Atlas Framework

Ultimate Security Research Platform

VPN + Axiom + Ars0n OSINT + Distributed Collection

Version: 1.0.0
Date: October 18, 2025
Author: Atlas Project Team

Executive Summary

Atlas Framework ist die ultimative, vollständig integrierte Security Research Platform, die vier kritische Komponenten in einer einzigen, leistungsstarken Lösung vereint:

1. **VPN Infrastructure Management** - WireGuard-basierte Multi-Cloud-VPN-Orchestrierung
2. **Axiom Fleet Orchestration** - Distributed computing across AWS, GCP, und DigitalOcean
3. **Ars0n OSINT Framework** - Automated reconnaissance und vulnerability scanning
4. **Distributed Data Collection** - Scalable scraping und API collection

Kernfunktionen

Unified Dashboard mit Interactive World Map

- Echtzeit-Visualisierung aller Infrastruktur-Komponenten
- VPN-Nodes (grüne Marker), Axiom-Instanzen (blaue Marker)
- OSINT-Jobs (orange), Collection-Workers (lila)
- Animierte Verbindungslinien zwischen Ressourcen

Multi-Cloud VPN Management

- One-click WireGuard VPN deployment
- QR-Code-Generierung für Mobile-Setup
- Per-instance VPN-Konfiguration
- Traffic-Monitoring und Client-Management

Axiom Fleet Orchestration

- Deploy fleets across AWS, GCP, DigitalOcean
- Auto-scaling und spot instance support
- Pre-installed security tools (nmap, nuclei, ffuf, etc.)
- Distributed command execution

Ars0n OSINT Integration

- Subdomain enumeration (Amass, Subfinder, Sublist3r)
- Vulnerability scanning (Nuclei, Nmap)
- Cloud asset discovery
- Automated reconnaissance workflows

Distributed Collection Engine

- Generic job scheduler für any workload
- Multi-strategy distribution (round-robin, least-loaded, regional)
- Rate limiting und IP rotation
- Result aggregation und export

Cost Tracking & Analytics

- Real-time cost breakdown per provider/service
- Usage statistics und trends
- Optimization recommendations
- Budget alerts

Systemarchitektur

High-Level Overview

```
┌─────────────────────────────────────────────────────────────┐
│                CDN Layer (CloudFront/Cloudflare)            │
│                    yourdomain.com                           │
└──────────────────────────┬──────────────────────────────────┘
                           │
          ┌────────────────┴────────────────┐
          │                                 │
┌─────────▼─────────┐            ┌──────────▼──────────┐
│  React Frontend   │            │  Kong API Gateway   │
│  (TypeScript)     │            │  Rate Limiting      │
└─────────┬─────────┘            └──────────┬──────────┘
          │                                 │
          └────────────────┬────────────────┘
                           │
          ┌────────────────┴────────────────┐
          │                                 │
┌─────────▼─────────┐            ┌──────────▼──────────┐
│  VPN Stack        │            │  Axiom Stack        │
│  WireGuard        │            │  SSH Control        │
└─────────┬─────────┘            └──────────┬──────────┘
          │                                 │
          ├────────────────┬────────────────┤
          │                │                │
```

```
┌─────▼────┐ ┌────▼────┐   ┌─────▼───────┐
│VPN      │ │Ars0n   │   │Collection│
│Nodes    │ │OSINT   │   │Workers   │
│Multi-   │ │Scanner│   │Puppeteer │
│Cloud    │ │        │   │          │
└────────────┘  └─────────┘   └─────────────┘
     │           │              │
     └──────────────┴───────────────┘
                 │
        ┌──────────▼──────────┐
        │   PostgreSQL 15   │
        │   + Redis 7       │
        └────────────────────┘
```

Technology Stack

Frontend

- React 18 mit TypeScript
- Tailwind CSS (dark mode optimized)
- Leaflet.js für interactive maps
- Recharts für data visualization
- Zustand für state management
- React Query für API caching

Backend

- Node.js 18+ mit Express.js
- Socket.io für WebSocket real-time updates
- JWT authentication
- PostgreSQL 15 mit comprehensive schema
- Redis 7 für caching und queues

API Gateway

- Kong Gateway 3.5
- Rate limiting (100 req/15min)
- Authentication plugins
- Service routing

Cloud Integration

- AWS SDK (EC2, CloudFront, S3, RDS)
- GCP SDK (Compute Engine, Cloud CDN)
- DigitalOcean API
- SSH2 für Axiom-Controller-Communication

DevOps

- Docker und Docker Compose
- Terraform für Infrastructure-as-Code
- Prometheus + Grafana monitoring (optional)
- Nginx reverse proxy

Database Schema Highlights

Core Tables

- `users` - User accounts mit JWT authentication
- `cloud_providers` - Encrypted cloud credentials
- `vpn_nodes` - WireGuard VPN infrastructure
- `vpn_clients` - Client configurations
- `axiom_fleets` - Fleet definitions
- `axiom_instances` - Individual instances
- `osint_jobs` - Reconnaissance jobs
- `osint_findings` - Subdomains, vulnerabilities
- `collection_jobs` - Data collection tasks
- `collection_results` - Scraped data
- `cost_records` - Cost tracking
- `audit_logs` - Complete audit trail

346 Zeilen comprehensive schema mit indexes, views, und constraints.

Installation & Deployment

Prerequisites

System Requirements

- Docker 20.10+ und Docker Compose 2.0+
- Node.js 18+
- PostgreSQL 15+ (via Docker)
- Redis 7+ (via Docker)
- 4GB RAM minimum, 8GB recommended
- 20GB disk space

Cloud Accounts

- AWS Account mit API credentials

- GCP Account mit service account
- DigitalOcean API token (optional)

Axiom Setup

- Axiom controller deployed on EC2/GCE
- SSH key access configured

Quick Start mit Docker

```
# 1. Clone repository
git clone &lt;repository-url&gt;
cd atlas-framework

# 2. Configure environment
cp .env.atlas.example .env.atlas
nano .env.atlas  # Edit with your credentials

# 3. Generate secure keys
JWT_SECRET=$(openssl rand -base64 32)
ENCRYPTION_KEY=$(openssl rand -base64 32)

# 4. Setup Axiom SSH key
mkdir -p ./keys
ssh-keygen -t rsa -b 4096 -f ./keys/axiom_rsa -N ""
# Add public key to Axiom controller

# 5. Deploy all services
chmod +x deploy_atlas.sh
./deploy_atlas.sh

# 6. Access dashboard
# Frontend: http://localhost:3000
# API Gateway: http://localhost:8000
# Kong Admin: http://localhost:8001
```

Manual Installation

Backend Setup

```
cd backend
npm install
createdb atlas
psql -d atlas -f ../atlas_schema.sql
npm start
```

Frontend Setup

```
cd frontend
npm install
npm start
```

Services starten

```
docker-compose -f docker-compose-atlas.yml up -d
```

Production Deployment

1. CDN Setup mit AWS CloudFront

```
cd infra/terraform
terraform init
terraform plan
terraform apply
```

2. CDN Setup mit Cloudflare (kostenlos!)

```
# Deploy Cloudflare Worker
wrangler publish cloudflare-workers/atlas-cdn.js

# Configure DNS
# CNAME: yourdomain.com -&gt; your-worker.workers.dev
# CNAME: api.yourdomain.com -&gt; your-alb.amazonaws.com
```

3. SSL Certificates

```
# AWS: Use ACM (automatic with Terraform)
# Cloudflare: Automatic SSL (free)
# Let's Encrypt: Manual setup
certbot certonly --dns-cloudflare \
  -d yourdomain.com -d *.yourdomain.com
```

4. Security Hardening

- Enable HTTPS-only
- Configure firewall rules (Security Groups)
- Set up VPN-only access für admin
- Enable 2FA für all users
- Regular security audits

Feature Documentation

1. VPN Infrastructure Management

Deploy VPN Node

```
# Via API
POST /api/vpn/deploy
{
  "provider": "aws",
  "region": "us-east-1",
  "quantity": 1
}

# Via Axiom
ax init --provider aws
ax deploy vpn-node-1 \
  --region us-east-1 \
  --size t3.micro
```

WireGuard Configuration

Automatisch generiert:

```
[Interface]
PrivateKey = &lt;client-private-key&gt;
Address = 10.10.0.2/32
DNS = 1.1.1.1

[Peer]
PublicKey = &lt;server-public-key&gt;
Endpoint = 54.123.45.67:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25
```

Mobile Setup

1. Navigate zu VPN Dashboard
2. Click "QR Code" button
3. Scan mit WireGuard App
4. Activate VPN

Traffic Monitoring

- Real-time in/out bytes
- Connected clients count
- Bandwidth graphs (last 24h)

2. Axiom Fleet Orchestration

Deploy Fleet

```
// Via API
const response = await fetch('/api/axiom/fleet/deploy', {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${token}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    fleetName: 'recon-fleet-alpha',
    provider: 'aws',
    instanceType: 'c5.xlarge',
    regions: ['us-east-1', 'us-west-2', 'eu-west-1'],
    quantity: 10,
    vpnNodeId: 'vpn-001',
    modules: ['nmap', 'nuclei', 'ffuf', 'subfinder']
  })
});
```

Via Axiom CLI

```
# Initialize Axiom
ax init --provider aws

# Deploy fleet
ax fleet recon-alpha \
  -i 10 \
  --region us-east-1,us-west-2,eu-west-1 \
  --size c5.xlarge \
  --spot

# Install modules
ax fleet recon-alpha -m nmap,nuclei,ffuf

# Execute distributed scan
ax scan targets.txt -m nuclei -o results.json
```

Auto-Scaling

```
// Scale up
await axiomService.scaleFleet('fleet-001', 'up', 5);

// Scale down
await axiomService.scaleFleet('fleet-001', 'down', 3);

// Auto-scale based on queue
if (queueSize &gt; 1000) {
  await axiomService.scaleFleet(fleetId, 'up', 10);
}
```

Cost Optimization mit Spot Instances

```
# 70% savings!
ax fleet my-fleet --spot --max-price 0.05
```

3. Ars0n OSINT Framework

Start Reconnaissance

```
const job = await ars0nService.startScan({
  userId: user.id,
  jobName: 'HackerOne Recon',
  target: 'hackerone.com',
  scanType: 'full',
  modules: [
    'fire-starter',    // Subdomain enumeration
    'fire-scanner',    // Vulnerability scanning
    'fire-cloud'       // Cloud asset discovery
  ],
  axiomFleetId: 'fleet-001',
  vpnNodeId: 'vpn-001'
});
```

Supported Modules

Subdomain Enumeration:

- Amass (comprehensive)
- Subfinder (fast)
- Sublist3r (passive)
- Assetfinder
- Certificate transparency logs
- ShuffleDNS (bruteforce)

Vulnerability Scanning:

- Nuclei (templates)
- Nmap (port scanning)
- Masscan (fast scanning)
- Testssl.sh (SSL/TLS)

Web Discovery:

- GAU (GetAllURLs)
- Waybackurls
- Hakrawler
- GoSpider

- Subdomainizer

Findings Management

```
-- Query findings by severity
SELECT * FROM osint_findings
WHERE job_id = 'osint-001'
  AND severity IN ('high', 'critical')
  AND verified = true
ORDER BY created_at DESC;
```

Export Results

```
# CSV Export
GET /api/osint/job/${jobId}/findings?format=csv

# JSON Export
GET /api/osint/job/${jobId}/findings?format=json

# Upload to Bug Bounty Platform
curl -X POST https://hackerone.com/reports \
  -H "Authorization: Bearer $TOKEN" \
  -d @findings.json
```

4. Distributed Collection

Create Collection Job

```
const job = await collectionService.createJob({
  userId: user.id,
  jobName: 'Market Research Data',
  jobType: 'web-scraping',
  targetList: userList,  // Array of targets
  command: 'node scraper.js',
  axiomFleetId: 'fleet-001',
  distributionStrategy: 'round-robin',
  rateLimit: {
    requestsPerSecond: 1,
    delayBetweenRequests: 1000
  }
});
```

Distribution Strategies

Round-Robin

```
Instance 1: Targets 1, 4, 7, 10, ...
Instance 2: Targets 2, 5, 8, 11, ...
Instance 3: Targets 3, 6, 9, 12, ...
```

Least-Loaded

```
Assign to instance with:
- Lowest CPU usage
- Fewest active jobs
- Best network performance
```

Regional

```
Group targets by region:
- US targets → US instances
- EU targets → EU instances
- APAC targets → APAC instances
```

IP Rotation

```
// Rotate through VPN nodes
for (const target of targets) {
  const vpnNode = getNextVPNNode();
  await routeThroughVPN(vpnNode);
  await scrapeTarget(target);
}
```

Results Aggregation

```
// Collect from all instances
const results = await Promise.all(
  instances.map(instance =&gt;
    getInstanceResults(instance.id)
  )
);

// Merge and deduplicate
const merged = deduplicateResults(results.flat());

// Export
await exportToCSV(merged, 'results.csv');
```

Cost Optimization

Monthly Cost Breakdown

Minimal Setup (Testing)

- VPN nodes (3): $16.50
- Database (self-hosted): $30
- CDN (Cloudflare): $0

- Storage: $1.15
  Total: ~$48/month

Active Research (Part-Time)

- VPN nodes (3): $16.50
- Axiom spot (4h/day): $60
- Database (RDS): $25
- Redis: $12
- CDN: $0
- Storage: $5
  Total: ~$118.50/month

Heavy Usage (Full-Time)

- VPN nodes (5): $30
- Axiom spot (8h/day): $120
- Database (RDS): $50
- Redis: $20
- CDN: $10
- Storage: $20
  Total: ~$250/month

Cost Optimization Strategies

1. Spot Instances (70% savings)

```
ax fleet my-fleet --spot --max-price 0.05
```

2. Regional Arbitrage

- US East (cheapest AWS): $0.17/hr
- US Central (cheapest GCP): $0.19/hr
- NYC1 (cheapest DO): $0.006/hr

3. Auto-Scaling

```
// Stop idle instances
if (instance.idleTime &gt; 1800) {
  await stopInstance(instance.id);
}

// Scale down at night
if (currentHour &gt; 22 || currentHour &lt; 6) {
  await scaleFleet(fleetId, 'down', 5);
}
```

4. Reserved Instances

- 1-year: 30% discount
- 3-year: 60% discount

5. Free Tiers

- AWS: 750 hrs/month t2.micro
- GCP: 1× e2-micro always free
- Cloudflare: Unlimited CDN
- DO: $200 credit

6. Monitoring & Alerts

```
resource "aws_budgets_budget" "atlas" {
  limit_amount = "100"
  limit_unit   = "USD"
  time_unit    = "MONTHLY"
}
```

ROI Analysis

Compared to Individual Tools:

- Steg.ai (watermarking): $500/month
- ProxyRack VPN: $100/month
- Bug bounty tools: $200/month
- Cloud management: $150/month

Atlas Framework: $118/month
Savings: $832/month

Break-Even:

- 1 medium bug ($500) = 4 months
- 1 high bug ($2000) = 16 months
- 1 critical bug ($10k+) = Years!

Security & Compliance

Authentication & Authorization

JWT-Based Authentication

```
// Login
const { token } = await fetch('/api/auth/login', {
  method: 'POST',
  body: JSON.stringify({ email, password })
});

// Store token
localStorage.setItem('token', token);

// Use in requests
fetch('/api/vpn/nodes', {
  headers: { 'Authorization': `Bearer ${token}` }
});
```

Role-Based Access Control

- **Admin**: Full access
- **Researcher**: Deploy resources, view costs
- **Viewer**: Read-only access

API Key Management

```
# Generate API key
POST /api/auth/api-key
Response: { "key": "atlas_sk_..." }

# Use API key
curl -H "X-API-Key: atlas_sk_..." \
  https://api.yourdomain.com/vpn/nodes
```

Data Protection

Credential Encryption

```
// Encrypt before storing
const encrypted = encrypt(credentials, ENCRYPTION_KEY);
await db.query(
  'INSERT INTO cloud_providers (credentials_encrypted) VALUES ($1)',
  [encrypted]
);

// Decrypt when needed
const decrypted = decrypt(stored, ENCRYPTION_KEY);
```

Network Security

- TLS 1.3 für all communications
- Certificate pinning in production
- VPN-only access option
- Firewall automation via Security Groups

Audit Logging

Comprehensive Audit Trail

```
-- Log all actions
INSERT INTO audit_logs (
  user_id, action, resource_type,
  resource_id, ip_address, details
) VALUES (
  $1, 'VPN_DEPLOYED', 'vpn_node',
  $2, $3, $4
);

-- Query audit logs
SELECT * FROM audit_logs
WHERE user_id = 'user-001'
  AND action LIKE '%DEPLOY%'
  AND created_at &gt; NOW() - INTERVAL '7 days'
ORDER BY created_at DESC;
```

Audit Log Retention

- Default: 90 days
- Configurable per compliance requirements
- Auto-archiving to S3 Glacier

Compliance Features

GDPR Compliance

- User data export capability
- Right to be forgotten (delete user)
- Data processing agreements
- Privacy policy enforcement

SOC 2 Readiness

- Complete audit trail
- Encryption at rest and in transit
- Access controls und MFA
- Regular security audits

Use Cases

1. Bug Bounty Research

Scenario: Reconnaissance auf multiple targets

```
# 1. Deploy infrastructure
# VPN for anonymity
./deploy_vpn.sh --regions us-east-1,eu-west-1,ap-southeast-1

# Axiom fleet for distributed scanning
ax fleet bug-bounty -i 20 --spot --regions us-east-1,us-west-2

# 2. Start reconnaissance
# Subdomain enumeration
ax scan targets.txt -m subfinder,amass -o subdomains.txt

# Vulnerability scanning
ax scan subdomains.txt -m nuclei -o vulns.json

# 3. Monitor results in dashboard
# Navigate to Ars0n page
# Filter by severity: high, critical
# Verify findings
# Submit to bug bounty platform
```

ROI: 1 critical bug ($10k) pays for 3+ years of Atlas

2. Security Research

Scenario: Cloud asset discovery für Fortune 500 company

```
// Deploy cloud reconnaissance
const job = await ars0nService.startScan({
  target: 'target-company.com',
  scanType: 'cloud',
  modules: ['fire-cloud', 'cloud_enum', 's3scanner'],
  axiomFleetId: 'research-fleet'
});

// Monitor findings
const findings = await ars0nService.getFindings(job.id);
const cloudAssets = findings.filter(f =&gt; f.type === 'cloud-asset');

// Report
console.log(`Found ${cloudAssets.length} cloud assets`);
```

3. Market Research

Scenario: Competitive intelligence data collection

```
// Create collection job
const job = await collectionService.createJob({
  jobName: 'Competitor Pricing Analysis',
  targetList: competitorList,
  command: 'node price-scraper.js',
  axiomFleetId: 'collection-fleet',
  distributionStrategy: 'regional',
  rateLimit: { requestsPerSecond: 0.5 }
});

// Export results
const results = await collectionService.getResults(job.id);
await exportToExcel(results, 'competitor-analysis.xlsx');
```

4. DevOps & Infrastructure Management

Scenario: Multi-cloud cost optimization

```
// Deploy monitoring across clouds
const awsFleet = await axiomService.deployFleet({
  provider: 'aws',
  regions: ['us-east-1', 'eu-west-1'],
  quantity: 5
});

const gcpFleet = await axiomService.deployFleet({
  provider: 'gcp',
  regions: ['us-central1', 'europe-west1'],
  quantity: 5
});

// Analyze costs
const costs = await analyticsService.getCosts({
  groupBy: 'provider'
});

// Get recommendations
const recs = await analyticsService.getRecommendations();
// Output: "Switch 3 instances to spot: Save $120/month"
```

API Reference

Authentication

POST /api/auth/register

```
Request:
{
  "email": "user@example.com",
  "password": "SecurePass123!",
  "name": "John Doe"
}

Response:
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": { "id": "user-001", "email": "user@example.com" }
}
```

VPN Management

GET /api/vpn/nodes
POST /api/vpn/deploy
GET /api/vpn/:nodeId/config
GET /api/vpn/:nodeId/qr
DELETE /api/vpn/:nodeId

Axiom Orchestration

GET /api/axiom/fleets
POST /api/axiom/fleet/deploy
GET /api/axiom/fleet/:fleetId
POST /api/axiom/fleet/:fleetId/scale
DELETE /api/axiom/fleet/:fleetId
POST /api/axiom/fleet/:fleetId/execute

OSINT Framework

GET /api/osint/jobs
POST /api/osint/scan
GET /api/osint/job/:jobId
GET /api/osint/job/:jobId/findings
POST /api/osint/job/:jobId/stop

Collection Jobs

GET /api/collection/jobs
POST /api/collection/job
GET /api/collection/job/:jobId/results
GET /api/collection/job/:jobId/export/:format

Analytics

GET /api/analytics/overview
GET /api/analytics/costs
GET /api/analytics/usage
GET /api/analytics/recommendations

Troubleshooting

Common Issues

Database Connection Errors

```
# Check PostgreSQL
docker-compose ps postgres

# View logs
docker-compose logs postgres

# Restart
docker-compose restart postgres
```

Axiom SSH Connection Fails

```
# Test SSH key
ssh -i ./keys/axiom_rsa axiom@$AXIOM_CONTROLLER_IP

# Check key permissions
chmod 600 ./keys/axiom_rsa

# Verify controller IP
echo $AXIOM_CONTROLLER_IP
```

VPN Deployment Fails

```
# Check cloud credentials
aws sts get-caller-identity
gcloud auth list

# Verify region availability
aws ec2 describe-regions
```

High Costs Alert

```
# Check running instances
aws ec2 describe-instances --query 'Reservations[].Instances[?State.Name==`running`]'

# Terminate idle resources
./scripts/cleanup-idle.sh
```

```
# Enable auto-scaling down
```

Legal & Ethical Use

Authorized Use Cases

✅ **Bug Bounty Programs** with explicit authorization
✅ **Penetration Testing** with written client permission
✅ **Security Research** on owned infrastructure
✅ **DevOps & Cloud Management** for authorized resources
✅ **Educational Purposes** in controlled environments

Prohibited Activities

❌ **Unauthorized Access** to systems or networks
❌ **ToS Violations** of any platform
❌ **Malicious Activities** including DDoS
❌ **Illegal Data Collection** without consent
❌ **Privacy Violations** or surveillance

User Responsibilities

1. **Obtain Authorization** for all security testing
2. **Comply with Laws** and regulations
3. **Respect ToS** of cloud providers und platforms
4. **Secure Credentials** and access
5. **Ethical Use** aligned with security research norms

Risk Acknowledgment

Users acknowledge:

- Cloud resources incur real costs
- Misconfiguration may lead to security issues
- Improper use may violate laws
- Users bear full legal responsibility

Conclusion

Atlas Framework liefert eine comprehensive, production-ready Lösung für security researchers, bug bounty hunters, und DevOps teams. Die Plattform vereint VPN management, multi-cloud orchestration, OSINT automation, und distributed collection in einer einzigen, intuitiven Oberfläche.

Key Benefits:

- **Unified Platform**: Alles in einem tool
- **Cost-Effective**: $118/month vs $950+ für separate tools
- **Scalable**: From testing bis production-scale
- **Secure**: Enterprise-grade security features
- **Flexible**: Adaptable für various use cases

Next Steps:

1. Deploy Atlas Framework mit Docker Compose
2. Configure cloud providers in Settings
3. Deploy VPN infrastructure
4. Create Axiom fleet
5. Start OSINT reconnaissance
6. Monitor costs und optimize

Für questions, contributions, oder support, siehe project repository und community channels.

Document Version: 1.0.0
Last Updated: October 18, 2025
License: MIT
Project: Atlas Framework