XSS\_chapter4\_Doc

# 4.1.4.1 Core Scanning Architecture

## Technology Overview

| Technology | Purpose | Why It’s Used |
| --- | --- | --- |
| **BeautifulSoup** | HTML parsing/form discovery | Accurately extracts input fields |
| **Requests** | HTTP request handling | Maintains sessions/cookies |
| **Playwright** | Headless browser validation | Detects DOM-based/real-user impact |
| **Threading** | Parallel payload testing | Speeds up vulnerability discovery |
| **urllib3** | Proxy/SSL management | Enables secure proxy configurations |

## Initialization Process

**Configurable scan settings**

| Parameter | Description | Example/Default |
| --- | --- | --- |
| **Target URL** | Website to scan (single or multiple) | https://example.com |
| **Threads** | Parallel tests (higher = faster scan) | Default: 20 |
| **Crawling Depth** | How many pages deep to search | Default: 3 pages |
| **Proxy** | Route traffic through intermediary | http://proxy:port |
| **Report Format** | Output file type | JSON, CSV, or XML |

## Design Principles

**Why the scanner works this way**

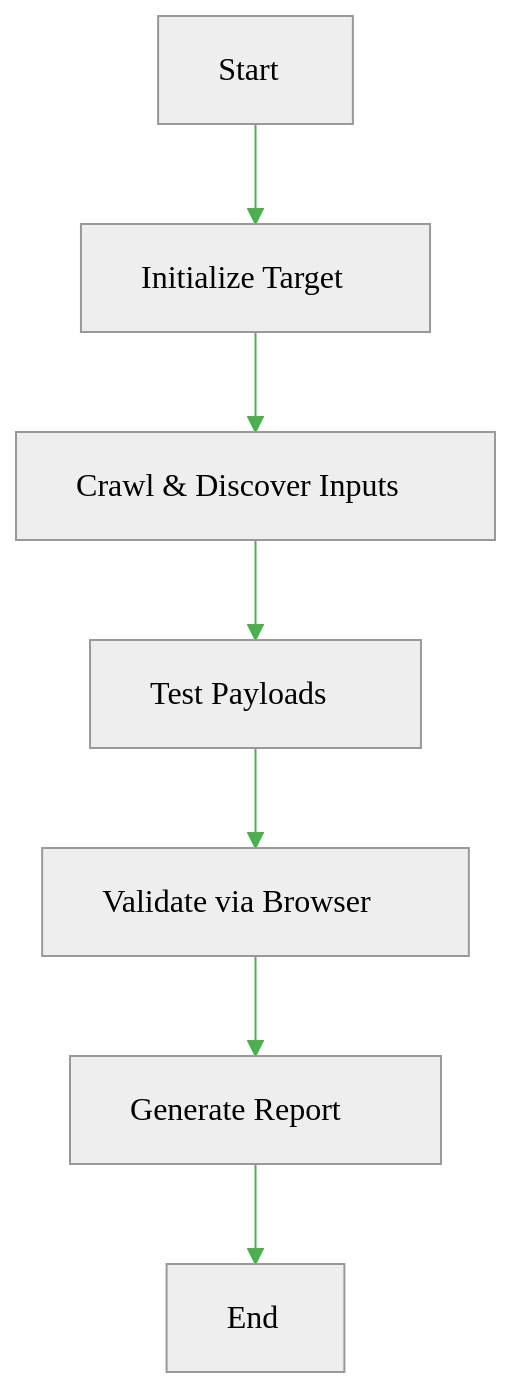
1. **Accuracy First**
   * Real browser emulation ensures vulnerabilities actually impact users
   * Multi-layered testing eliminates false positives
2. **Safety Controls**
   * Rate limiting prevents server overload

## Workflow Preview

*"The scanner crawls pages → injects payloads → verifies results → generates reports."*

# 4.1.4.2 Workflow Phases

## High-Level Workflow



### **Phase 1: Target Initialization**

*Prepares the scanner with user inputs and safety checks*

\*Key Steps\*\*:

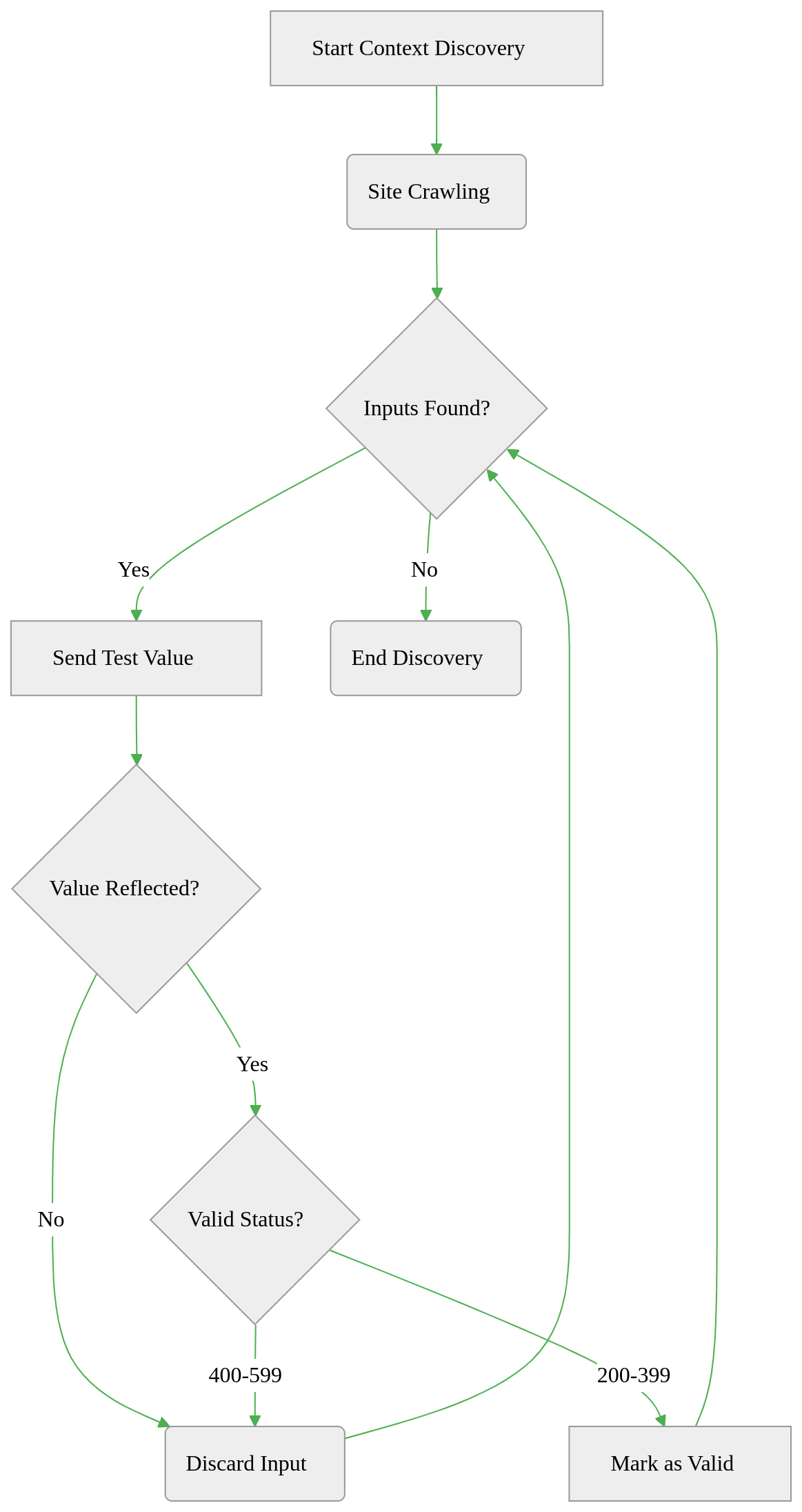
* **Input Validation**: Verify URL/file format and target reachability
* **Parameter Setup**: Define scan depth, threads, and output format
* **Network Prep**: Configure proxy for traffic inspection/debugging
* SSL verification disabled to simplify proxy debugging

**Outcome**:

* Ready-to-scan environment with defined boundaries
* Traffic routed through specified intermediaries (if proxy used)

### Phase 2: Context Discovery

*Identifies and prepares testable inputs while filtering noise*



#### Process Flow:

1. **Site Crawling**
   * Starts at initial URL
   * Follows same-domain links recursively
2. **Parameter Extraction**  
   **Forms**:
   * Discovers <input>, <textarea>, hidden fields (e.g., CSRF tokens)
   * Auto-fills context-aware values:

| Field Type | Example Value |
| --- | --- |
| Email | test@example.com |
| URL | http://test |
| Hidden | Preserves original |

* **URL Parameters**:
  + Extracts from query strings (e.g., ?search=)

1. **Validation & Filtering**
   * Tests parameter reflection:
     + Sends unique test values
     + Requires exact reflection to qualify
   * Removes parameters that:
     + Return errors (HTTP 400-599)
     + Show no value reflection

#### Data Structuring:

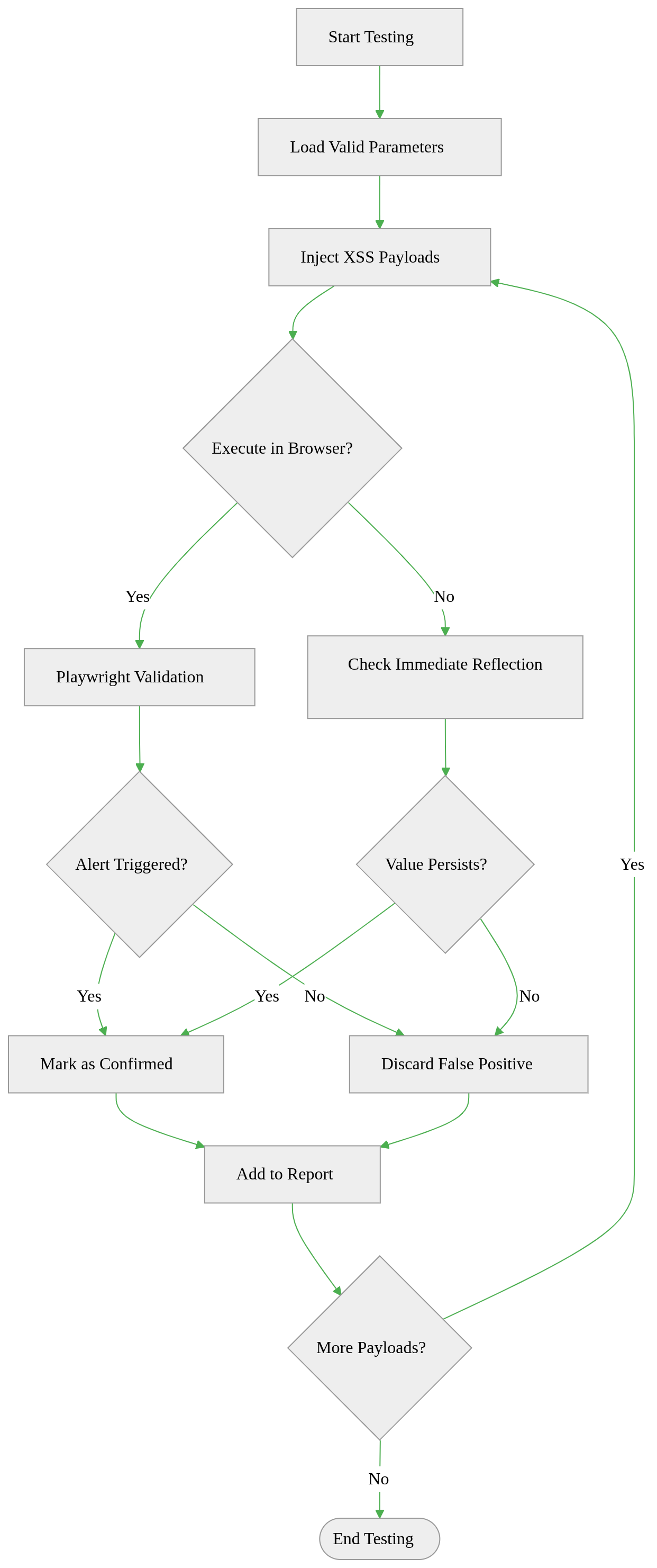
{  
 "context": "https://example.com/search (GET)",  
 "origin": "https://example.com",  
 "parameters": {  
 "q": {  
 "type": "text",  
 "pattern": ""  
 }  
 }  
}

#### Outcome:

* Curated list of testable inputs with:
  + Parameter types and validation rules
  + Associated HTTP methods (GET/POST)
  + Origin page URLs
* Eliminated redundant/unresponsive parameters

### Phase 3: Payload Testing & Validation

*Executes attacks and confirms vulnerabilities*



**Key Actions**:

1. **Payload Injection**:
   * Tests multiple attack vectors
2. **Validation Methods**:
   * **Direct Reflection**: Checks if payload appears in server response
   * **Redirect Handling**: Re-checks original page after redirects for stored payloads
   * **Browser Execution**: Verifies real impact using hidden browser
3. **False Positive Handling**:
   * Discards non-executed payloads
   * Ignores transient reflections

**Outcome**:

* Confirmed vulnerabilities categorized as:
  + **Reflected** (Immediate response)
  + **Stored** (Persistent after redirects/page reloads)
  + **DOM-Based** (Requires browser execution)

### Phase 4: Reporting & Output

*Formats and delivers verified vulnerabilities*

**Key Actions**:

1. **Data Organization**:
   * Structures findings into:
     + Vulnerability type (Reflected/Stored/DOM)
     + Parameter/input field affected
     + Successful payload
     + Source URL
     + Timestamp
2. **Export Formats**:
   * **JSON**: For developers and automated tools
   * **CSV**: For spreadsheets and manual review
   * **XML**: For enterprise system integration
3. **Progress Tracking**:
   * Real-time updates via callback system:
     + New parameters discovered
     + Vulnerabilities confirmed
     + Scan completion status