Distributed Systems

Trust and Monitoring in Distributed Systems

WEEK 16: Trust Models, Monitoring, Intrusion Detection, Anomaly Detection, Security Auditing.

Online Lecture Series - 16

Felix Edesa

Addis Ababa Science and Technology University

AASTU

Course Outline: Trust and Monitoring in Distributed Systems (Lecture 16)

Topics We Will Cover

- **Trust Models** — Reputation-based, PKI-based, and hybrid methods.
- **Monitoring Systems** — Observing distributed components via metrics, logs.
- **Intrusion Detection** — Detect malicious activity with signature-based.
- **Anomaly Detection** — Spot unusual patterns in network traffic, logs.
- **Security Auditing** — Audit access, configurations, and transactions.
- **Practical Applications** — Microservices monitoring, cloud auditing.
- **Industrial Case Studies** — Lessons from real-world large-scale deployments.

AASTU

Trust Models in Distributed Systems — Overview

Overview

- Trust models define how systems evaluate reliability and authenticity before sharing resources.
- Enable secure collaboration across untrusted or partially trusted nodes.
- Major categories: Reputation-Based, PKI-Based, and Hybrid Trust Models.
- Used in IoT networks, blockchain systems, and microservice ecosystems.

Industrial Example

- Google Cloud services apply layered trust verification for APIs.
- IoT smart city platforms evaluate sensor trust dynamically.

AASTU

Reputation-Based Trust — Concept

Key Concepts

- Builds trust from observed behavior and peer feedback.
- Each node maintains a reputation score based on reliability.
- Dynamic — trust changes with consistent good or bad actions.
- Supports decentralized decision-making without central authority.

Example

- In peer-to-peer storage, nodes with high data availability gain higher trust.
- Malicious or inactive peers lose credibility and are isolated.

AASTU

Reputation-Based Trust — Implementation in Systems

Approach

- Trust score = weighted sum of direct experience and peer recommendations.
- Uses feedback aggregation, Bayesian inference, or fuzzy logic models.
- Protects networks from Sybil or denial-of-service attacks.

Industrial Example

- Blockchain consensus nodes gain trust via consistent validation.
- Ride-sharing apps like Uber use ratings to detect bad drivers or users.

AASTU

Reputation-Based Trust — Challenges

Limitations

- Vulnerable to collusion or false feedback manipulation.
- Reputation data can be spoofed or biased by attackers.
- Trust convergence may be slow in large networks.
- Requires secure sharing and validation of trust updates.

Solutions

- Use cryptographically signed feedback.
- Integrate machine learning to detect false ratings.

AASTU

PKI-Based Trust — Concept

How It Works

- Based on public key cryptography and digital certificates.
- Certificate Authorities (CAs) validate identities.
- Enables mutual authentication between clients and servers.
- Used for encryption, signing, and secure communication.

Practical Example

- HTTPS websites rely on CA-signed certificates.
- Corporate VPNs use PKI to authenticate users.

AASTU

PKI-Based Trust — Real-World Application

Industrial Use

- Cloud providers issue certificates for API-to-API trust (mTLS).
- Governments use PKI for e-identity and digital signatures.
- Organizations automate certificate renewal using tools like Let's Encrypt.

Challenge

- Central CA compromise can break global trust.
- Requires periodic certificate revocation and audit.

AASTU

Hybrid Trust Models — Concept

Overview

- Combines static (PKI) trust with dynamic (reputation) trust mechanisms.
- Balances identity verification and behavioral evaluation.
- Provides flexible and adaptive trust in dynamic environments.
- Often used in multi-cloud and blockchain-based systems.

Practical Example

- Blockchain validator nodes: PKI for identity, reputation for reliability.
- IoT devices: certificates + behavioral monitoring for trust scoring.

AASTU

Hybrid Trust in Microservices Environment

Implementation

- Each service authenticates using PKI (service certificates).
- Runtime behavior (latency, uptime, error rate) used to calculate service trust score.
- Central trust manager aggregates and adjusts service-level permissions.

Example

- AWS microservices apply mTLS + runtime trust analytics.
- Helps isolate faulty or compromised components dynamically.

AASTU

Trust Models — Comparative Analysis

Comparison

- **Reputation-Based:** Dynamic, decentralized, but may be manipulated.
- **PKI-Based:** Strong identity assurance, centralized dependency.
- **Hybrid:** Combines both for adaptive reliability.

Example

- Blockchain: Hybrid model ensures both key validity and behavioral trust.
- IoT networks: balance between cryptographic identity and performance record.

AASTU

Recommendations

- Use PKI for strong identity; complement with continuous behavior-based scoring.
- Automate trust recalculation using machine learning.
- Regularly audit certificates and trust logs.
- Explore decentralized PKI (DPKI) for removing central trust anchors.

Emerging Trends

- AI-driven trust management systems.
- Quantum-safe certificates for future cryptography.
- Cross-domain trust federations in cloud-native ecosystems.

AASTU

Monitoring Systems — Overview

Concept

- **Monitoring systems** continuously collect data from distributed components.
- They ensure **visibility**, **performance tracking**, and **incident response**.
- Use a combination of **metrics**, **logs**, and **traces**.
- Enable engineers to detect problems before they affect users.
- Form the foundation of **observability** in modern systems.

Example

- Prometheus collects node metrics every 15 seconds.
- Grafana visualizes CPU, memory, and network usage.
- Alerts notify teams when CPU exceeds 90%.

Importance of Monitoring in Distributed Systems

Key Benefits

- **Early Detection:** Identifies latency or performance degradation.
- **Accountability:** Provides detailed logs for auditing and debugging.
- **Reliability:** Keeps services running smoothly across nodes.
- **Optimization:** Helps balance workloads and resource usage.
- **Security:** Detects suspicious patterns like repeated login failures.

Practical Case

- Netflix uses monitoring to detect failures in microservices.
- Monitoring dashboards show traffic spikes and API latency.

Metrics-based Monitoring

Concept

- **Metrics** are numeric indicators of system performance (CPU, memory, I/O).
- Collected periodically and stored in a **time-series database**.
- Enables trend analysis and performance forecasting.
- Metrics can trigger **automated alerts** using predefined thresholds.
- Common tools: **Prometheus**, **Grafana**, **Datadog**.

Example

- Alert rule: **CPU ¿ 80% for 5 minutes → send Slack alert**.
- Grafana graph shows 7-day CPU trend across servers.

Log-based Monitoring

Concept

- **Logs** capture detailed information about system events and errors.
- Used for debugging, forensic analysis, and user activity tracking.
- Logs are often aggregated into centralized systems for analysis.
- Key approaches: **Structured logging**, **Log rotation**, **Retention policies**.
- Popular tools: **ELK Stack (Elasticsearch, Logstash, Kibana)**, **Fluentd**.

Example

- Log entry: "POST /api/v1/orders - 500 Internal Server Error".
- Kibana dashboard shows 50 similar errors in the last hour.

Distributed Tracing for Deep Insight

Purpose

- **Distributed tracing** connects logs and metrics across microservices.
- Helps follow the lifecycle of a single request across multiple systems.
- Enables detection of bottlenecks between service calls.
- Common standards: **OpenTelemetry**, **Jaeger**, **Zipkin**.

Example

- Trace ID links front-end request to database query.
- Jaeger visualizes service-to-service latency (API $\rightarrow$ DB).

AASTU

Designing a Monitoring Architecture

Core Components

- **Data Collection:** Agents gather metrics/logs from nodes.
- **Storage Layer:** Databases store and index collected data.
- **Analysis:** Algorithms detect trends, thresholds, and anomalies.
- **Visualization:** Dashboards communicate insights to engineers.

Example

- Architecture: Node Exporter $\rightarrow$ Prometheus $\rightarrow$ Grafana $\rightarrow$ Alertmanager.
- Alerts routed to email or Slack for quick response.

Common Monitoring Challenges

Key Issues

- **Data Overload:** Too many metrics without context.
- **False Positives:** Poorly tuned alert thresholds.
- **Scalability:** Monitoring infrastructure can become a bottleneck.
- **Security:** Logs may contain sensitive user data.

Example

- Solution: Use log sanitization and automated filtering.
- Implement role-based access control on dashboards.

AASTU

Monitoring in Cloud Environments

Cloud Monitoring

- Cloud services provide built-in monitoring tools.
- **AWS CloudWatch**, **Azure Monitor**, **Google Operations**.
- Scales automatically with distributed instances.
- Offers predictive insights using machine learning.

Example

- CloudWatch alarms trigger Lambda functions on high CPU usage.
- Azure Monitor sends email alerts for failed deployments.

AASTU

Monitoring Best Practices

Recommendations

- Focus on **key service-level indicators (SLIs)**.
- Automate alerts and integrate with incident management tools.
- Rotate and archive logs regularly.
- Use **dashboards** for real-time system health overview.
- Combine metrics, logs, and traces for full observability.

Example

- PagerDuty integrated with Prometheus alerts.
- 3-tier dashboard: infrastructure $\rightarrow$ services $\rightarrow$ user-level KPIs.

AASTU

Intrusion Detection — Overview

Concept

- **Intrusion Detection Systems (IDS)** monitor network or host activity to identify suspicious behavior.
- Designed to detect attacks such as malware, port scanning, brute-force, or unauthorized access.
- Operate by analyzing patterns, signatures, and anomalies in traffic or system logs.
- A key layer of defense in distributed and cloud environments.

Example

- A web server IDS detects repeated failed logins from the same IP.
- System flags this as a potential **brute-force attack**.

Signature-Based Detection

Concept

- Uses a database of **known attack patterns or signatures**.
- Compares network packets or logs against stored signatures.
- Ideal for detecting **known threats** such as malware or exploits.
- Fast, reliable, and easy to maintain for well-documented attacks.

Example

- A signature detecting "SQL Injection" searches for patterns like "`OR 1=1`".
- Snort or Suricata triggers alerts when this pattern appears in HTTP requests.

AASTU

Architecture of a Signature-Based IDS

Main Components

- **Sensor/Agent:** Captures network or host activity.
- **Signature Database:** Stores patterns of known attacks.
- **Detection Engine:** Matches live data against signatures.
- **Alert System:** Generates notifications upon detection.
- **Management Console:** Displays and analyzes incidents.

Example

- Snort IDS uses packet sniffers as sensors.
- Alerts are visualized in Kibana via Elastic Stack integration.

Network-Based Intrusion Detection (NIDS)

Description

- Monitors **network traffic** for malicious activity.
- Operates on routers, firewalls, or gateways.
- Detects threats like DoS attacks, port scans, and packet floods.
- Provides real-time visibility into system-wide activity.

Example

- **Snort** captures packets and compares them with signatures.
- Detects ICMP flood patterns matching known attack behavior.

AASTU

Host-Based Intrusion Detection (HIDS)

Description

- Focuses on monitoring individual machines or servers.
- Analyzes system calls, file integrity, and local logs.
- Detects unauthorized modifications or privilege escalation.
- Suitable for cloud VMs and containerized environments.

Example

- **OSSEC** alerts when a root-owned file is modified.
- Detects new unauthorized user creation in Linux system logs.

AASTU

Maintaining and Updating Signatures

Best Practices

- Regularly update signature databases to detect new threats.
- Validate signatures from trusted vendors or open-source communities.
- Remove outdated or redundant rules to reduce noise.
- Use **threat intelligence feeds** for live updates.
- Automate signature updates in large-scale environments.

Example

- Suricata integrates with **Emerging Threats** feed.
- Automatically downloads and applies new detection rules daily.

Limitations of Signature-Based Detection

Challenges

- Cannot detect **zero-day attacks** or unknown threats.
- High dependency on updated signature databases.
- Generates false negatives when attacks deviate slightly from known patterns.
- Can struggle with encrypted traffic inspection.

Example

- New ransomware variant evades detection because its hash doesn't match known signatures.
- Solution: Combine with **anomaly-based detection**.

AASTU

Hybrid Intrusion Detection Approach

Concept

- **Hybrid IDS** combines signature-based and anomaly-based techniques.
- Signatures detect known attacks quickly.
- Anomalies catch deviations that may indicate zero-day exploits.
- Provides balance between accuracy and adaptability.

Example

- A hybrid IDS flags unknown malware due to unusual process behavior.
- Machine learning model confirms the anomaly after cross-check.

AASTU

Case Study — Snort in a Corporate Network

Scenario

- A financial company deploys Snort IDS to monitor internal traffic.
- System detects repeated unauthorized SSH connections.
- Administrator correlates alerts with login failures from an external IP.
- Identified as a targeted brute-force attack.

Outcome

- Attack blocked at firewall.
- IP added to blacklist and report shared with threat intelligence feed.

AASTU

Hands-on Lab — Implementing Signature-Based Detection

Lab Steps

- Install and configure **Snort** on a Linux system.
- Create a custom signature to detect suspicious HTTP GET requests.
- Test by sending simulated attack traffic using **Metasploit**.
- Observe and analyze alerts generated in the Snort console.

Expected Results

- Successful detection of known attack pattern.
- Visualization of alert logs via Kibana or Grafana dashboard.

AASTU

Intrusion Detection — Introduction

Overview

- **Intrusion Detection Systems (IDS)** monitor network or host activities for malicious actions.
- **Goal:** Identify unauthorized access, policy violations, or abnormal behaviors.
- **Types:** Host-based (HIDS) and Network-based (NIDS).
- **Signature-based IDS** focuses on known attack patterns.

Example Systems

- **Snort:** Open-source network IDS.
- **OSSEC:** Host-based IDS with log analysis.
- **Suricata:** Real-time network monitoring and intrusion detection.

AASTU

Signature-Based Detection — Concept

Core Idea

- **Signatures** are unique patterns representing specific attacks.
- IDS compares observed data to these known signatures.
- Effective for **well-known threats**.
- Easy to implement but may fail for **zero-day attacks**.

Practical Example

- A rule to detect SQL Injection:
- `alert tcp any any -> any 80 (content:"' OR 1=1 --"; msg:"SQL Injection"; sid:100001;)`
- **Snort** triggers an alert when it detects this pattern in traffic.

AASTU

Workflow of Signature-Based IDS

Detection Pipeline

- **Step 1:** Capture incoming network packets or logs.
- **Step 2:** Parse data and extract relevant fields.
- **Step 3:** Match data against known attack signatures.
- **Step 4:** Generate alerts and log details for security teams.

Example Tool Flow

- **Snort:** Packet capture → Preprocessor → Detection Engine → Alerting.
- Example alert: **"Possible SSH brute-force detected."**

AASTU

Advantages of Signature-Based IDS

Key Benefits

- **High Accuracy** for known attacks.
- **Low False Positives** due to specific pattern matching.
- **Simple to Deploy** with predefined signature databases.
- **Efficient Resource Usage** compared to anomaly-based methods.

Industry Example

- **Financial Institutions:** Use signature-based IDS to detect common phishing or ransomware attempts.
- Example: Detecting specific **ransomware payload signatures**.

AASTU

Limitations and Challenges

Issues

- Cannot detect **new or unknown attacks** (zero-day).
- Requires **frequent signature updates**.
- **Evasion techniques** like polymorphism or encryption can bypass detection.
- Heavy reliance on **predefined databases**.

Practical Case

- A worm uses encrypted payloads $\rightarrow$ IDS fails to detect.
- **Solution:** Combine with anomaly-based or behavioral IDS.

AASTU

Signature Update and Maintenance

Key Points

- **Regular updates** to signature databases are critical for effectiveness.
- Updates can come from **vendors, community feeds, or internal analysis**.
- Automating updates ensures **continuous protection**.
- Requires **version control** and validation to prevent false positives.

Practical Example

- **Snort** regularly fetches updated rule sets from the community.
- Organizations may create **custom signatures** for internal threats.
- Example: Detecting unauthorized SSH access.

AASTU

Deployment Considerations for Signature-Based IDS

Key Points

- **Placement** is critical: network perimeter, internal segments, or critical hosts.
- **Resource requirements:** CPU, memory for high-throughput traffic analysis.
- **Scalability:** Must handle growing traffic volumes in distributed systems.
- **Integration:** Should work with SIEM (Security Information and Event Management) tools.

Industrial Example

- Deploy **network-based IDS** at the gateway for cloud applications.
- **Host-based IDS** on critical servers for enterprise applications.

AASTU

Combining Signature-Based and Anomaly-Based IDS

Key Concepts

- Signature-based IDS detects **known threats**.
- Anomaly-based IDS detects **unknown or abnormal behavior**.
- **Hybrid approach** provides comprehensive coverage.
- Reduces the likelihood of **false negatives**.

Practical Example

- Detects known malware with signatures.
- Detects unusual login patterns or traffic spikes with anomaly detection.
- Example: **APT (Advanced Persistent Threat)** detection in enterprise networks.

AASTU

Alerting and Response in Signature-Based IDS

Key Points

- IDS generates **real-time alerts** upon detecting a signature match.
- Alerts include **severity, source, destination, and attack type**.
- Integration with **incident response systems** is crucial.
- Automated response options: block IP, terminate session, or quarantine host.

Industrial Example

- **Web application firewall** triggered when SQL injection signature is detected.
- Automated firewall rules block offending IPs.
- Security team reviews alerts for compliance and investigation.

AASTU

Practical Tips for Signature-Based IDS

Recommendations

- Keep signature databases **up-to-date** regularly.
- Combine signature-based with **anomaly-based detection**.
- Continuously monitor **logs, alerts, and performance metrics**.
- Conduct **periodic testing** and tuning to reduce false positives.

Industrial Example

- **Financial institutions:** update signature feeds daily for new threats.
- **Cloud providers:** combine signature and anomaly detection for multi-tenant systems.

AASTU

Security Auditing — Introduction

Key Concepts

- **Security auditing** monitors and reviews system activities.
- Audits include **access control, configurations, and transaction logs**.
- Helps detect **policy violations, misconfigurations, and anomalies**.
- Essential for **compliance** with standards like ISO 27001, PCI-DSS.

Industrial Example

- Cloud providers audit user access logs daily.
- Banks monitor transaction logs for unusual activity.
- Microservices environments track API usage and access patterns.

AASTU

Types of Security Audits

Key Points

- **Access Audits** — who accessed what, when, and from where.
- **Configuration Audits** — verify settings of servers, firewalls, and apps.
- **Transaction Audits** — track data flow and modifications in databases.
- **Compliance Audits** — ensure adherence to legal and corporate standards.

Practical Example

- AWS CloudTrail for tracking user activity in cloud environments.
- Database triggers for recording financial transactions.
- Server configuration snapshots using Ansible or Puppet.

Tools and Techniques

Key Tools

- **SIEM** (Security Information and Event Management) systems: Splunk, ELK.
- **Audit logs** and system event collectors.
- **Configuration management tools**: Ansible, Chef, Puppet.
- **Automated compliance tools** to check against security baselines.

Practical Example

- Use Splunk dashboards to monitor unauthorized login attempts.
- ELK stack visualizes anomalies in API usage.
- Puppet checks server configs against security baselines weekly.

AASTU

Auditing in Cloud Environments

Key Points

- Track user activity across cloud platforms.
- Audit configuration changes and access policies.
- Integrate cloud logs with SIEM for centralized analysis.
- Automated compliance reporting for regulations like GDPR and PCI-DSS.

Practical Example

- AWS CloudTrail logs user actions.
- Azure Monitor collects access and config changes.
- GCP Security Command Center detects misconfigured permissions.

Auditing Blockchain Transactions

Key Concepts

- Transactions are **immutable and verifiable**.
- Every action is timestamped and digitally signed.
- Audit trails allow **regulatory compliance and forensic analysis**.

Practical Example

- Monitor Ethereum smart contract transactions for anomalies.
- Verify token transfers and ownership changes.
- Detect unauthorized contract modifications.

AASTU

Generating Audit Reports

Key Points

- Summarize findings in clear, actionable reports.
- Include **security gaps, misconfigurations, and anomalies**.
- Reports can feed into **management dashboards and SIEM**.
- Supports **decision-making and compliance audits**.

Industrial Example

- Banks generate monthly audit reports for regulators.
- Cloud providers use dashboards for tenant compliance monitoring.
- Enterprises use automated alerts for high-severity issues.

AASTU

Industrial Case Study — Cloud Security Auditing

Scenario

- Multi-tenant cloud environment for enterprise SaaS applications.
- Audit access, configuration changes, and tenant activity logs.
- Integrate findings into SIEM for automated detection.
- Maintain compliance with GDPR, ISO 27001, and SOC2 standards.

Lessons Learned

- Continuous monitoring is essential for multi-tenant security.
- Automated auditing reduces human error.
- Audit trails support forensic investigations and compliance reporting.

AASTU

Questions?

Thank you for your attention!

References

Suggested References

1. M. van Steen and A. S. Tanenbaum, *Distributed Systems*, 4th ed., Version 4.02, February 2024.
2. G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed., Addison-Wesley, 2012.
3. N. Lynch, *Distributed Algorithms*, 2nd ed., Morgan Kaufmann, 1996.
4. R. E. Bryant and D. O'Hallaron, *Computer Systems: A Programmer's Perspective*, 4th ed., Pearson, 2024.

AASTU