Problem 12. Zero-Trust Access Control Model Simulator Problem Statement

● Demonstrate and test Zero Trust security models. More About the Problem

● Legacy perimeter security is ineffective.

● Organizations struggle to adopt Zero Trust.

What Companies Are Doing Now

● Zero Trust frameworks.

● Identity-based access control.

● Limited simulation or visualization tools.

PROBLEM STATEMENT (REFRAMED, HUMAN & SIMPLE)

The Hidden Weakness in Modern Cybersecurity

Organizations today use:

- strong passwords
- multi-factor authentication
- Zero Trust security
- AI threat detection

Yet **major security incidents still happen**.

Why?

Because **most breaches are not caused by hackers breaking systems — they are caused by humans making mistakes.**

Examples:

- An employee shares the wrong file in a hurry
- A stressed admin approves access without reading
- A tired user downloads sensitive data at the wrong time
- During incidents, people panic and act fast

These actions are **not malicious**, but they cause **serious security damage**.

⚠️ The Core Problem

Current security systems **treat all authenticated users the same**, regardless of:

- stress
- fatigue
- rush
- cognitive overload
- abnormal behavior caused by pressure

Security systems check:

“Who are you?”
“What device are you using?”

They do **not** check:

“Are you in a safe state to perform this action right now?”

This creates a **blind spot where human mistakes turn into security incidents**.

🔍 WHY EXISTING SOLUTIONS FAIL

Traditional Access Control

- Access based on role
- One-time decision at login

❌ No awareness of how the user behaves later

Zero Trust Security

- Continuous verification
- Strong focus on identity and device

❌ Assumes humans are always careful and rational

Data Loss Prevention (DLP)

- Detects suspicious actions

❌ Mostly reacts **after** a mistake

AI Threat Detection

- Focuses on external attackers

❌ Ignores internal human error

🚨 The Gap

Cybersecurity protects systems and networks, but not human moments of failure.

This gap is **real, costly, and currently unaddressed**.

🟢 OUR IDEA (REFRAMED SOLUTION)

SentinelMind

Human-Aware Zero Trust Security

🧠 What SentinelMind Does (Plain Language)

SentinelMind makes security decisions based not only on identity, but also on how reliable the human is at that moment.

It understands:

- when a user is calm and consistent
- when a user is rushed or overloaded
- when a user's behavior becomes risky

And it **adjusts access accordingly**.

🛠️ How It Works (Easy to Visualize)

SentinelMind observes **behavior inside the system**, such as:

- sudden bursts of activity
- repeated access attempts
- unusual action sequences
- rapid privilege changes
- abnormal download patterns

From this, it calculates a **Human Reliability Score**.

This is **not surveillance**.
No personal or health data is used.
Only system interaction patterns.

🔄 Dynamic Access Control

Human Reliability	System Action
Stable & focused	Normal access
Rushed / overloaded	Extra confirmation
High risk	Temporary restriction

The system **slows users down when mistakes are likely**, instead of blocking them blindly.

🧪 WHY A SIMULATOR IS IMPORTANT

Instead of building a hidden backend tool, we built a **simulator** that:

- visually shows **how access decisions change**
- demonstrates **what happens during human error**
- allows judges to test “what-if” scenarios

Example:

Same user, same login — different behavior → different access

This makes Zero Trust **understandable and explainable**.

🌍 WHY THIS MATTERS IN THE REAL WORLD

- Prevents accidental insider breaches
- Reduces damage during high-pressure situations
- Protects data **without blaming users**
- Makes Zero Trust more realistic and humane

ONE-LINE SUMMARY (VERY IMPORTANT)

SentinelMind secures systems by protecting them from human moments of failure, not just cyber attacks.

INNOVATION

- No existing access-control system measures **human reliability**
- No Zero Trust model adapts to **cognitive risk**
- This approach bridges **cybersecurity + human behavior**

That combination is rare — and powerful.

🧩 SentinelMind – Technology Stack

Frontend (Simulator + Visualization)

🎨 Purpose

- Visualize **human behavior → risk → access decision**
- Let judges **interact live** with scenarios

🛠️ Stack

- **React.js** – interactive UI
- **Tailwind CSS** – clean, fast styling
- **D3.js / Recharts** – risk score graphs, timelines

💡 Why

- React = industry standard
- Visual explanation > raw code (huge judge advantage)

Backend (Decision & Policy Engine)

🧠 Purpose

- Calculate **Human Reliability Score (HRS)**
- Apply Zero Trust policies dynamically

🛠️ Stack

- **Node.js (Express)** *or* **Python (FastAPI)**
- **Policy Engine (custom rule + weight-based logic)**

💡 Why

- Simple logic → easy to explain
- Fast response for live simulation

Human Reliability Scoring Engine (STAR FEATURE)

🔬 Purpose

- Detect **cognitive risk from behavior patterns**

🛠️ Stack

- **Python** (for scoring logic)
- **Scikit-learn** (optional, lightweight ML)
- **Rule + anomaly hybrid model**

Signals used:

- Action frequency
- Access sequence deviation
- Sudden permission changes
- Failed attempt bursts

⚠️ No personal data. No biometrics.

Event & Behavior Tracking

📊 Purpose

- Capture **user actions** for analysis

🛠️ Stack

- **Kafka (optional)** or **Redis Streams**
- **JSON-based event logging**

💡 Why

- Simulates real enterprise telemetry
- Easy to scale later

Access Control & Zero Trust Layer

🔐 Purpose

- Enforce **dynamic access decisions**

🛠️ Stack

- **Open Policy Agent (OPA)** *(optional but powerful)*
- **JWT-based access tokens**
- **Policy-as-code (Rego / JSON rules)**

💡 Why

- Industry-grade Zero Trust concept
- Judges recognize OPA instantly

Simulator Mode (Demo Engine)

🎮 Purpose

- “What-if” testing for judges

🛠️ Stack

- **Mock user profiles**
- **Synthetic behavior generator**
- **Scenario toggles (stress, rush, attack)**

💡 This is what makes you stand out

Data Storage

🗄️ Purpose

- Store events, policies, simulations

🛠️ Stack

- **PostgreSQL** – structured data
- **MongoDB** – behavior logs
- **Redis** – real-time scoring cache

AI / Intelligence (Optional but impressive)

🤖 Purpose

- Pattern detection
- Risk trend analysis

🛠️ Stack

- **Isolation Forest** (anomaly detection)
- **Time-series analysis**
- **Explainable AI (SHAP-style outputs)**

Judges LOVE explainability.

Deployment & Security

🚀 Purpose

- Show production readiness

🛠️ Stack

- **Docker** – containerized services
- **Firebase / AWS / GCP** – hosting
- **HTTPS + OAuth 2.0**

.

🧠 APPROACH

How SentinelMind Works (High Level)

1. **Observe behavior** (inside the system only)
2. **Calculate Human Reliability Score (HRS)**
3. **Feed HRS into Zero Trust decision engine**
4. **Adapt access in real time**
5. **Explain every decision clearly**

Not blocking users — **protecting systems during risky human moments**

USE CASES (Very Important)

🟢 Use Case 1: Accidental Insider Data Leak

Scenario:
An employee under deadline pressure starts downloading many sensitive files rapidly.

Traditional system:
Allows access (user is authenticated).

SentinelMind:

- Detects abnormal burst behavior
- HRS drops
- Temporarily limits bulk downloads
- Adds confirmation friction

✅ **Leak prevented before it happens**

🟢 Use Case 2: Compromised Credentials

Scenario:
An attacker logs in using stolen credentials.

Traditional system:
Login succeeds → full access.

SentinelMind:

- Detects unfamiliar behavior patterns
- Abnormal action sequence
- HRS immediately low
- Access restricted to minimal scope

✅ **Damage contained, even after breach**

🟢 Use Case 3: High-Stress Incident Response

Scenario:
During a cyber incident, admins act fast and make risky changes.

Traditional system:
Full admin power, high risk of mistakes.

SentinelMind:

- Detects panic-driven behavior
- Introduces step-up verification
- Delays irreversible actions

✅ **Prevents catastrophic human errors**

🟢 Use Case 4: Government / Defense Environment

Scenario:
Authorized user accesses classified systems at unusual times from a trusted device.

Traditional system:
Access granted.

SentinelMind:

- Context + behavior evaluated
- Partial access granted
- Sensitive operations restricted

✅ **Security without full denial**

DEPENDENCIES (Be Honest – Judges Respect This)

🔗 Technical Dependencies

- Access to **system activity logs**
- User action telemetry (API calls, file access events)
- Integration with authentication layer (JWT / OAuth)
- Policy engine (custom or OPA)

All are **standard enterprise components**

🔗 Non-Technical Dependencies

- Clear organizational policies
- Ethical use guidelines

- Transparency to users

No personal data, no biometrics, no health inference

SHOW-STOPPERS (Critical Section – DO NOT SKIP)

Judges WILL look for this.

⚠️ Show-Stopper 1: “Isn’t human behavior subjective?”

Reality:
Yes — which is why SentinelMind **does not make binary decisions**.

Mitigation:

- Uses **risk ranges**, not labels
- Only adjusts *sensitivity of actions*
- Human always stays in the loop

⚠️ Show-Stopper 2: False Positives (Blocking real users)

Risk:
Legitimate users may get restricted during high workload.

Mitigation:

- Gradual response (slow down, not lock out)
- Temporary restrictions
- Clear explanations shown to user

⚠️ Show-Stopper 3: Privacy Concerns

Risk:
Fear of monitoring employees.

Mitigation:

- No personal data
- No biometric data
- No external behavior tracking
- Only system-level actions

⚠️ Show-Stopper 4: Adoption Resistance

Risk:
Organizations resist change.

Mitigation:

- Simulator demonstrates value before deployment
- Policy-controlled rollout
- Works alongside existing Zero Trust systems

WHY THIS IS STILL FEASIBLE (Important)

- Fully **simulatable** for hackathon
- No real user data required
- No heavy ML training
- Explainable logic
- Clear demo flow

🔥 “WHY NOW?” (VERY IMPORTANT)

Judges ask: *Why does this matter TODAY?*

Add a slide titled:

Why Existing Security Still Fails in 2025

Include:

- 80–90% breaches involve human error
- Zero Trust protects systems, not human behavior
- AI increases speed → **mistakes become costlier**

💡 This makes your idea feel **urgent**, not academic.

⭐ STAR FEATURE SLIDE (Single Focus)

Title:

What No Existing System Does

One bold sentence in center:

“Security systems do not measure human reliability in real time.”

Then:

- ❌ Identity only
- ❌ Device only
- ❌ Network only
- ✅ **Human behavior**

Judges remember **one big idea**, not 10 features.

🧠 “BEFORE vs AFTER” SLIDE (Golden)

Split slide in half.

Before SentinelMind

- Login once → full trust
- Human panic ignored
- Mistakes = breaches

After SentinelMind

- Continuous trust re-evaluation
- Human risk detected early

- Damage contained

This slide alone can win you points.

🎮 LIVE DEMO FLOW (Even if demo is small)

Title:

How Judges Can Test SentinelMind

Show 3 steps:

1. Select user + scenario
2. Change behavior (rush / normal)
3. Watch access change live

Judges love **control**.

🧪 FAILURE HANDLING SLIDE (Most Teams Skip This)

Title:

What Happens When Things Go Wrong

Include:

- False positives handled by gradual restriction
- Users never permanently locked out
- Full transparency of decisions

This shows **maturity**.

⚖️ ETHICS & PRIVACY SLIDE (Big Trust Builder)

Title:

Built for Trust, Not Surveillance

Bullets:

- No personal data
- No biometrics
- No health inference
- Behavior only inside system

Judges are sensitive to this.

📊 IMPACT METRICS (Hypothetical but Logical)

Title:

Expected Impact

Example metrics:

- ↓ Accidental insider leaks
- ↓ Damage after credential compromise
- ↑ Decision explainability
- ↑ User trust

Even **estimated metrics** help.

🏗️ DEPLOYMENT PATH (Reality Check)

Title:

From Simulator to Real World

Stages:

1. Simulator (hackathon)
2. Pilot in SOC
3. Enterprise integration

Shows you thought **beyond the event**.

🧩 COMPETITOR COMPARISON (BUT VERY CAREFUL)

Simple table:

Feature	Traditional Zero Trust	SentinelMind
Identity-based	✅	✅
Device context	✅	✅
Human reliability	❌	✅
Explainable decisions	❌	✅

🔁 SentinelMind — Complete Prototype Workflow (User Perspective)

🟦 ACTOR TYPES IN PROTOTYPE

Your prototype will clearly show **3 roles**:

1. **User** (employee / admin)
2. **System (SentinelMind Engine)**
3. **Security Admin / Judge (viewer mode)**

🟢 PHASE 1: User Entry (Normal Day)

Login Screen

What user sees:

- Standard login (username + password)
- Optional MFA

👉 Nothing fancy. This is intentional.

Judge takeaway:

"This integrates with existing systems."

🟢 PHASE 2: Continuous Behavior Observation (Invisible to User)

Background Behavior Tracking

What happens silently:

- Every action generates an event:
  - File open
  - Download
  - Permission request
  - API call
  - Time between actions

What user sees:
Nothing. No alerts. No popups.

Judge takeaway:

"Not intrusive. No surveillance vibes."

🟡 PHASE 3: Human Reliability Scoring (Core Engine)

Human Reliability Score (HRS) Calculation

System continuously computes:

- Action speed
- Sequence deviation
- Volume spikes
- Error bursts

Internal output:

HRS = 78 (Stable)

User experience:
Full access continues normally.

🟠 PHASE 4: Risky Human Moment Begins (Key Moment)

Behavior Change (Simulated or Real)

Example:

- User starts downloading many sensitive files quickly
- Tries accessing restricted folder
- Requests privilege escalation

System detects:

- Sudden spike
- Unusual sequence

HRS drops:

HRS = 42 (Unreliable)

🔴 PHASE 5: Dynamic Access Adjustment (STAR FEATURE)

Access Decision Changes

Instead of blocking everything:

User sees:

- Bulk download disabled
- Sensitive action requires confirmation
- Short delay added before irreversible action

Clear explanation shown:

“Access limited temporarily due to unusual activity pattern.”

Judge takeaway:

“This prevents mistakes without punishing users.”

🟣 PHASE 6: Explainability Layer (VERY IMPORTANT)

Why Did This Happen? (Explainable AI UI)

User / Judge can click “Why?”

System shows:

- New device detected
- Rapid action frequency
- Unusual access path

Graphical breakdown:

+20 Rapid actions

+15 Unusual sequence

+10 Privilege escalation

Risk Score: High

This slide wins trust.

🔵 PHASE 7: Recovery & Trust Rebuild

Normalization

Once behavior stabilizes:

- User slows down
- Correct access path
- Time passes

HRS rises back:

HRS = 70

Access automatically returns to normal.

User takeaway:

“System supports me, not controls me.”

🟤 PHASE 8: Admin / Judge View (Simulator Mode)

Simulator Dashboard (For Demo)

Judges can:

- Toggle stress level
- Simulate stolen credentials
- Switch user roles
- Change time/device

And instantly see:

- HRS graph
- Access decisions
- Damage containment

This is where judges engage emotionally.

⚫ PHASE 9: Incident Scenario (Optional Killer Demo)

“Credentials Stolen” Mode

- Attacker logs in
- Behavior totally different

System response:

- Immediate low HRS
- Read-only access
- Sensitive operations blocked

Judge takeaway:

“Even after breach, damage is limited.”

📦 WHAT YOUR PROTOTYPE WILL INCLUDE (CHECKLIST)

UI Screens

- Login screen
- User dashboard
- Access denied / restricted message
- “Why was this action limited?” panel
- Simulator control panel

Backend Logic

- Event logger
- HRS calculator
- Policy engine
- Decision explainer

Visuals

- Risk score meter
- Timeline graph
- Before vs After comparison

⚠️ IMPORTANT: What NOT to Build

❌ Full authentication system
❌ Heavy ML training
❌ Real user monitoring
❌ Complex enterprise integration

Judges care about **concept clarity + demo**, not completeness.