Decoding after sanitization (generalized)

Tracks the return value of an HTML sanitizer into an escape-sequence decoder, indicating an ineffective sanitization attempt.

```
import javascript
import DataFlow
import DataFlow::PathGraph

/**
 * A call to a function that may introduce HTML meta-characters by
 * replacing `%3C` or `\u003C` with `<`.
 */
class DecodingCall extends CallNode {
  string kind;
  Node input;

  DecodingCall() {
    getCalleeName().matches("decodeURI%") and
    input = getArgument(0) and
    kind = "URI decoding"
    or
    input = this.(JsonParserCall).getInput() and
    kind = "JSON parsing"
  }

  /** Gets the decoder kind, to be used in the alert message. */
  string getKind() { result = kind }

  /** Gets the input being decoded. */
  Node getInput() { result = input }
}

class DecodingAfterSanitization extends TaintTracking::Configuration {
  DecodingAfterSanitization() { this = "DecodingAfterSanitization" }

  override predicate isSource(Node node) { node instanceof HtmlSanitizerCall }

  override predicate isSink(Node node) { node = any(DecodingCall c).getInput() }
}

from DecodingAfterSanitization cfg, PathNode source, PathNode sink, DecodingCall decoder
where
  cfg.hasFlowPath(source, sink) and
  decoder.getInput() = sink.getNode()
select sink.getNode(), source, sink,
  decoder.getKind() + " invalidates the HTML sanitization performed $@.", source.getNode(),
"here"
```