Security Best Practices & Audit Guide

Smart Contract Security

Non-Custodial Architecture

Novisca employs **Program Derived Accounts (PDAs)** to ensure non-custodial fund management:

```
// Only user's escrow can be modified by that user
seeds = [b"escrow", user.key().as_ref()]
```

Security Guarantees:

- ✅ No private keys on servers
- ✅ Only user can withdraw (enforced by PDA)
- ✅ All transactions signed by user wallet
- ✅ Transparent on-chain verification

Oracle Security

Dual Oracle Strategy:

- Primary: Pyth Oracle (real-time, decentralized)
- Fallback: Switchboard Oracle
- Median price with 5% deviation threshold
- Circuit breaker on extreme movements

```
const pythPrice = await pyth.getLatestPrice(symbol);
const sbPrice = await switchboard.getPrice(symbol);

// Validate 5% deviation
const deviation = Math.abs(pythPrice - sbPrice) / sbPrice;
require!(deviation < 0.05, "Oracle deviation too high");

// Use median
const medianPrice = (pythPrice + sbPrice) / 2;
```

CPI Safety (Cross-Program Invocation)

All external program calls validated:

```
// Verify Kamino program address
require_eq!(lending_program.key(), KAMINO_PROGRAM_ID);

// Validate account ownership
require_eq!(lending_pool.owner, KAMINO_PROGRAM_ID);
```

Account Validation Checklist

- [ ] Correct program ownership
- [ ] Correct account structure
- [ ] Correct PDA derivation
- [ ] Signer verification
- [ ] Writable account validation

API Security

Authentication

```
// Verify wallet signature
const verified = nacl.sign.detached.verify(
  message,
  signature,
  wallet
);
require!(verified, "Invalid signature");
```

Rate Limiting

- Public endpoints: 100 req/min
- Authenticated: 1000 req/min
- Per-IP tracking
- Exponential backoff for failures

Input Validation

```
// Validate all inputs
const walletSchema = z.string().refine((val) => {
  try {
    new PublicKey(val);
```

```
    return true;
  } catch {
    return false;
  }
});

const amountSchema = z.number().positive().max(1e18);
```

SQL Injection Prevention

```
// ✅ GOOD: Parameterized queries
const user = await db.user.findUnique({
  where: { wallet: userWalletInput }
});

// ❌ BAD: String concatenation
const query = `SELECT * FROM users WHERE wallet = '${userInput}'`;
```

Network Security

DDoS Protection

- CloudFlare or AWS Shield
- Rate limiting per IP
- Geographic blocking if needed
- Automatic blacklist of suspicious IPs

HTTPS/TLS

- TLS 1.3 minimum
- Strong cipher suites
- Certificate pinning for mobile apps

CORS Configuration

```
cors({
  origin: ['https://novisca.io', 'https://app.novisca.io'],
  credentials: true,
  methods: ['GET', 'POST', 'PUT', 'DELETE'],
```

```
  allowedHeaders: ['Content-Type', 'Authorization']
})
```

Data Security

Sensitive Data Handling

```
// ❌ Never log sensitive data
console.log(privateKey); // SECURITY RISK

// ✅ Log only necessary info
console.log(`User ${wallet} executed trade`);
```

Database Encryption

```
-- Enable SSL for database connections
postgresql:// user:pass@host/db?sslmode=require

-- Encrypt sensitive columns
ALTER TABLE users ADD COLUMN encrypted_data BYTEA;
```

Backup Security

- Encrypted backups
- Multiple geographic locations
- Regular restore testing
- Access control and audit logs

Key Management

Private Key Storage

```
// ✅ Use environment variables
const privateKey = process.env.KEEPER_PRIVATE_KEY;

// ✅ Or use AWS Secrets Manager
const secret = await secretsManager.getSecretValue({
```

```
  SecretId: 'keeper-private-key'
});

// ❌ NEVER hardcode
const badKey = "3LkR9m2bV8x9nLm0pO1qR2sTu3vW4xY5z";
```

Keeper Bot Keys

- Separate from admin keys
- Limited permissions
- Rotating keys monthly
- Monitored for unusual activity

Audit Trail

Transaction Logging

```
// Log all state-changing operations
emit!(TradeEvent {
  user: wallet,
  symbol,
  side,
  size,
  entryPrice,
  timestamp: Clock::get()?.unix_timestamp
});
```

Admin Actions

- All admin actions logged with timestamp
- Multisig required for critical operations
- Regular audit of logs

Regular Security Audits

Pre-Launch (Mainnet)

- [ ] Full smart contract audit (third-party)
- [ ] Penetration testing
- [ ] Load testing (DDoS resilience)
- [ ] Oracle security review

- [ ] Access control review

Ongoing

- [ ] Monthly security review
- [ ] Quarterly penetration testing
- [ ] Annual full audit
- [ ] Bug bounty program active

Incident Response

Security Incident Procedures

1. **Detection**
    - Monitor alerts
    - Review logs
    - Validate issue
2. **Response**
    - Contain: Stop bleeding (pause operations if needed)
    - Eradicate: Fix the root cause
    - Recover: Restore normal operations
3. **Communication**
    - Notify affected users
    - Post-mortem analysis
    - Update security measures

Contact

- Security Issues: security@novisca.io
- Report format: Include steps to reproduce, severity level
- Response time: <24 hours for critical issues

Compliance

Regulatory Considerations

- User KYC/AML (if applicable)
- Jurisdiction-specific requirements
- Tax reporting for users
- Privacy policy compliance

Terms of Service

- Clear liability limitations
- Risk disclosures
- User agreement required
- Regular updates

User Security Education

- Wallet security best practices
- Never share private keys
- Use hardware wallets for large amounts
- Verify domains before connecting wallet
- Beware of phishing attempts

Monitoring & Alerts

Key Metrics to Monitor

```
// API Performance
- Response time < 200ms (p95)
- Error rate < 0.1%
- Uptime > 99.9%

// Blockchain
- Transaction confirmation < 30 seconds
- Oracle price deviations > 5%
- Liquidation frequency

// Security
- Failed authentication attempts
- SQL query errors
- Unauthorized access attempts
- Large withdrawals
```

Additional Resources

- OWASP Top 10: https://owasp.org/Top10/
- Solana Security: https://docs.solana.com/developing/programming-model/security
- Anchor Security: https://book.anchor-lang.com/
- Web3 Security: https://consensys.io/diligence/