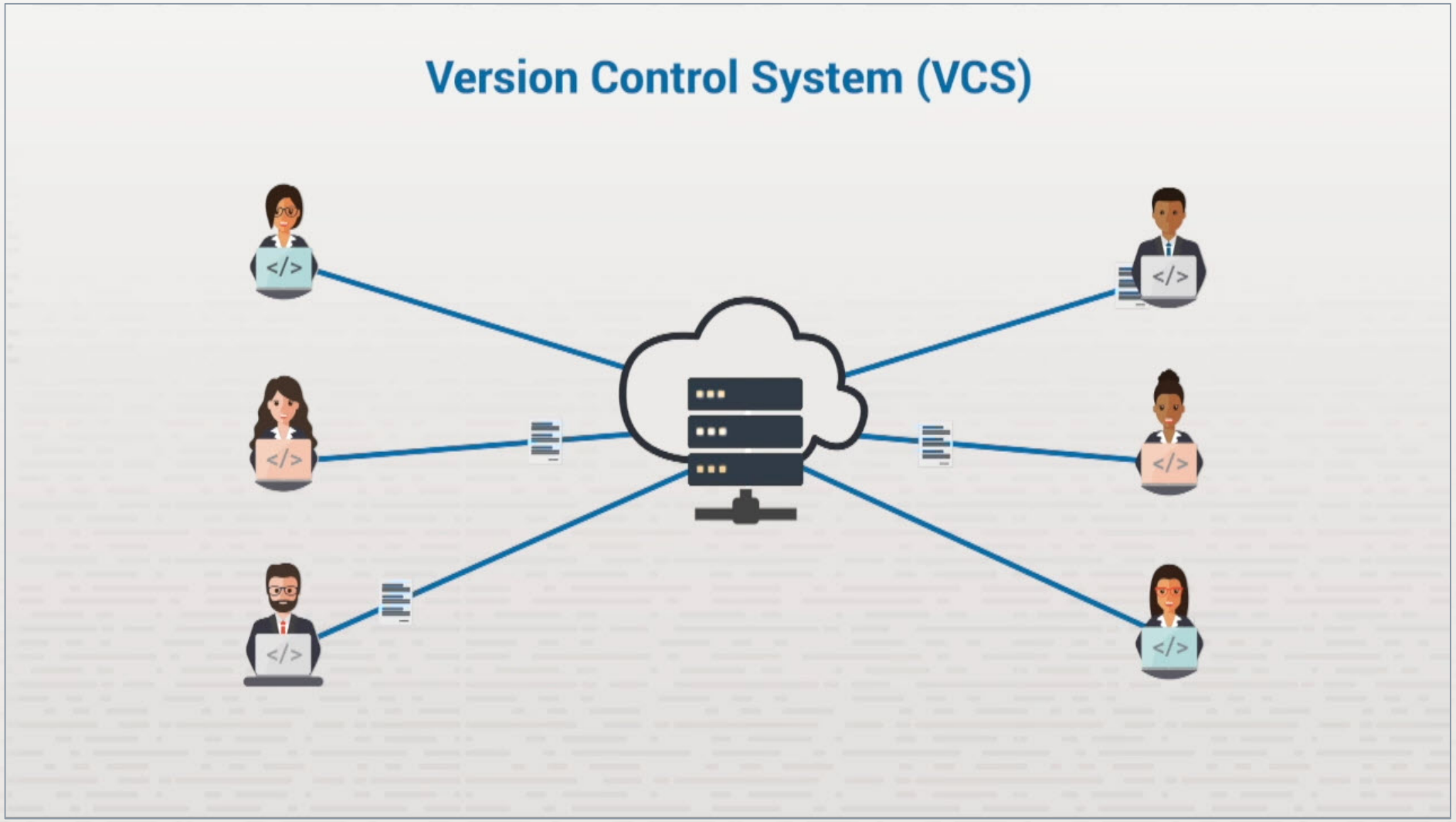
- Control functionality
- Error detection
- Configuration issues
- Transaction process tracking

# Summary

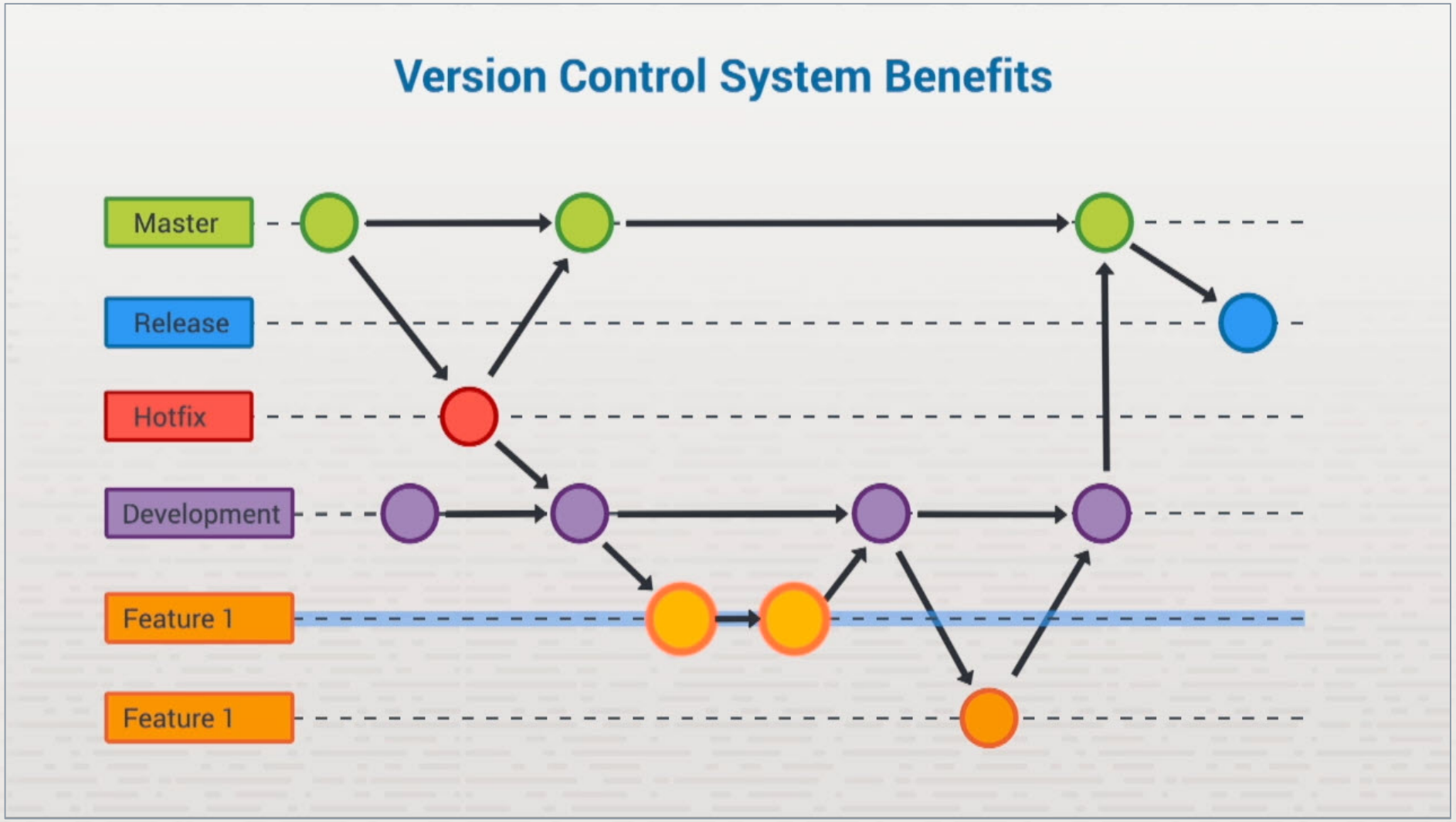
- Automated courses of action
- Continuous integration
- Continuous delivery
- Continuous deployment
- Continuous validation
- Continuous monitoring

# Version Control Management

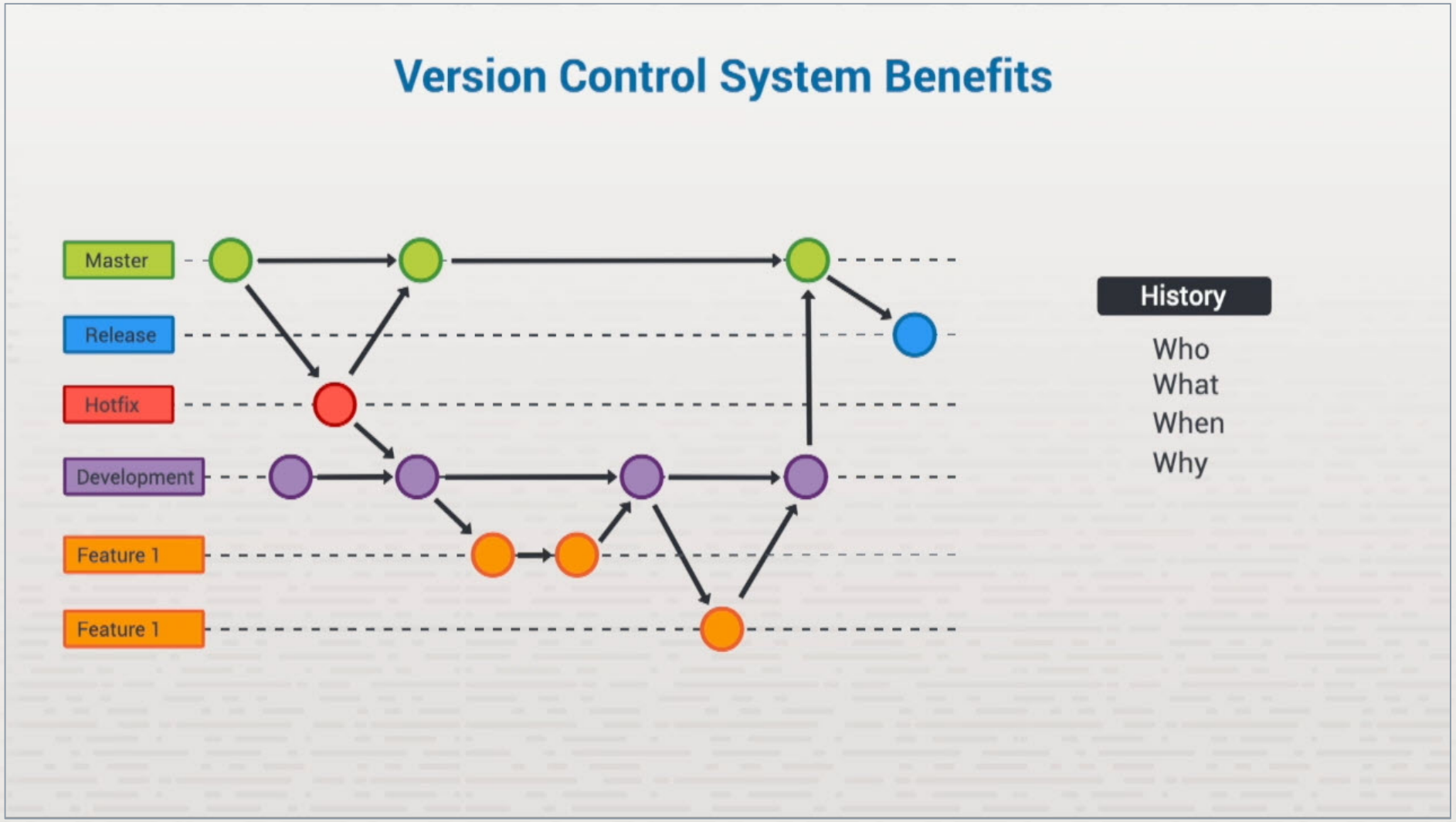
# Version Control Management



# Version Control Management



# Version Control Management



# Provisioning

- Don't give extra permissions
- Deprovision when complete
- Principle of least privilege

# Summary

- Version control system
- Provisioning
- Deprovisioning

# Secure Coding Concepts

# Normalization

- Relational organization
- Reduces redundancies
- Protects against:
  - Insertion anomalies
  - Update anomalies
  - Deletion anomalies
- Reduces disk space
- Efficient access for data manipulation

# Secure Coding Concepts

## Normalization

| Library Card Number | Name | Address | Book Number | Book Title |
|---|---|---|---|---|
| 001002003 | James Smith | 500 W Street | ZWX3347 | Five People You Meet in Coding Class |
| 001002003 | James Smith | 500 W Street | TLK9962 | The Art of Motorcycle Repair |
| 001002004 | Suzy Davis | 1100753 N Oak Dr. | BRM8215 | How to Garden |
| 001002004 | Suzy Davis | 1100753 N Oak Dr. | CRP5576 | Wildflowers of the Rockies |
| 001002005 | Holly Jensen | 946 Lantern Street | LZD4726 | Pride and Prejudice |

# Secure Coding Concepts

## Normalization

| Library Card Number | Name | Address | Book Number | Book Title |
|---|---|---|---|---|
| 001002003 | James Smith | 500 W Street | ZWX3347 | Five People You Meet in Coding Class |
| 001002003 | James Smith | 500 W Street | TLK9962 | The Art of Motorcycle Repair |
| 001002004 | Suzy Davis | 1100753 N Oak Dr. | BRM8215 | How to Garden |
| 001002004 | Suzy Davis | 1100753 N Oak Dr. | CRP5576 | Wildflowers of the Rockies |
| 001002005 | Holly Jensen | 946 Lantern Street | LZD4726 | Pride and Prejudice |

| Library Card Number | Name | Address |
|---|---|---|
| 001002003 | James Smith | 500 W Street |
| 001002004 | Suzy Davis | 1100753 N Oak Dr. |
| 001002005 | Holly Jensen | 946 Lantern Street |

| Library Card Number | Book Number | Book Title |
|---|---|---|
| 001002003 | ZWX3347 | Five People You Meet in Coding Class |
| 001002003 | TLK9962 | Motorcycle Repair Manual |
| 001002004 | BRM8215 | How to Garden |
| 001002004 | CRP5576 | Wildflowers of the Rockies |
| 001002005 | LZD4726 | Pride and Prejudice |

# Secure Coding Concepts

## Stored Procedures

One or more database statements stored as a group in a database's dictionary that execute all the statements in the collection when called

# Stored Procedures

- Centralize code
- Eliminate need to reproduce code
- Keep programs' calling rules consistent
- Protect code from users
- Limit injection attacks

# Secure Coding Concepts

## Code Obfuscation/Code Camouflage

- Alters executable code
- Retains function
- Most methods can be reverse-engineered
- Is harder to debug or manipulate

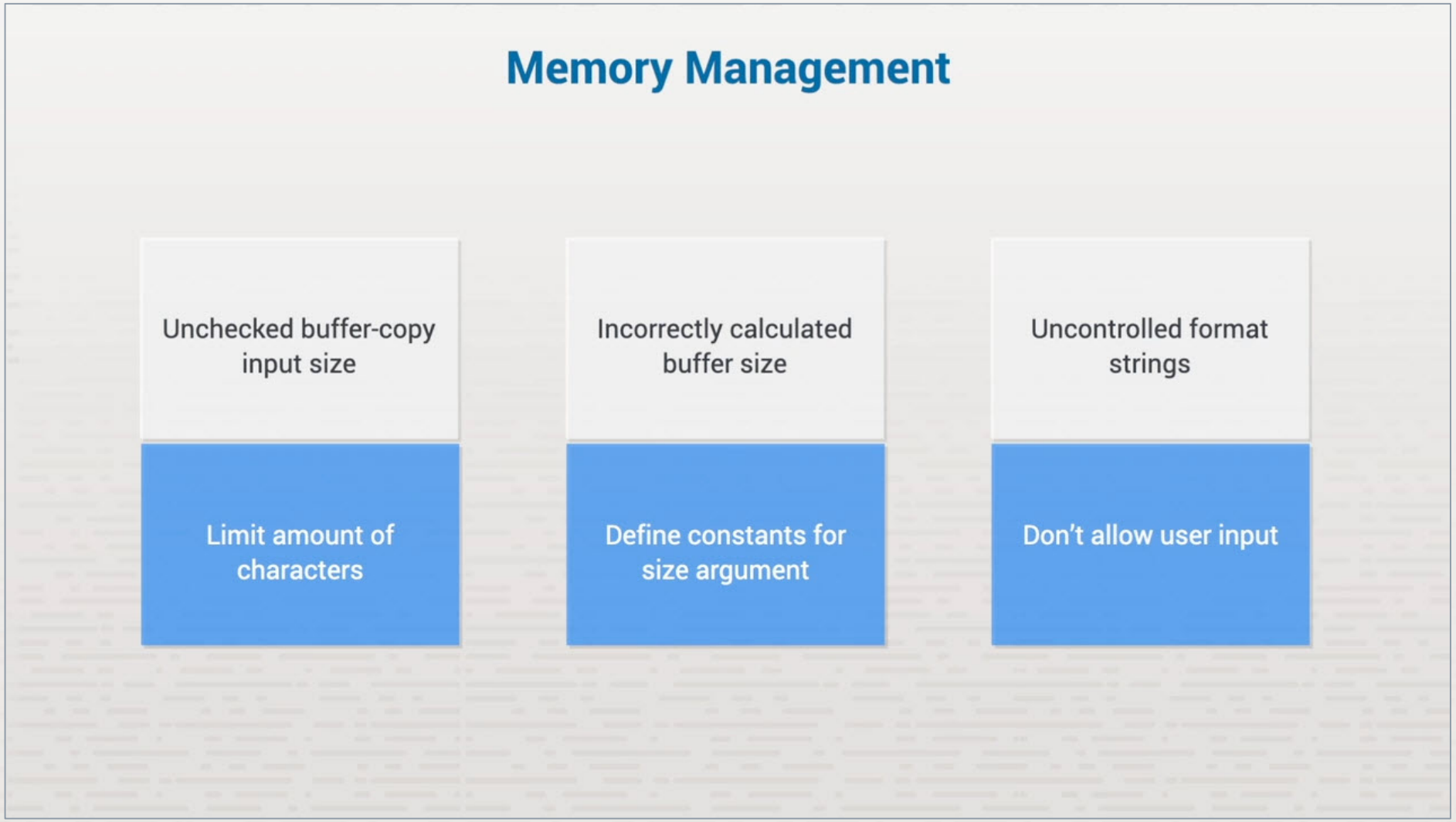
# Code Reuse

- Rule of three
- Only reuse
- Avoid duplication
- Comprehensively tested

# Dead Code

- Non-executable code
- Not used in any other computation
- Remove when found

# Secure Coding Concepts



# Secure Coding Concepts

## Third-Party Libraries and Software Development Kits

# Secure Coding Concepts

## Third-Party Libraries and Software Development Kits

**Risks:**

- Someone else's code - could be flawed
- Bundles include extra code
- Open source
- No bug-fix urgency

# Data Exposure

- Weak or non-existent encryption
- Coding flaws
- Misapplied database uploads

# Data Exposure Prevention

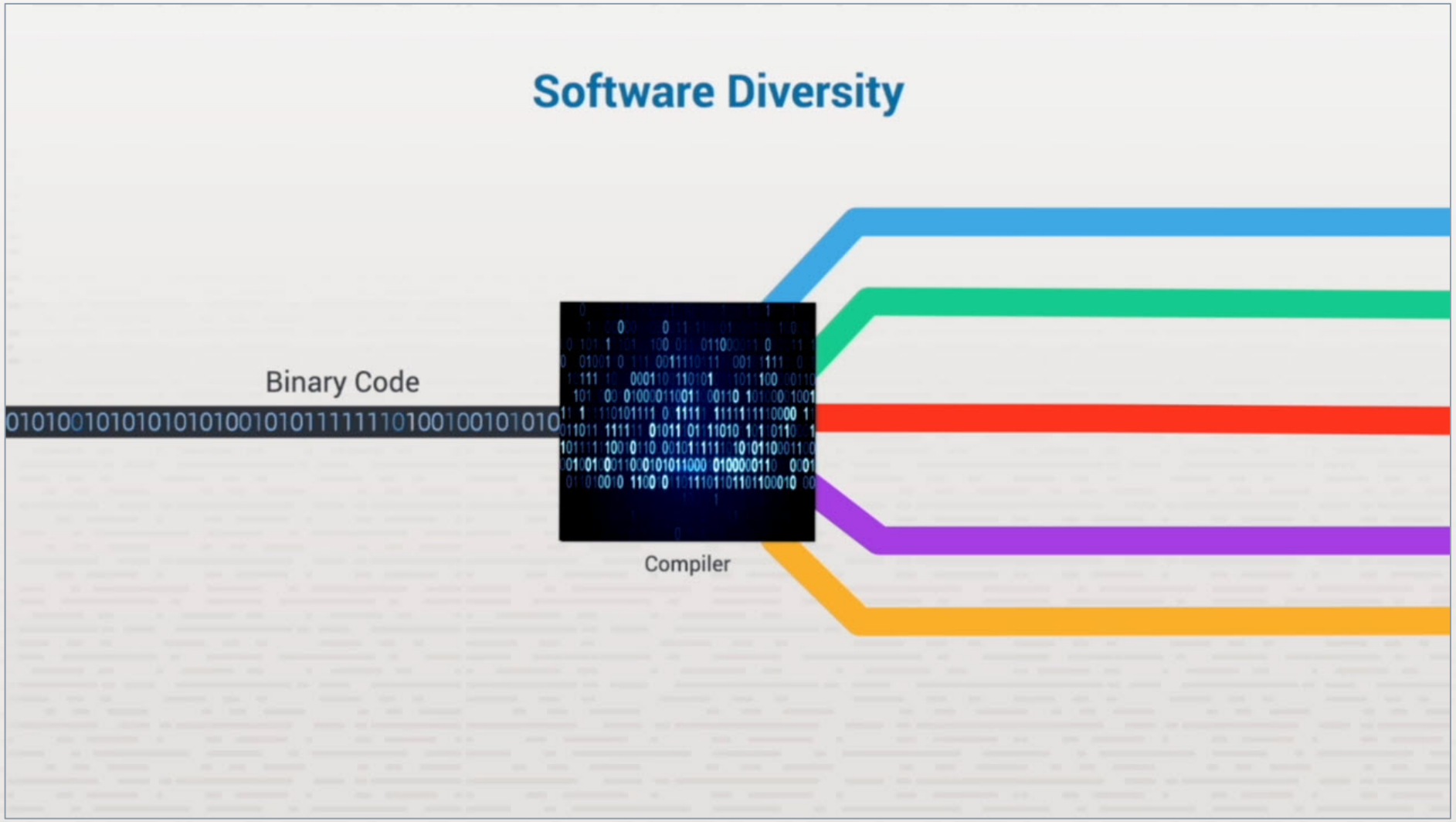
- Encrypt data
  - In transit
  - At rest
- Disable caching on forms
- Implement hashed and salted passwords

# Summary

- Normalization
- Stored procedures
- Code obfuscation
- Code reuse/dead code
- Memory management
- Third-party libraries
- SDKs
- Data exposure

# Application Hardening

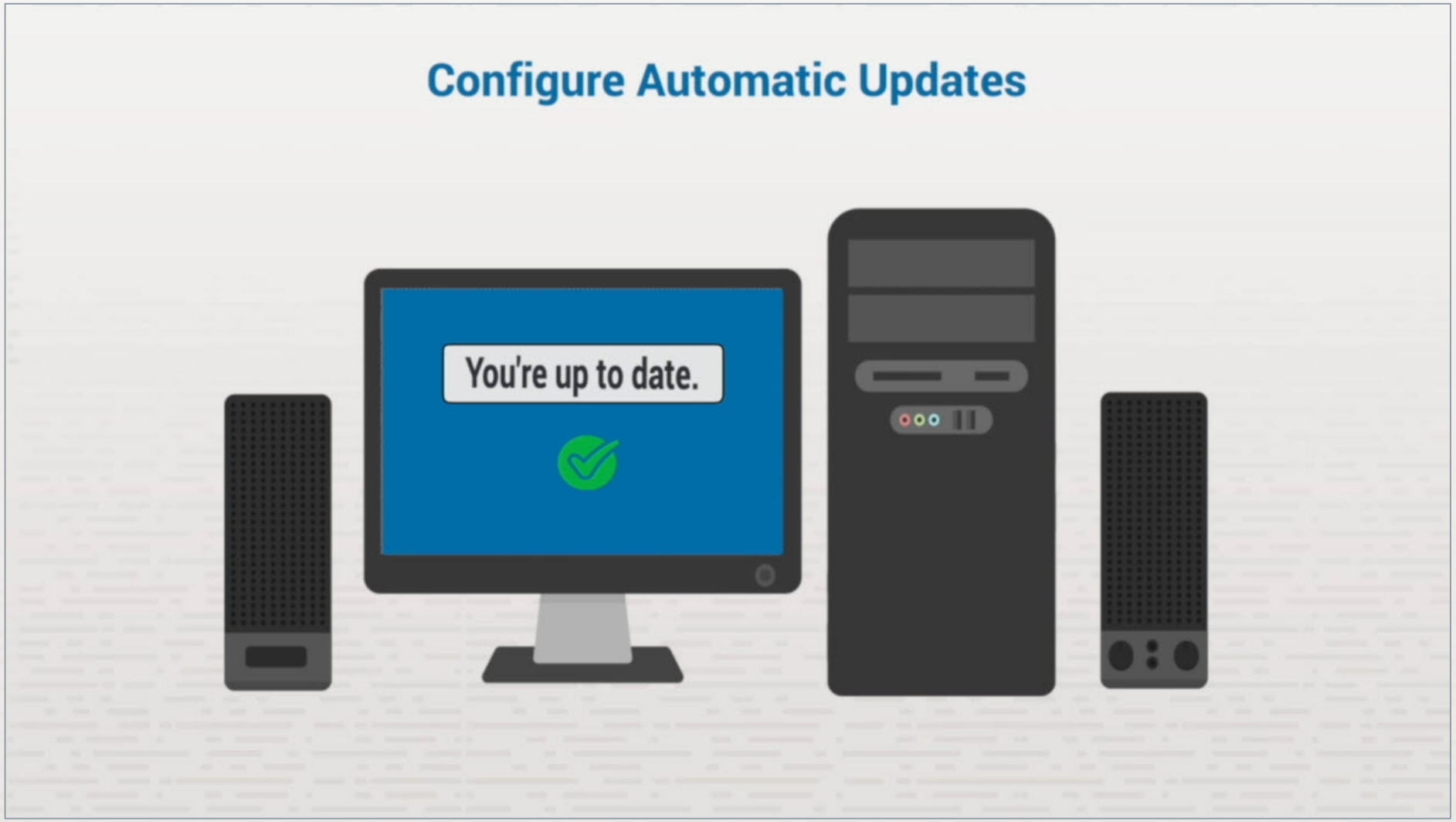
# Application Hardening



# Application Hardening

## Remove Unnecessary Applications

# Application Hardening

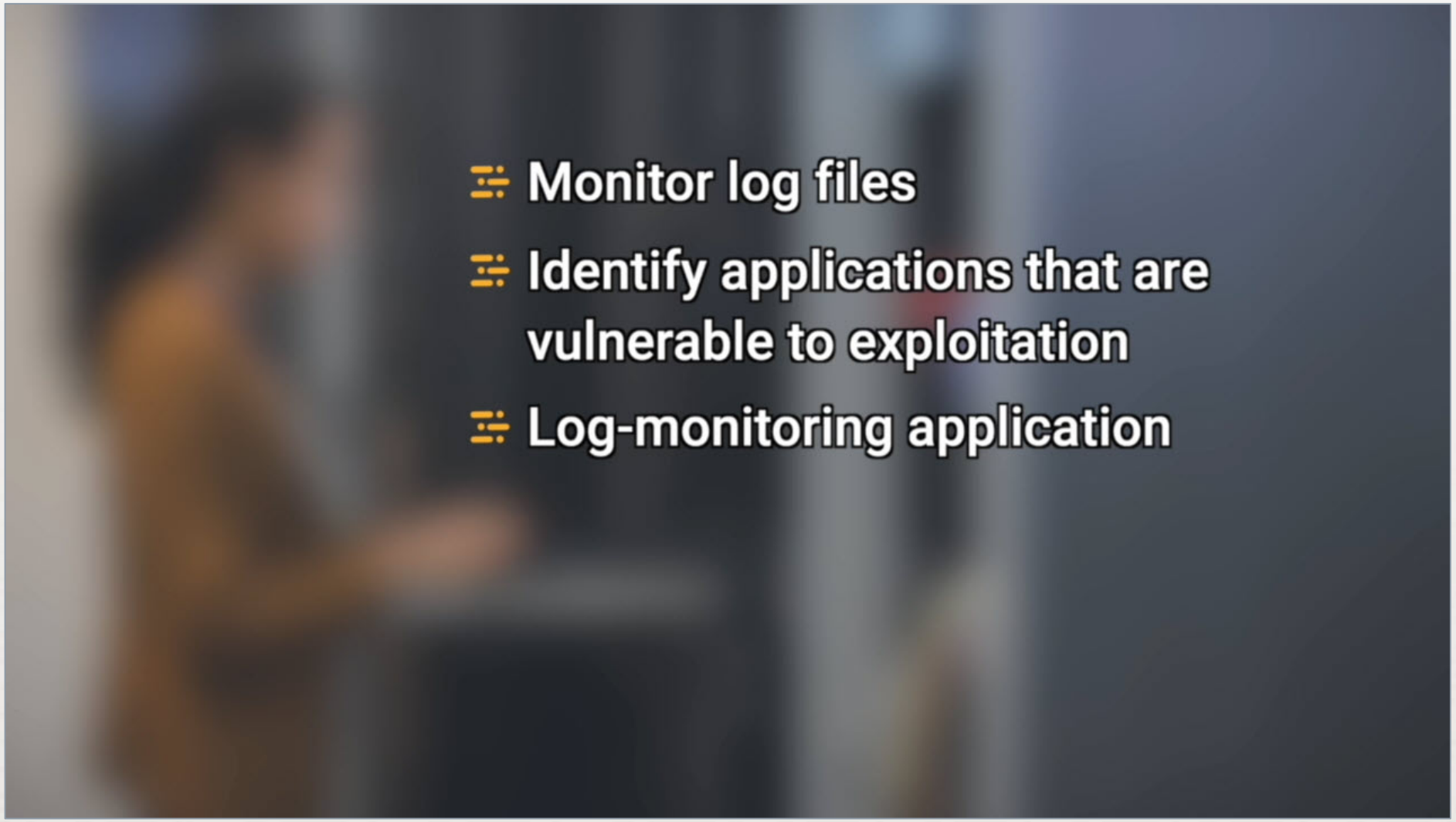


# Input Protection

- Process spawning >>Block child processes >>Make exceptions as needed
- New executable file creation >>Block creating executable files
- Existing executable modification >>Block modifying executable files

# Application Hardening

# Application Hardening



# Application Hardening

- Apply rules to all apps >>Check higher-risk apps >>Create specific hardening rules >>Create exceptions as needed
- Monitor logs with no exceptions >>Determine need for exceptions

# Summary

- Software diversity
- Unnecessary apps
- Automatic updates
- User input protection
- Log file monitoring

# In-Class Practice

Do the following labs:

- 10.4.10 Implement Application Whitelisting with AppLocker
- 10.4.12 Implement Data Execution Preventions

# Class Discussion

- What are two common standardized software development models?
- How should security be implemented in the different stages of development?
- What are the responsibilities of developers after a product is released?
- What are some important application hardening techniques?