Cybersecurity Vulnerability Risk Tracker (VRT)

ISE 305: Database Design and Practice

Masumul Mozumder, Rasha Hoda, Bushra Paracha

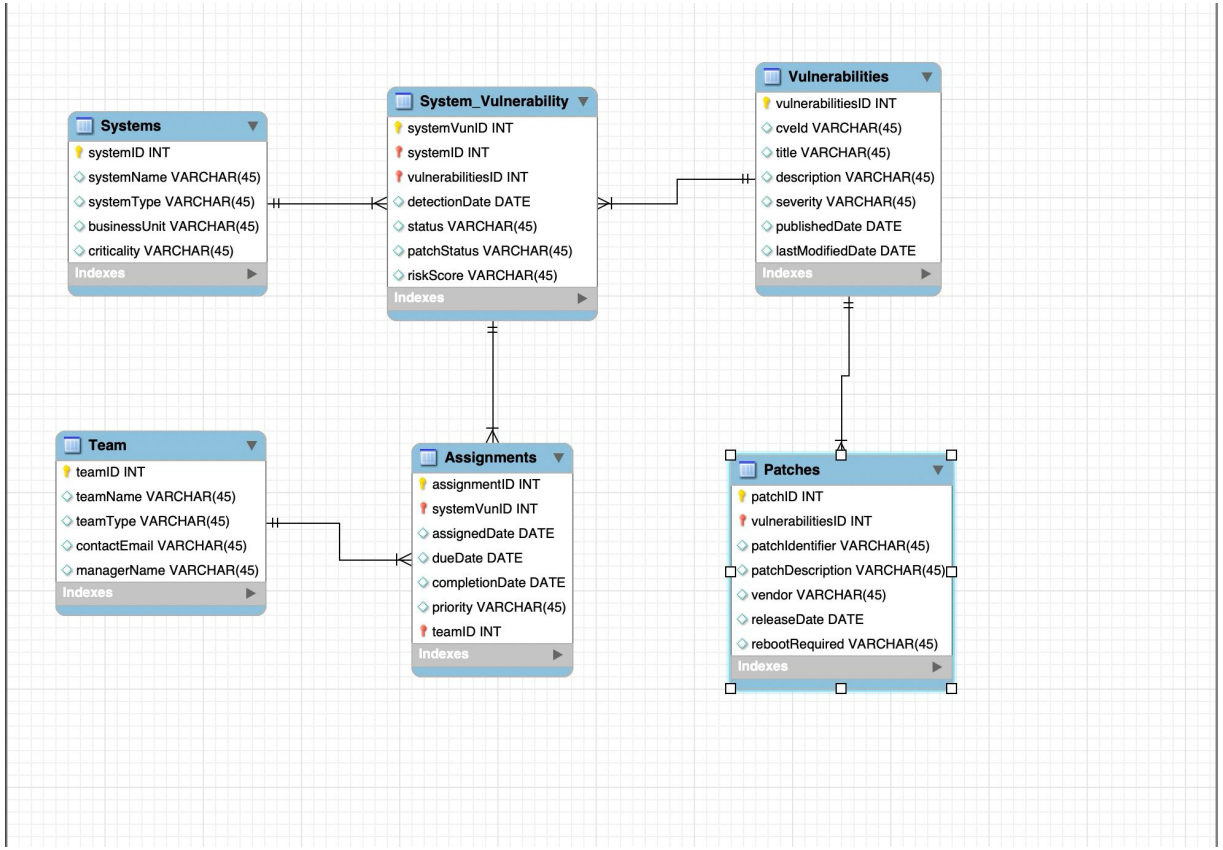
Project Overview - Cybersecurity Vulnerability Risk Tracker (VRT)

- Relational database that tracks vulnerabilities, affected systems, remediation assignments, and patching progress
- Based on how enterprises and federal agencies manage vulnerability risk
- Helps spot high-risk and overdue vulnerabilities across teams and business units
- Gives security teams one place to see systems, vulnerabilities, teams, and patch status so they can decide what to fix first

System Architecture - How the VRT Works System Architecture

- **Systems:** Represents assets across business units; each system can have multiple vulnerabilities.
- **Vulnerabilities:** CVE-based records with severity, dates, and descriptions.
- **System_Vulnerability:** Junction table linking systems ↔ vulnerabilities while tracking detection date, status, and risk score.
- **Patches:** Vendor patches mapped to specific vulnerabilities.
- **Teams:** Responsible remediation groups (e.g., AppSec, Infrastructure, SOC).
- **Assignments:** Tracks which team was assigned to remediate which system vulnerability and by when.

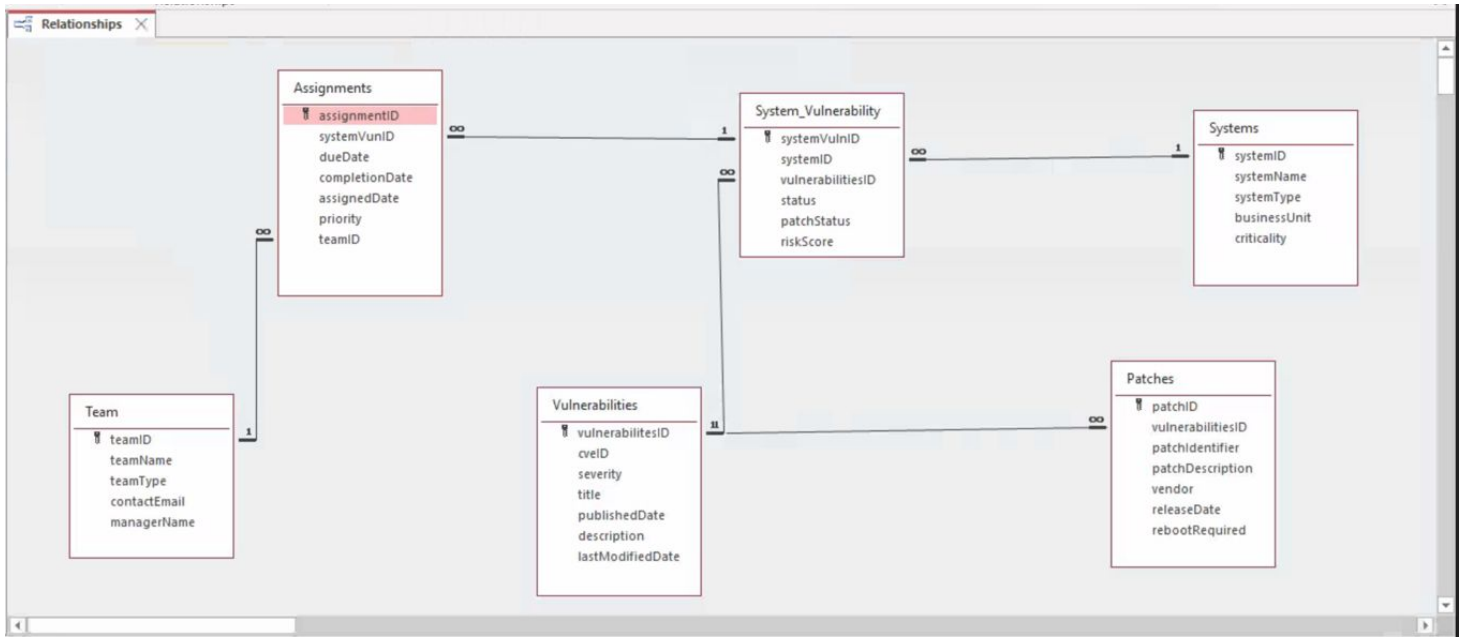
Relational Data Model (Conceptual Design)



- Defines core entities and relationships
- Shows primary keys, foreign keys, and cardinalities
- Designed before implementing in MS Access
- Basis for table creation and constraints

MS Access Relationships (Implemented RDM)

- All relationships implemented with PK–FK links
- Referential integrity enforced
- Matches conceptual RDM structure
- Used for forms, queries, and data population



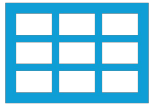
Data Quality, Constraints & Integrity Overview

- Primary keys: systemID, vulnerabilitiesID, patchID, teamID, assignmentID, systemVunID
- Foreign keys:
  - System_Vulnerability.systemID → Systems.systemID
  - System_Vulnerability.vulnerabilitiesID → Vulnerabilities.vulnerabilitiesID
  - Assignments.systemVunID → System_Vulnerability.systemVunID
  - Assignments.teamID → Team.teamID
  - Patches.vulnerabilitiesID → Vulnerabilities.vulnerabilitiesID
- Required fields: key IDs, severity, detectionDate, teamName, etc.
- Lookup / controlled values: severity, patchStatus, rebootRequired (Yes/No)
- Data types: INT for IDs, DATE for timelines, Short Text for names/descriptions

Data Flow: Vulnerability Lifecycle

Vulnerability Discovered: Imported from NVD/CVE dataset or detected internally.

Linked to an Affected System: Tracked through System_Vulnerability with detection date and risk score.



Assigned to a Team: Assignment table captures priority, due date, and completion status.

Patch Released: Patch table stores vendor, identifier, and whether reboot is required.

Status Updated: PatchStatus and status fields updated until risk is remediated.

Real-World Impact / Use Cases

- Enterprise Security Teams: Track critical CVEs across hundreds of systems; ensure high-risk vulnerabilities get assigned and fixed.
- Government & Compliance: Supports reporting for FISMA, NIST, FedRAMP by showing patch status, severity, and overdue risks.
- IT Operations / Patch Management: Quickly see which systems need patches, which patches are available, and where delays exist.
- Incident Response: During a cyber incident, analysts can identify vulnerable systems and prioritize containment steps.

Which systems have the highest total vulnerability risk scores?

```
SELECT s.systemID,
s.systemName,
    SUM(sv.riskScore) AS
totalRisk
FROM Systems AS s
INNER JOIN
System_Vulnerability AS sv
    ON s.systemID =
sv.systemID
GROUP BY s.systemID,
s.systemName
ORDER BY
SUM(sv.riskScore) DESC;
```

- Combines system + vulnerability data
- Summarizes total organizational risk
- Helps see which systems need priority attention

Query 1: Results + Explanation

- Shows which systems carry the highest risk load
- Useful for CISOs, managers, security teams
- Helps decide where to allocate resources first

systemID	systemName	totalRisk
21	System_21	419
47	System_47	419
28	System_28	417
49	System_49	410
22	System_22	398
3	System_3	388
11	System_11	376
37	System_37	369
17	System_17	368
32	System_32	368
36	System_36	365
45	System_45	364
41	System_41	359
38	System_38	358
20	System_20	353
13	System_13	353
19	System_19	352
4	System_4	349
27	System_27	349
29	System_29	348
35	System_35	347
8	System_8	344
15	System_15	342
30	System_30	340
9	System_9	338
34	System_34	331
46	System_46	327
40	System_40	318
18	System_18	317
23	System_23	313
2	System_2	312
5	System_5	303
43	System_43	299
48	System_48	298
16	System_16	295
14	System_14	293
10	System_10	293
12	System_12	286
42	System_42	285

Which systems contain critical vulnerabilities, and what is their status?

```
SELECT v.vulnerabilitiesID,
       v.cveID,
       v.severity,
       sv.status,
       sv.systemID
FROM Vulnerabilities AS v
INNER JOIN System_Vulnerability AS sv
    ON v.vulnerabilitiesID = sv.vulnerabilitiesID
WHERE v.severity = "Critical";
```

- Filters the dataset to only Critical-severity vulnerabilities
- Shows the current status of each vulnerability (Open, In Progress, Resolved)
- Helps identify which systems are affected by the highest-severity issues

Query 2: Results + Explanation

- Shows all Critical vulnerabilities across systems
- Helps identify gaps in remediation(e.g., Critical vulnerabilities still Open or In Progress)
- Supports risk assessment and compliance reviews(auditors focus heavily on Critical findings)
- Highlights systems needing urgent attention

vulnerabilitiesID	cveID	severity	status	systemID
5	CVE-2025-1248	Critical	Open	40
5	CVE-2025-1248	Critical	Resolved	19
6	CVE-2025-6221	Critical	Resolved	38
6	CVE-2025-6221	Critical	Resolved	24
6	CVE-2025-6221	Critical	Open	18
7	CVE-2025-9242	Critical	In Progress	33
11	CVE-2025-4124	Critical	Open	32
11	CVE-2025-4124	Critical	Resolved	6
14	CVE-2025-6205	Critical	In Progress	41
14	CVE-2025-6205	Critical	In Progress	47
15	CVE-2025-5423	Critical	Open	43
15	CVE-2025-5423	Critical	Open	50
16	CVE-2025-5928	Critical	Open	14
16	CVE-2025-5928	Critical	In Progress	22
16	CVE-2025-5928	Critical	Open	11
22	CVE-2025-6188	Critical	Open	46
22	CVE-2025-6188	Critical	Open	33
27	CVE-2016-7836	Critical	In Progress	37
27	CVE-2016-7836	Critical	In Progress	8
30	CVE-2021-2255	Critical	In Progress	40
30	CVE-2021-2255	Critical	Resolved	49
30	CVE-2021-2255	Critical	Open	2
32	CVE-2021-4322	Critical	In Progress	18
32	CVE-2021-4322	Critical	In Progress	30
33	CVE-2013-3918	Critical	Open	19
41	CVE-2025-4008	Critical	Open	8
43	CVE-2025-5968	Critical	Open	35
50	CVE-2025-5086	Critical	In Progress	18
57	CVE-2025-5517	Critical	Open	14
57	CVE-2025-5517	Critical	Open	23
57	CVE-2025-5517	Critical	Open	20
58	CVE-2025-5781	Critical	In Progress	32
58	CVE-2025-5781	Critical	In Progress	6
58	CVE-2025-5781	Critical	Open	39
59	CVE-2025-7775	Critical	Open	2
59	CVE-2025-7775	Critical	Open	21
67	CVE-2025-8088	Critical	Open	40
69	CVE-2013-3893	Critical	Open	9
72	CVE-2022-4079	Critical	Open	49

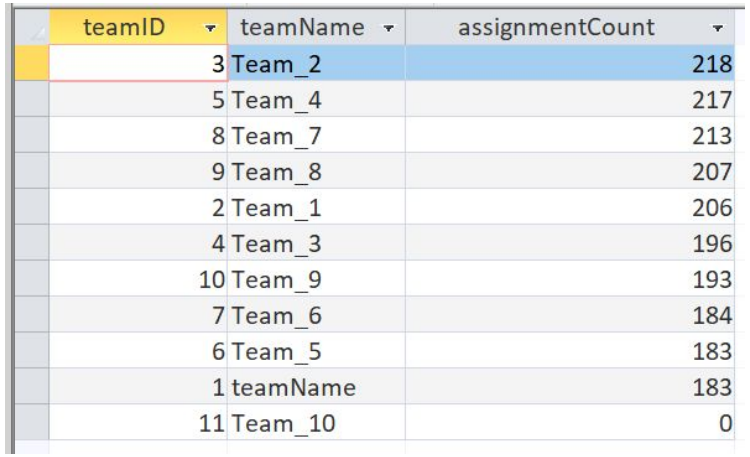
How many vulnerability assignments does each team have?

```
SELECT t.teamID,
t.teamName,


COUNT(a.assignmentID)
AS assignmentCount

FROM Team AS t

LEFT JOIN Assignments AS
a

      ON t.teamID = a.teamID

GROUP BY t.teamID,
t.teamName

ORDER BY
COUNT(a.assignmentID)
DESC;
```

- Shows workload distribution
- Uses a LEFT JOIN to include teams with 0 assignments
- Helps identify overloaded teams

Query 3: Results + Explanation

- Ensures fair workload balancing
- Helps managers allocate resources and build schedules
- Prevents bottlenecks in remediation efforts



teamID	teamName	assignmentCount
3	Team_2	218
5	Team_4	217
8	Team_7	213
9	Team_8	207
2	Team_1	206
4	Team_3	196
10	Team_9	193
7	Team_6	184
6	Team_5	183
1	teamName	183
11	Team_10	0

How many open vulnerabilities do we have at each severity level?

```
SELECT
    v.severity,
    COUNT(*) AS openVulnCount
FROM
    Vulnerabilities AS v
    INNER JOIN System_Vulnerability AS sv
        ON v.vulnerabilitiesID = sv.vulnerabilitiesID
WHERE
    sv.status = 'Open'
GROUP BY
    v.severity
ORDER BY
    openVulnCount DESC;
```

- Counts how many open vulnerabilities exist at each severity level
- Helps identify how much high-severity risk is still unresolved
- Shows whether your environment has more High, Medium, or Critical open issues
- Useful for prioritizing remediation and reporting to leadership

Query 4: Results + Explanation

- High severity has the most unresolved items
- Critical vulnerabilities are still significantly high → major concern
- Low severity still needs attention but is lower priority
- Gives a clear severity-based distribution of open issues

severity	openVulnCo
High	521
Medium	473
Critical	402
Low	153

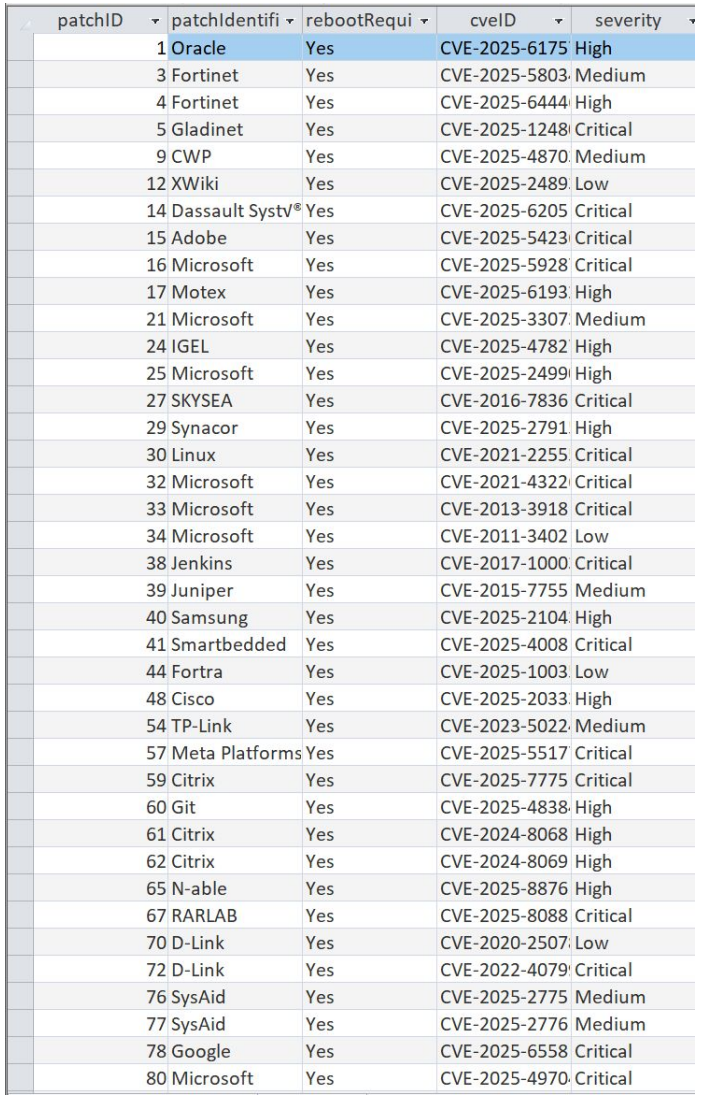
Which patches require a reboot and which vulnerability do they fix?

```
SELECT p.patchID,
p.patchIdentifier,
p.rebootRequired,
    v.cveID, v.severity
FROM Patches AS p
INNER JOIN
Vulnerabilities AS v
    ON
p.vulnerabilitiesID =
v.vulnerabilitiesID
WHERE
p.rebootRequired =
"Yes";
```

- Identifies patches impacting uptime
- Links patches → vulnerabilities
- Important for scheduling maintenance windows

Query 5: Results + Explanation

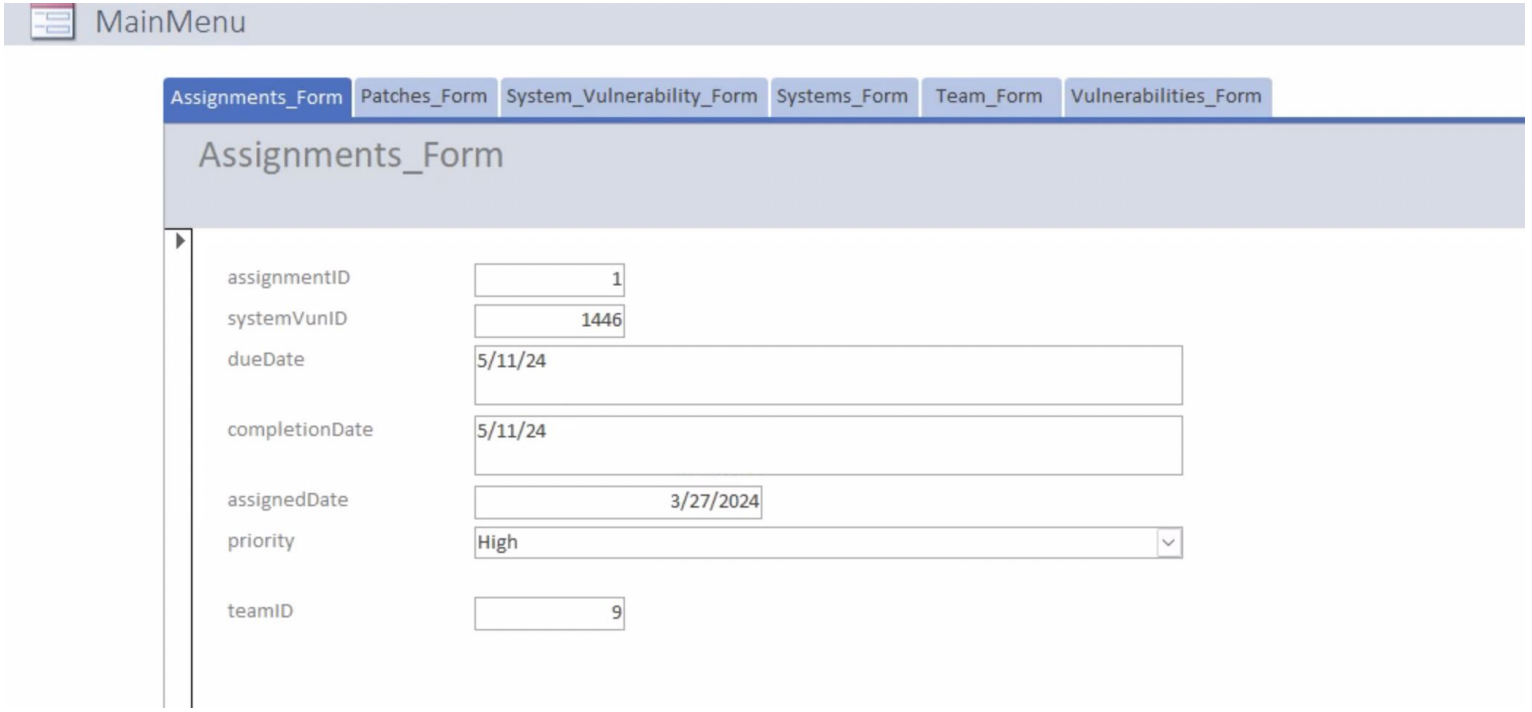
- Helps plan downtime
- Reduces disruption to business operations
- Ensures high-risk patches get proper scheduling



patchID	patchIdentifi	rebootRequi	cveID	severity
1	Oracle	Yes	CVE-2025-6175	High
3	Fortinet	Yes	CVE-2025-5803	Medium
4	Fortinet	Yes	CVE-2025-6444	High
5	Gladinet	Yes	CVE-2025-1248	Critical
9	CWP	Yes	CVE-2025-4870	Medium
12	XWiki	Yes	CVE-2025-2489	Low
14	Dassault Syst®	Yes	CVE-2025-6205	Critical
15	Adobe	Yes	CVE-2025-5423	Critical
16	Microsoft	Yes	CVE-2025-5928	Critical
17	Motex	Yes	CVE-2025-6193	High
21	Microsoft	Yes	CVE-2025-3307	Medium
24	IGEL	Yes	CVE-2025-4782	High
25	Microsoft	Yes	CVE-2025-2499	High
27	SKYSEA	Yes	CVE-2016-7836	Critical
29	Synacor	Yes	CVE-2025-2791	High
30	Linux	Yes	CVE-2021-2255	Critical
32	Microsoft	Yes	CVE-2021-4322	Critical
33	Microsoft	Yes	CVE-2013-3918	Critical
34	Microsoft	Yes	CVE-2011-3402	Low
38	Jenkins	Yes	CVE-2017-1000	Critical
39	Juniper	Yes	CVE-2015-7755	Medium
40	Samsung	Yes	CVE-2025-2104	High
41	Smartbedded	Yes	CVE-2025-4008	Critical
44	Fortra	Yes	CVE-2025-1003	Low
48	Cisco	Yes	CVE-2025-2033	High
54	TP-Link	Yes	CVE-2023-5022	Medium
57	Meta Platforms	Yes	CVE-2025-5517	Critical
59	Citrix	Yes	CVE-2025-7775	Critical
60	Git	Yes	CVE-2025-4838	High
61	Citrix	Yes	CVE-2024-8068	High
62	Citrix	Yes	CVE-2024-8069	High
65	N-able	Yes	CVE-2025-8876	High
67	RARLAB	Yes	CVE-2025-8088	Critical
70	D-Link	Yes	CVE-2020-2507	Low
72	D-Link	Yes	CVE-2022-4079	Critical
76	SysAid	Yes	CVE-2025-2775	Medium
77	SysAid	Yes	CVE-2025-2776	Medium
78	Google	Yes	CVE-2025-6558	Critical
80	Microsoft	Yes	CVE-2025-4970	Critical

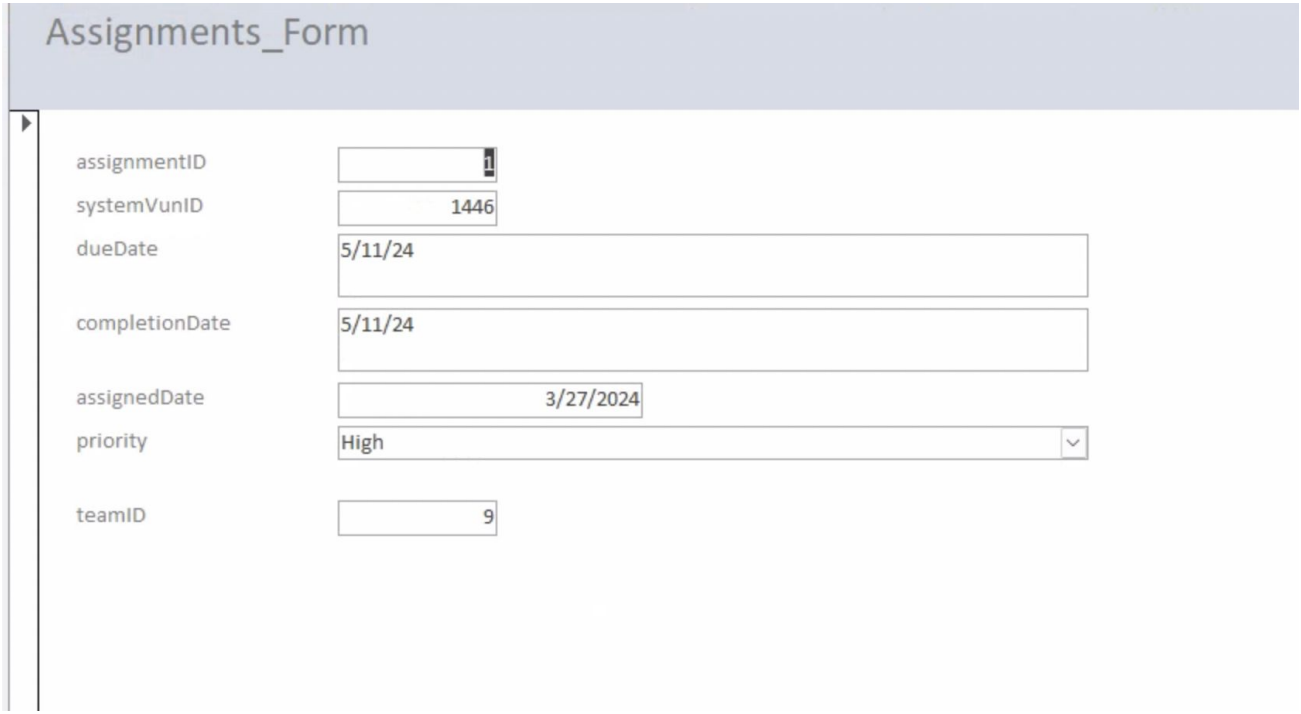
Forms Overview (Purpose of Forms in the VRT System)

- Our MS Access forms provide a user-friendly interface for entering and viewing data across all tables.
- Each form supports a specific part of the vulnerability lifecycle: systems, vulnerabilities, teams, patches, and assignments.
- Forms enforce referential integrity by only allowing valid IDs and linked records (e.g., teamID must exist before assignment).
- Consistent layout and required fields help ensure data quality and accurate reporting.
- **Why forms are important:**
- Reduce data-entry errors
- Enforce 1-M relationships visually
- Allow analysts to quickly review records
- Mimic real enterprise vulnerability-management dashboards



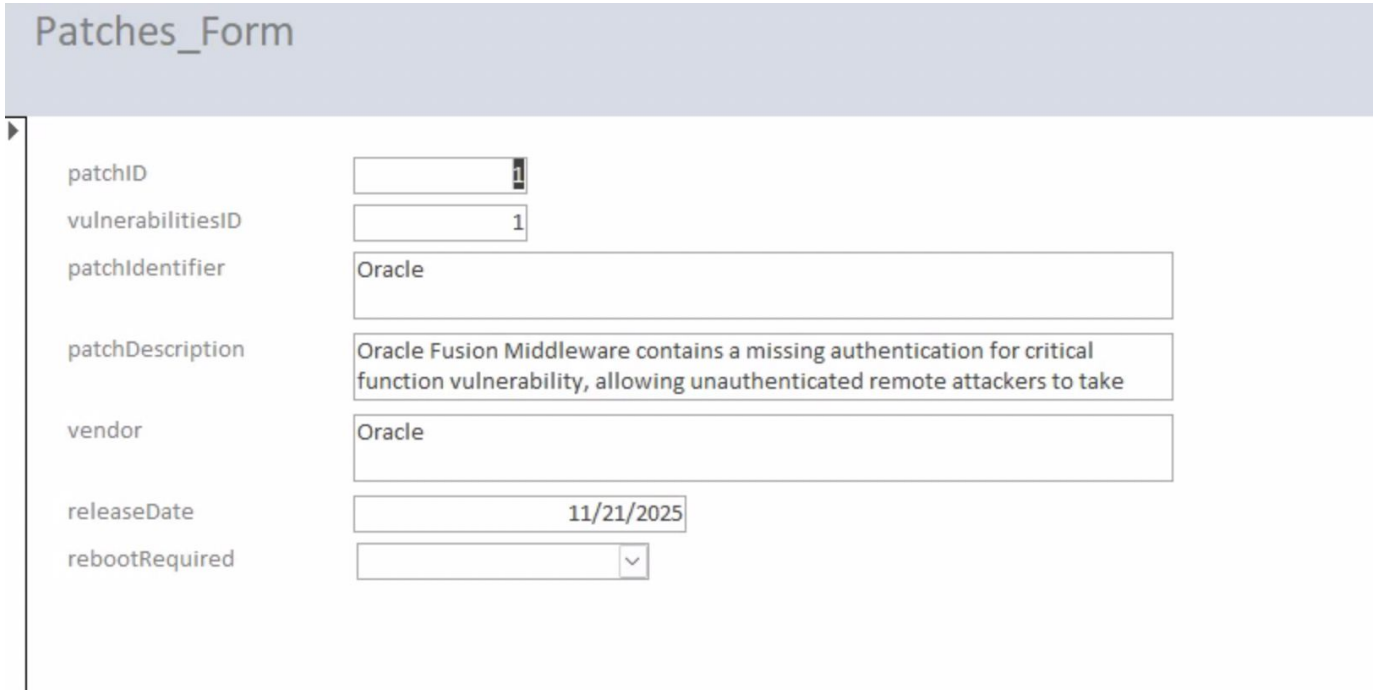
Form 1: Assignments Form (Parent–Child Relationship)

- The *Assignments Form* allows security teams to assign remediation responsibility for each detected system vulnerability.
- **Parent Table:**
- **SystemVulnerability** (systemVunID)
- **Child Table:**
- **Assignments** (assignmentID)
- **What it shows**
- assignmentID
- systemVunID
- dueDate
- completionDate
- assignedDate
- priority
- teamID
- **Why it is useful:**
- Connects responsibly teams to active vulnerabilities
- Tracks deadlines and completion progress
- Helps identify overdue or high-priority remediation tasks
- Supports reporting on workload distribution



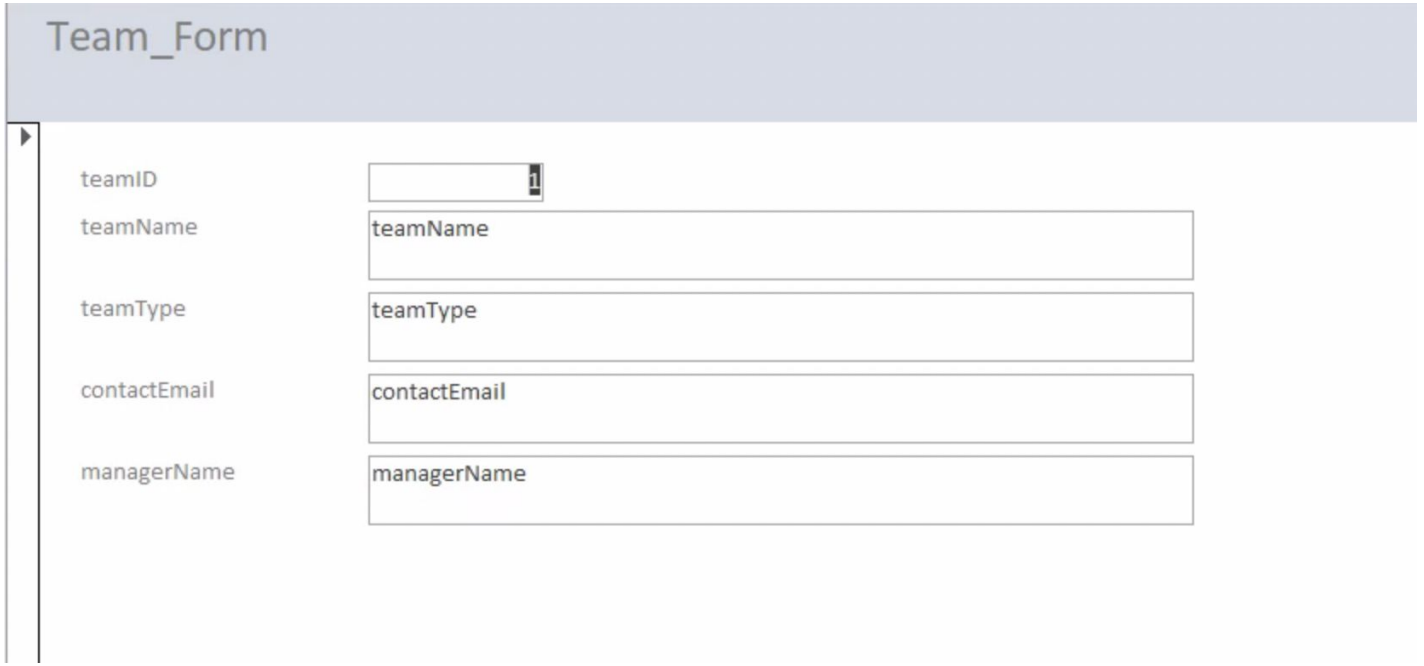
Form 2: Patches Form (Lookup + Detail Form)

- **Purpose:**
  The *Patches_Form* captures all vendor-issued patches mapped to vulnerabilities.
- **Fields shown:**
- patchID
- vulnerabilitiesID (FK → Vulnerabilities table)
- patchIdentifier
- patchDescription
- vendor
- releaseDate
- rebootRequired
- **Why it is useful:**
- Links each patch to the vulnerability it resolves
- Tracks release dates and reboot requirements
- Supports reporting such as:
  Which patches require a reboot?



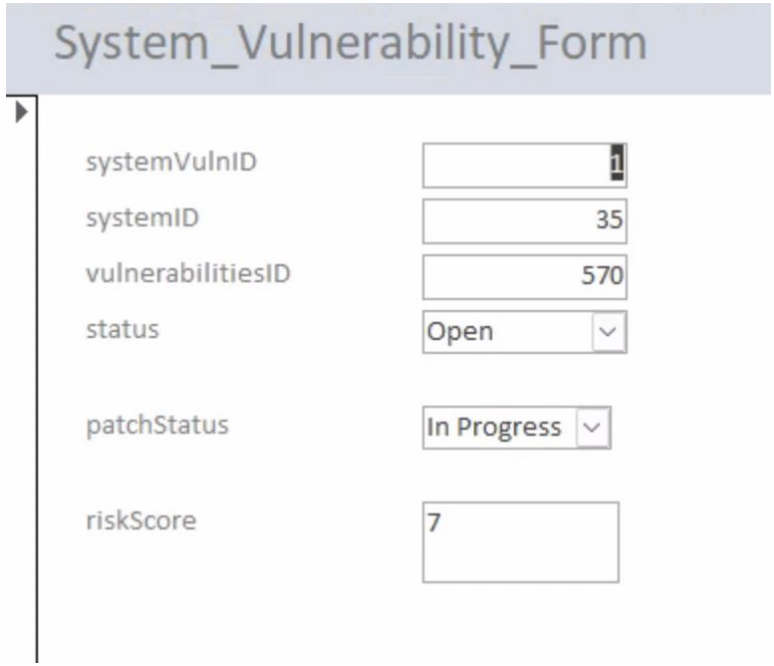
Form 3: Team Form (Single-Table Entry Form)

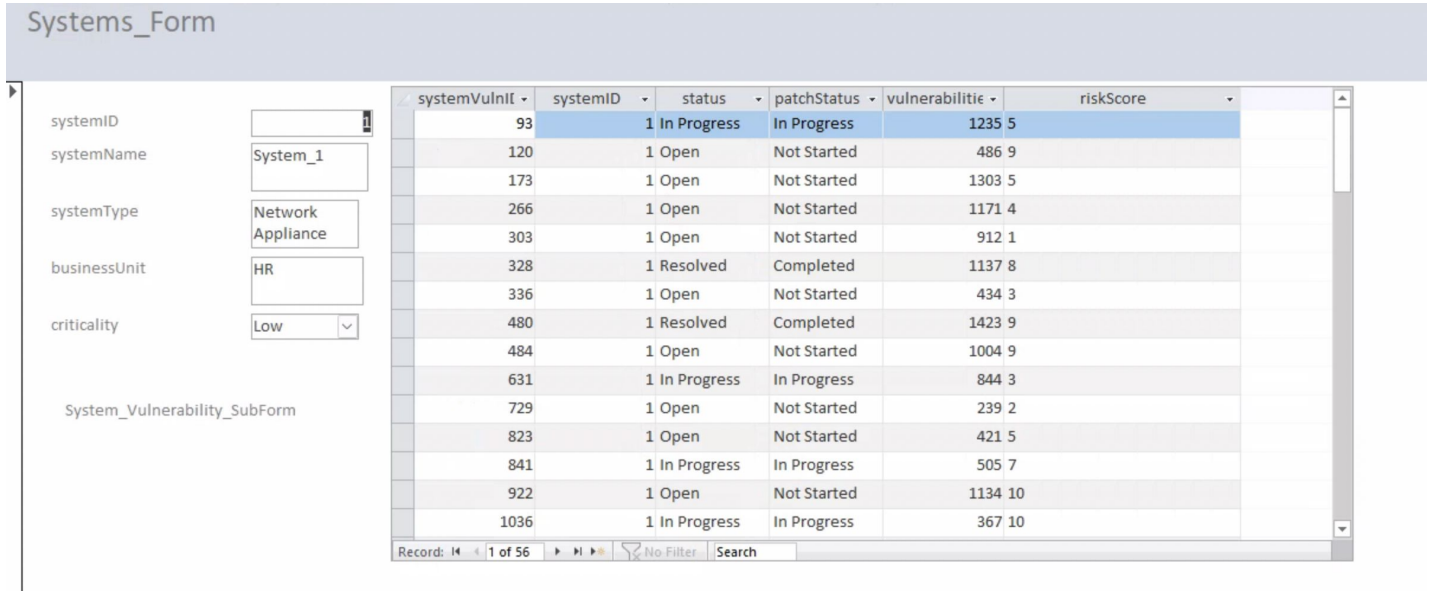
- **Purpose:**
  The *Team_Form* stores information about the remediation teams responsible for addressing vulnerabilities.
- **Fields shown:**
- teamID
- teamName
- teamType
- contactEmail
- managerName
- **Why it is useful:**
- Provides contact and ownership information for vulnerability assignments
- Used as a lookup when creating Assignment records
- Supports Query 3 (assignment count by team)



Form 4: Systems-Vulnerabilities Parent Child Form

- **Description:**
  A hierarchical form based on the 1-M relationship:
- **Systems (1)**
  → **System_Vulnerability (Many)**
- **Why it matters:**
- Enables viewing all vulnerabilities for a selected system
- Supports risk reporting and helps analysts understand which assets have the highest exposure
- Aligns perfectly with your Queries 1 & 2 (risk scores & critical vulnerabilities)





External Dataset Usage

- Used the CISA Known Exploited Vulnerabilities (KEV) Catalog
  - Real-world dataset of actively exploited vulnerabilities
- Imported KEV CSV data into our Vulnerabilities table
- Filled fields such as: CVE ID, title, description, severity, and dates
- Adds realism by reflecting true cybersecurity threats rather than made-up sample data

Challenges & Future Enhancements

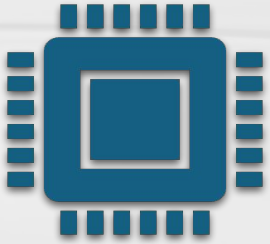
Challenges:

Designing a normalized schema for multiple interconnected tables

Balancing practical cybersecurity fields with database simplicity

Ensuring proper PK/FK constraints and relational accuracy

Managing data variety and formatting from NVD/CVE datasets



Future Improvements:

Add automated vulnerability alerts

Integrate dashboards (Power BI / Tableau)

Add user authentication and role-based access control

Build automated patch compliance alerts

Add CVSS scoring breakdowns and exploit likelihood

Team Roles & Individual Contributions

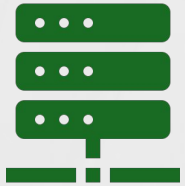
Masumul Mozumder

Led the overall project structure and direction

Designed the RDM (tables, keys, relationships)

Created the SQL queries and analysis

Prepared external dataset integration (CISA KEV import)



Rasha Hoda

Entered and validated large parts of the dataset in Access

Built several Access forms (parent–child + data entry forms)

Helped populate tables based on RDM structure

Assisted with reviewing queries and formatting the database

Bushra Paracha

Supported data entry across multiple tables

Assisted with initial RDM brainstorming and refinement

Contributed to form population and minor table adjustments

Helped test queries and verify table relationships

Conclusion & Key Takeaways

Our VRT system shows how relational databases can simplify cybersecurity operations by connecting vulnerabilities, impacted systems, responsible teams, and patch activity--all in one place.

Key Takeaways:

- Risk Visibility: Quickly see which systems face the highest risk
- Operational Tracking: Monitor patch progress and vulnerability status
- Team Coordination: Understand workload distribution and assignments
- Real Data: Supports analysis using real-world vulnerability datasets
- Scalable Design: Can grow with more systems, data sources, or security features