STUNIR Security Policy

This document outlines the security considerations, best practices, and reporting procedures for the STUNIR project.

Overview

STUNIR is a deterministic build verification system that handles cryptographic signatures, file hashing, and build attestation. Security is paramount to ensure the integrity and trustworthiness of the attestation system.

Security Architecture

Cryptographic Components

Ed25519 Digital Signatures

- **Location**: `tools/pack_attestation/Attestation.hs`
- **Implementation**: Uses the `cryptonite` library for Ed25519 (RFC 8032)
- **Key Features**:
- 128-bit security level
- Constant-time signature verification (timing attack resistant)
- Cryptographically secure key generation
- Proper key validation before use

Merkle Tree Directory Hashing

- **Location**: `tools/native/rust/stunir-native/src/crypto.rs`
- **Implementation**: SHA-256 based Merkle tree
- **Key Features**:
- Deterministic ordering via sorted paths
- Symlink rejection for security
- Path traversal protection
- Depth limits to prevent DoS

Input Validation

Path Validation (`tools/security/validation.py`)

All file paths are validated to prevent:
- **Directory traversal attacks** (`../` sequences)
- **Null byte injection** (`\x00` in paths)
- **Absolute path escape** (when relative paths expected)
- **Symlink attacks** (optionally rejected)

```python
from tools.security.validation import validate_path

# Safe usage
safe_path = validate_path(user_input, base_dir="/repo", allow_symlinks=False)
```

JSON/CBOR Validation

- Maximum nesting depth enforcement
- String length limits
- Schema validation support
- UTF-8 encoding validation

Subprocess Security (`tools/security/subprocess_utils.py`)

CRITICAL: Never use `shell=True` with subprocess calls.

```python
# WRONG - Vulnerable to injection
subprocess.run(f"git commit -m '{user_message}'", shell=True)

# CORRECT - Safe argument list
from tools.security.subprocess_utils import run_command
run_command(["git", "commit", "-m", user_message])
```

Key protections:
- Shell metacharacter detection and rejection
- Command whitelist enforcement (optional)
- Timeout protection against hung processes
- Proper error handling with context

Security Fixes Applied

Phase 1 Critical Security Issues (Resolved)

Issue	Location	Fix Applied
Placeholder Ed25519 signature	`Attestation.hs`	Real Ed25519 implementation using cryptonite
Directory hash stub	`crypto.rs`	Full Merkle tree implementation
Bare `except:` clauses	Multiple Python files	Specific exception types
Missing input validation	Various	Comprehensive validation module

Detailed Fix Descriptions

1. Ed25519 Signature Implementation

Before: `signature = "ed25519_placeholder"`

After: Full implementation with:
- Key generation via `Crypto.Random.getRandomBytes`
- Key loading/saving in hex format
- Canonical CBOR encoding for deterministic signing
- Verification with proper error handling

2. Merkle Tree Directory Hashing

Before: `Ok("DIR_HASH_TODO".to_string())`

After: Complete implementation with:
- Recursive directory traversal
- Sorted path ordering (Unicode codepoint)
- SHA-256 file hashing
- Combined Merkle root computation
- Depth and size limits

3. Exception Handling

Before:

```
try:
    int(value)
except:  # Catches everything including KeyboardInterrupt!
    pass
```

After:

```
try:
    int(value)
except (ValueError, TypeError):
    # Specific handling for expected failures
    pass
```

Security Testing

Running Security Tests

```
# Run all security tests
cd /home/ubuntu/stunir_repo
python -m pytest tests/security/ -v

# Run specific test category
python -m pytest tests/security/test_validation.py -v
python -m pytest tests/security/test_subprocess.py -v
```

Test Coverage

The security test suite covers:
- Path traversal attack vectors
- Command injection attempts
- JSON/CBOR malicious inputs
- File size DoS prevention
- Signature verification edge cases

Reporting Security Issues

Responsible Disclosure

If you discover a security vulnerability in STUNIR:

1. **DO NOT** open a public GitHub issue
2. Email security concerns to the maintainers directly
3. Include:
   - Description of the vulnerability
   - Steps to reproduce
   - Potential impact assessment
   - Any suggested fixes

Severity Classification

Severity	Description	Examples
Critical	Remote code execution, signature bypass	Command injection, signature forgery
High	Information disclosure, DoS	Path traversal to sensitive files
Medium	Limited impact vulnerabilities	Resource exhaustion with large files
Low	Minor issues	Verbose error messages

Security Best Practices

For Contributors

1. **Never trust user input**
   - Always validate paths before file operations
   - Sanitize strings before logging
   - Use parameterized queries/commands

2. **Use the security utilities**
   `python`
   `   from tools.security import validate_path, run_command`

3. **Avoid bare exceptions**
   ```python
   # BAD
   except:
   pass

# GOOD
except (ValueError, TypeError) as e:
   ```

```
logging.error(f"Validation failed: {e}")
```

1. **Never use shell=True**

```python
# BAD
subprocess.run(cmd, shell=True)

# GOOD
run_command(cmd_list)
```

1. **Validate cryptographic inputs**
   - Check key lengths before use
   - Verify signature formats
   - Validate hash hex strings

For Operators

1. **Key Management**
   - Store Ed25519 private keys securely
   - Rotate keys periodically
   - Use hardware security modules for production

2. **File Permissions**
   - Restrict write access to attestation outputs
   - Protect receipt directories
   - Audit file access

3. **Monitoring**
   - Log signature verification failures
   - Alert on unusual file sizes
   - Monitor for path traversal patterns

Dependencies

Security-Critical Dependencies

Component	Library	Purpose
Ed25519 Signing	cryptonite (Haskell)	Digital signatures
SHA-256 Hashing	sha2 (Rust)	File/directory hashing
CBOR Encoding	cborg (Haskell)	Canonical serialization

Dependency Updates

- Review dependency updates for security fixes
- Pin versions in production
- Use tools like `cargo audit` and `cabal outdated`

Audit History

Date	Scope	Findings	Resolution
2026-01-28	Phase 1 Security Review	6 critical issues	All resolved

Changelog

2026-01-28 - Phase 1 Security Fixes

- Implemented real Ed25519 signatures in Attestation.hs
- Added Merkle tree directory hashing in crypto.rs
- Created security utilities module (`tools/security/`)
- Fixed bare exception handling in Python files
- Added comprehensive security test suite
- Created this SECURITY.md documentation

Last updated: 2026-01-28
STUNIR Security Team