M4: Insufficient Input/Output Validation

Threat Agents

Application Specific

Insufficient validation and sanitization of data from external sources, such as user inputs or network data, in a mobile application can introduce severe security vulnerabilities. Mobile apps that fail to properly validate and sanitize such data are at risk of being exploited through attacks specific to mobile environments, including SQL injection, Command Injection, and cross-site scripting (XSS) attacks.

These vulnerabilities can have detrimental consequences, including unauthorized access to sensitive data, manipulation of app functionality, and potential compromise of the entire mobile system.

Inadequate output validation can result in data corruption or presentation vulnerabilities, allowing malicious actors to inject malicious code or manipulate sensitive information displayed to users.

Attack Vectors

Exploitability DIFFICULT

Insufficient input/output validation exposes our application to critical attack vectors, including SQL injection, XSS, command injection and path traversal. These vulnerabilities can lead to unauthorized access, data manipulation, code execution, and compromise of the entire backend system.

Security Weakness

Prevalence COMMON

Detectability EASY

Insufficient input/output validation vulnerability occurs when an application fails to properly check and sanitize user input or validate and sanitize output data. This vulnerability can be exploited in the following ways:

Insufficient Input Validation: When user input is not thoroughly checked, attackers can manipulate it by entering unexpected or malicious data. This can bypass security measures and lead to code execution vulnerabilities or unauthorized system access.

Insufficient Output Validation: If output data is not properly validated and sanitized, attackers can inject malicious scripts that get executed by users' browsers. This can lead to cross-site scripting (XSS) attacks, enabling data theft, session hijacking, or the manipulation of displayed content.

Lack of Contextual Validation: Failing to consider the specific context or expected data formats can result in vulnerabilities like SQL injection or format string vulnerabilities. These occur when unvalidated user input is directly incorporated into database queries or improperly handled in format string functions,

allowing attackers to manipulate queries or execute arbitrary code.

Failure to Validate Data Integrity: Without validating data integrity, the application becomes vulnerable to data corruption or incorrect processing. Attackers can tamper with critical system variables or introduce malformed data that disrupts the application's functionality.

These vulnerabilities often arise from errors in application logic, incomplete implementation of validation checks, lack of security awareness, or insufficient testing and code review practices.

Technical Impacts

Impact SEVERE

Insufficient input/output validation vulnerability can have several technical impacts on the affected application:

Code Execution: A malicious actor can exploit this vulnerability to execute unauthorized code within the application's environment, bypassing the security measures.

Data Breaches: Insufficient validation can enable attackers to manipulate input, potentially leading to unauthorized access and extraction of sensitive data.

System Compromise: Attackers can gain unauthorized access to the underlying system, compromising it and potentially taking control.

Application Disruption: Malicious input can cause disruptions, crashes or data corruption, impacting the application's reliability and functionality.

Reputation Damage: Successful exploitation of this vulnerability can result in reputational harm due to data breaches and loss of customer trust.

Legal and Compliance Issues: Inadequate validation may lead to legal liabilities, regulatory penalties and non compliance with data protection regulations.

Business Impacts

Impact SEVERE

Insufficient input/output validation vulnerability has significant technical and business implications. From an application standpoint, the impacts include:

- Code Execution: Attackers can exploit this vulnerability to execute unauthorized code, potentially leading to system compromise and unauthorized access.
- Data Breaches: Insufficient validation allows attackers to manipulate input, resulting in data breaches and unauthorized access to sensitive information.
- System Disruptions: Exploitation of the vulnerability can cause application crashes, instability, or data corruption, leading to service disruptions and operational inefficiencies.
- Data Integrity Issues: Insufficient validation may result in data corruption, incorrect processing, or inaccurate outputs, compromising the reliability and integrity of the system.

On the business side, the impacts include:

- Reputation Damage: Successful exploitation of the vulnerability can result in data breaches, system disruptions, and customer distrust,

damaging the organization's reputation and brand image.
- Legal and Compliance Consequences: Non-compliance with data protection regulations due to insufficient validation can lead to legal liabilities, regulatory penalties, and potential financial losses.
- Financial Impact: Data breaches or system disruptions caused by the vulnerability can result in financial losses due to incident response, remediation costs, legal fees, and potential loss of revenue.

Am I Vulnerable To 'Insufficient Input/Output Validation'?

An application can be vulnerable to insufficient input/output validation due to:

- Lack of Input Validation: Failure to properly validate user input can expose the application to injection attacks like SQL injection, command injection, or XSS.
- Inadequate Output Sanitization: Insufficient sanitization of output data can result in XSS vulnerabilities, allowing attackers to inject and execute malicious scripts.
- Context-Specific Validation Neglect: Neglecting to consider specific validation requirements based on data context can create vulnerabilities, such as path traversal attacks or unauthorized access to files.
- Insufficient Data Integrity Checks: Not performing proper data integrity checks can lead to data corruption or unauthorized

modification, compromising reliability and security.
- Poor Secure Coding Practices: Neglecting secure coding practices, such as using parameterized queries or escaping/encoding data, contributes to input/output validation vulnerabilities.

How Do I Prevent 'Insufficient Input/Output Validation'?

To prevent "Insufficient Input/Output Validation" vulnerabilities:

- Input Validation:
  - Validate and sanitize user input using strict validation techniques.
  - Implement input length restrictions and reject unexpected or malicious data.
- Output Sanitization:
  - Properly sanitize output data to prevent cross-site scripting (XSS) attacks.
  - Use output encoding techniques when displaying or transmitting data.
- Context-Specific Validation:
  - Perform specific validation based on data context (e.g., file uploads, database queries) to prevent attacks like path traversal or injection.
- Data Integrity Checks:
  - Implement data integrity checks to detect and prevent data corruption or unauthorized modifications.
- Secure Coding Practices:
  - Follow secure coding practices, such as using parameterized queries and prepared

statements to prevent SQL injection.

- Regular Security Testing:
  - Conduct regular security assessments, including penetration testing and code reviews, to identify and address vulnerabilities.

Example Attack Scenarios

Scenario #1 Remote Code Execution via Malicious Input

An attacker identifies a mobile application lacking proper input validation and sanitization. By crafting a malicious input containing unexpected characters, they exploit the application's behavior. Due to insufficient validation, the application mishandles the input, leading to vulnerabilities. The attacker successfully executes arbitrary code, gaining unauthorized access to the device's resources and sensitive data.

Scenario #2 Injection Attacks via Insufficient Output Validation

An attacker identifies a mobile application with inadequate output validation and sanitization. They exploit an entry point where user-generated content or untrusted data is processed. By crafting malicious input containing code or scripts (e.g., HTML, JavaScript, SQL), the attacker takes advantage of the lack of output validation. Submitting the crafted input through user interaction, the application fails to validate or sanitize it, allowing the execution of injected code or unintended operations. The attacker successfully executes injection-based attacks like cross-site scripting (XSS) or SQL injection,

compromising the application's integrity and gaining access to sensitive information.

Scenario #3 Remote Code Execution via Malformed Output

An attacker identifies a mobile application that processes user-provided data and generates dynamic output.The attacker crafts specially formatted data that exploits the application's insufficient output validation. The attacker submits the malformed data to the application, either through direct interaction or by exploiting an exposed API. The application fails to properly validate or sanitize the generated output, allowing the attacker's crafted data to execute code or trigger unintended actions. By exploiting this vulnerability, the attacker achieves remote code execution, gaining control over the mobile device, its resources, or sensitive data.

References

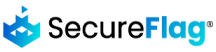
- OWASP
  - Input Validation Cheat Sheet
  - Improper Data Validation
- External
  - External References
  - Improper Input Validation

Edit on GitHub

Spotlight: Monzo

At Monzo we're building a new kind of bank. One that's built for your smartphone and designed for the way we live today. Founded in early 2015, our mission is to build the best bank account in the world. In August 2016 we became a regulated bank, and a team of more than 2,000 work from our London HQ and remotely around the world.

Corporate Supporters



Become a corporate supporter