Approach & Summary

Date: August 17, 2025

Objective

The goal of this assessment was to perform Cybersecurity Spend Record Matching by aligning unstructured supplier spend descriptions to a controlled product repository. The expected outcome was two matched files (`matched_spend_records.csv` and `matched_spend_records_hard.csv`) plus an approach summary explaining the method.

Constraints

- All work was completed manually using Microsoft Excel for Web (no subscription features, no scripting).
- Matching had to be reproducible, step-driven, and auditable.

Step 1: Data Preparation

- Loaded provided files into Excel:
  - `SpendTbl` (spend records).
  - `HardTbl` (subset of challenging records).
  - `KeywordMapTbl` (product repository, later cleaned).
- Added helper columns in `SpendTbl`/`HardTbl`:
  - `supplier_norm` → lowercase, trimmed supplier.
  - `text_norm` → lowercase spend description.
  - `text_norm_compact` → spaces/punctuation removed.
  - `product_key_manual` → blank column, used if manual overrides required.

Step 2: Repository Cleanup (Quick Cleanup)

- Fixed obvious typos and removed stray spaces.
- Normalized `vendor_name` → `vendor_norm` (consistent lowercase vendor).
- Added high-value aliases for reliable matching, e.g.:
  - `Falcon` → `CrowdStrike Falcon Enterprise`
  - `Zscaler IA` → `Zscaler Internet Access`
  - `Firepower 2000` / `Firepower 2100` → `Cisco Firepower Series`
  - `MDE`, `Defender O365`, `Defender Cloud Apps`
  - `Fortianalyzer`
- Assigned priorities:
  - Aliases = 10–20
  - Main repository entries = 50
- Created helper fields:
  - `key_compact` (alias with spaces removed)
  - `key_len` (length, for tiebreaking)

- `km_order` (row order for deterministic matching)

Step 3: Matching Setup in Inspector

Built an `Inspector` sheet to test matches one row at a time, controlled by a `row_pointer`:

- **Preview**: Shows active row's normalized text and supplier.
- **insp_hit**: TRUE/FALSE check for each repository key using formula:

```
=OR(
  IFERROR(ISNUMBER(SEARCH([@key], Inspector!$C$6)),FALSE),
  IFERROR(ISNUMBER(SEARCH([@key_compact], Inspector!$C$7)),FALSE)
)
```

- **Matched Key**:
  - If `product_key_manual` filled → use that.
  - Otherwise return first matching repository key by priority/length/order.
- **Lookups**: Pulled `product_id`, `product_name`, and `vendor_name` from `KeywordMapTbl`.
- **Vendor Alignment**: Compared spend's `supplier_norm` with repository's `vendor_norm`.
- **Confidence & Method**:
  - High = keyword + vendor match
  - Medium = keyword only
  - Low = fallback/noisy matches
  - Method labelled as `keyword`, `vendor+keyword`, or `fallback`

Step 4: Batch Processing

- Instead of processing rows one-by-one, used `SEQUENCE` (to generate sequential row indices) and `AGGREGATE` (to handle errors and aggregate results) to pull 20 matches at a time in `Inspector`.
- Copied results back into `SpendTbl`/`HardTbl` → five output columns:
  1. `product_id`
  2. `product_name`
  3. `vendor_name`
  4. `confidence`
  5. `method`
- Pasted values so the output files were static and reproducible.

Step 5: Outputs

- **`yaser_matched_spend_records.csv`** (prefixed with 'yaser_' for user-specific identification)
  - Contains all spend rows + five final output columns.
  - Mix of high, medium, low confidence matches; majority matched via keyword.
- **`yaser_matched_spend_records_hard.csv`** (prefixed with 'yaser_' for user-specific identification)
  - Contains subset of hard rows, processed with the same logic.

    - Expected fewer rows, with higher manual review importance.
- Both CSVs validated: no structural errors, headers correct, columns align with instructions.

Step 6: Quality Checks

- Spot-checked multiple rows: confirmed keys like `falcon`, `zscaler ia`, `firepower 2000` matched correctly.
- Verified `row_pointer = 1` starts correctly (header skipped).
- Confirmed priority order worked (10 → 20 → 50).
- Confirmed vendor alignment flag worked consistently.

Known Limitations

- Some borderline cases rely on keyword fragments; may require manual overrides (`product_key_manual`).
- Excel Web lacks scripting, so scaling was slower than a programmatic solution.

Conclusion

The matching process successfully mapped spend descriptions to repository products using a transparent, Excel-only approach. The workflow included repository cleanup, inspector-based testing, batch application, and two final CSV outputs. This ensured reproducibility, clarity, and alignment with the instructions.

Deliverables produced:

- `yaser_matched_spend_records.csv`
- `yaser_matched_spend_records_hard.csv`
- This Approach & Summary document