Continuous Compliance

Martin Kellogg\textsuperscript{a}, Martin Schäf\textsuperscript{b}, Serdar Tasiran\textsuperscript{b}, Michael D. Ernst\textsuperscript{a,b}

\textsuperscript{a}University of Washington  \textsuperscript{b}Amazon Web Services
What is compliance?

You

Customer
What is compliance?

You

How do I know it’s secure?

Customer
What is compliance?

You

Auditor

Customer

You

Customer
What is compliance?

You

Auditor

Customer

You

Customer
# Audit workflow

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>code review, keep compliance in mind</td>
</tr>
<tr>
<td>Preparation</td>
<td></td>
</tr>
<tr>
<td>Review</td>
<td></td>
</tr>
</tbody>
</table>
## Audit workflow

<table>
<thead>
<tr>
<th>Stage</th>
<th>Traditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>code review, keep compliance in mind</td>
</tr>
<tr>
<td>Preparation</td>
<td>gather evidence from each engineering team</td>
</tr>
<tr>
<td>Review</td>
<td></td>
</tr>
</tbody>
</table>
# Audit workflow

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>code review, keep compliance in mind</td>
</tr>
<tr>
<td>Preparation</td>
<td>gather evidence from each engineering team</td>
</tr>
<tr>
<td>Review</td>
<td>randomly sample, manually check evidence</td>
</tr>
</tbody>
</table>
## Audit workflow

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Problems:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>code review, keep compliance in mind</td>
<td></td>
</tr>
<tr>
<td>Preparation</td>
<td>gather evidence from each engineering team</td>
<td></td>
</tr>
<tr>
<td>Review</td>
<td>randomly sample, manually check evidence</td>
<td></td>
</tr>
</tbody>
</table>
## Audit workflow

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Problems:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>code review, <em>keep compliance in mind</em></td>
<td>Cost</td>
</tr>
<tr>
<td>Preparation</td>
<td>gather evidence from <em>each</em> engineering team</td>
<td></td>
</tr>
<tr>
<td>Review</td>
<td>randomly sample, <em>manually check evidence</em></td>
<td></td>
</tr>
</tbody>
</table>
## Audit workflow

<table>
<thead>
<tr>
<th>Development</th>
<th>Traditional</th>
<th>Problems:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>code review, keep compliance in mind</td>
<td>Cost</td>
</tr>
<tr>
<td>Preparation</td>
<td>gather evidence from each engineering team</td>
<td>Judgment</td>
</tr>
<tr>
<td>Review</td>
<td>randomly sample, manually check evidence</td>
<td></td>
</tr>
</tbody>
</table>
## Audit workflow

<table>
<thead>
<tr>
<th>Development</th>
<th>Traditional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>code review, keep compliance in mind</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preparation</th>
<th>Traditional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gather evidence from each engineering team</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Review</th>
<th>Traditional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>randomly sample, manually check evidence</td>
</tr>
</tbody>
</table>

### Problems:
- Cost
- Judgment
- Sampling
# Audit workflow

<table>
<thead>
<tr>
<th>Development</th>
<th>Traditional</th>
<th>Problems:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>code review, keep compliance in mind</td>
<td>⚫ Cost</td>
</tr>
<tr>
<td>Preparation</td>
<td>gather evidence from each engineering team</td>
<td>⚫ Judgment</td>
</tr>
<tr>
<td>Review</td>
<td>randomly sample, manually check evidence</td>
<td>⚫ Sampling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>⚫ Regressions</td>
</tr>
</tbody>
</table>
Continuous Compliance

- Build **verification tools** for compliance controls
- On each commit, run verifier in **continuous integration**
- Report failures directly to developers
## Audit workflow

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development</strong></td>
<td>code review, keep compliance in mind</td>
<td>none</td>
</tr>
<tr>
<td><strong>Preparation</strong></td>
<td>gather evidence from each engineering team</td>
<td>none</td>
</tr>
<tr>
<td><strong>Review</strong></td>
<td>randomly sample, manually check evidence</td>
<td>none</td>
</tr>
</tbody>
</table>
# Audit workflow

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development</strong></td>
<td>code review, keep compliance in mind</td>
<td>write specifications, verifier runs in CI</td>
</tr>
<tr>
<td><strong>Preparation</strong></td>
<td>gather evidence from each engineering team</td>
<td></td>
</tr>
<tr>
<td><strong>Review</strong></td>
<td>randomly sample, manually check evidence</td>
<td></td>
</tr>
</tbody>
</table>
## Audit workflow

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development</strong></td>
<td>code review, keep compliance in mind</td>
<td>write specifications, verifier runs in CI</td>
</tr>
<tr>
<td><strong>Preparation</strong></td>
<td>gather evidence from each engineering team</td>
<td>none</td>
</tr>
<tr>
<td><strong>Review</strong></td>
<td>randomly sample, manually check evidence</td>
<td></td>
</tr>
</tbody>
</table>
## Audit workflow

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development</strong></td>
<td>code review, keep compliance in mind</td>
<td>write specifications, verifier runs in CI</td>
</tr>
<tr>
<td><strong>Preparation</strong></td>
<td>gather evidence from each engineering team</td>
<td>none</td>
</tr>
<tr>
<td><strong>Review</strong></td>
<td>randomly sample, manually check evidence</td>
<td>auditor checks output of verifier</td>
</tr>
</tbody>
</table>
# Audit workflow

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development</strong></td>
<td>code review, keep compliance in mind</td>
<td>write specifications, verifier runs in CI</td>
</tr>
<tr>
<td><strong>Preparation</strong></td>
<td>gather evidence from each engineering team</td>
<td>none</td>
</tr>
<tr>
<td><strong>Review</strong></td>
<td>randomly sample, manually check evidence</td>
<td>auditor checks output of verifier</td>
</tr>
</tbody>
</table>

- **Cost**
- **Judgment**
- **Sampling**
- **Regressions**
Audit workflow

<table>
<thead>
<tr>
<th>Development</th>
<th>Traditional</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>code review, keep compliance in mind</td>
<td>write specifications, verifier runs in CI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preparation</th>
<th>Traditional</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gather evidence from each engineering team</td>
<td>none</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Review</th>
<th>Traditional</th>
<th>Continuous</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>randomly sample, manually check evidence</td>
<td>auditor checks output of verifier</td>
</tr>
</tbody>
</table>
Contributions

- **Idea**: verification is a good fit for compliance
Contributions

- **Idea**: verification is a good fit for compliance
- **Engineering**: we built verifiers for five compliance controls
Contributions

- **Idea**: verification is a good fit for compliance
- **Engineering**: we built verifiers for five compliance controls
- **Experimental**: open-source experiments and comparisons
Contributions

- **Idea**: verification is a good fit for compliance
- **Engineering**: we built verifiers for five compliance controls
- **Experimental**: open-source experiments and comparisons
- **Experiential**: verifiers in the compliance process at AWS
Compliance controls

Controls:

- HTTP vs HTTPS
- Cryptographic key length
- Cryptographic algorithm selection
- Cloud data store initialization
- Hard-coded credentials
Compliance controls

**Controls:**
- HTTP vs HTTPS
- Cryptographic key length
- Cryptographic algorithm selection
- Cloud data store initialization
- Hard-coded credentials

**Techniques:**
Compliance controls

**Controls:**
- HTTP vs HTTPS
- Cryptographic key length
- Cryptographic algorithm selection
- Cloud data store initialization
- Hard-coded credentials

**Techniques:**
- Constant propagation
Compliance controls

**Controls:**
- HTTP vs HTTPS
- Cryptographic key length
- Cryptographic algorithm selection
- Cloud data store initialization
- Hard-coded credentials

**Techniques:**
- Constant propagation
- + enum analysis
Compliance controls

**Controls:**
- HTTP vs HTTPS
- Cryptographic key length
- Cryptographic algorithm selection
- Cloud data store initialization
- Hard-coded credentials

**Techniques:**
- Constant propagation
- + enum analysis
- + regex matching
Compliance controls

**Controls:**
- HTTP vs HTTPS
- Cryptographic key length
- Cryptographic algorithm selection
- Cloud data store initialization
- Hard-coded credentials

**Techniques:**
- Constant propagation
- + enum analysis
- + regex matching
- + accumulation analysis
Compliance controls

**Controls:**
- HTTP vs HTTPS
- Cryptographic key length
- Cryptographic algorithm selection
- Cloud data store initialization
- Hard-coded credentials

**Techniques:**
- Constant propagation
- + enum analysis
- + regex matching
- + accumulation analysis
- + dataflow
Analysis strategy
Analysis strategy: type systems

- Familiar to developers
- Predictable
- Scalable
- Sound
Evaluation

1. Run all verifiers on 492 open-source projects
2. Compare verifiers to existing tools
3. Case study of a verifier in a real, industrial compliance workflow
4. Case study of two verifiers as part of industrial security scans
Evaluation

1. Run all verifiers on 492 open-source projects

2. Compare verifiers to existing tools

3. Case study of a verifier in a real, industrial compliance workflow

4. Case study of two verifiers as part of industrial security scans
Open-source projects

- 492 projects from GitHub, 5.7 million LoC
  - Use type inference and build scanning to automate process
Open-source projects

- 492 projects from GitHub, 5.7 million LoC
  - Use type inference and build scanning to automate process

- Triage into 4 categories:
  - **verified**, no warnings
  - **false positives**: warnings, no real violations
  - **true positives**: all warnings are real violations
  - **true and false positives**: some warnings are real violations
Open-source projects

Real violations:

- verified (157)
- true positives (176)
- true and false positives (77)

False warnings:

- false positives (82)
Open-source projects

Real violations:

- verified (157)
- true positives (176)
- true and false positives (77)

False warnings:

- false positives (82)

Takeaways:

- ~1/2 open-source projects have compliance violations
Open-source projects

False warnings:
- False positives (82)
- True and false positives (77)

Real violations:
- Verified (157)
- True positives (176)

Takeaways:
- ~1/2 open-source projects have compliance violations
- ~2/3 projects cause no false positives from our tools
Evaluation

1. Run all verifiers on 492 open-source projects

2. Compare verifiers to existing tools

3. Case study of a verifier in a real, industrial compliance workflow

4. Case study of two verifiers as part of industrial security scans
Comparison with other tools

- Used a CryptoAPIBench, a previously-published benchmark
- Only compared on categories covered by our tools (11/16)
- Four other tools:
  - SpotBugs
  - Coverity
  - CogniCrypt_{SAST} (CrySL)
  - CryptoGuard
## Comparison with other tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>SpotBugs</th>
<th>Coverity</th>
<th>CrySL</th>
<th>CryptoGuard</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Comparison with other tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>SpotBugs</th>
<th>Coverity</th>
<th>CrySL</th>
<th>CryptoGuard</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.69</td>
<td><strong>1.0</strong></td>
<td>0.79</td>
<td><strong>1.0</strong></td>
<td>0.97</td>
</tr>
<tr>
<td>Recall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Bold values indicate superior performance.
## Comparison with other tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>SpotBugs</th>
<th>Coverity</th>
<th>CrySL</th>
<th>CryptoGuard</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.69</td>
<td>1.0</td>
<td>0.79</td>
<td>1.0</td>
<td>0.97</td>
</tr>
<tr>
<td>Recall</td>
<td>0.32</td>
<td>0.38</td>
<td>0.61</td>
<td>0.88</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Comparison with other tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>SpotBugs</th>
<th>Coverity</th>
<th>CrySL</th>
<th>CryptoGuard</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.69</td>
<td>1.0</td>
<td>0.79</td>
<td>1.0</td>
<td>0.97</td>
</tr>
<tr>
<td>Recall</td>
<td>0.32</td>
<td>0.38</td>
<td>0.61</td>
<td>0.88</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Only ours are suitable for compliance: auditors won’t accept a tool that has **false negatives**
Evaluation

1. Run all verifiers on 492 open-source projects
2. Compare verifiers to existing tools
3. Case study of a verifier in a real, industrial compliance workflow
4. Case study of two verifiers as part of industrial security scans
AWS case study 1: auditor acceptance

- key-length verifier
- verified in CI for 7 core AWS services
- replaced existing manual compliance workflow
- auditors accepted output of tool: all services compliant
AWS case study 1: auditor acceptance

- key-length verifier
- verified in CI for 7 core AWS services
- **replaced** existing manual compliance workflow
- auditors accepted output of tool: **all services compliant**

“It eliminates [the need for] a lot of trust”

- external auditor
Why does it eliminate the need for trust?

```java
public SecretKey getKMSKey(int keyLength) {
    GenerateDataKeyRequest r = new GenerateDataKeyRequest();
    if (keyLength == 128) {
        r.withKeySpec(DataKeySpec.AES_128);
    } else {
        r.withKeySpec(DataKeySpec.AES_256);
    }
    ...
```
Why does it eliminate the need for trust?

```java
public SecretKey getKMSKey(int keyLength) {
    GenerateDataKeyRequest r = new GenerateDataKeyRequest();
    if (keyLength == 128) {
        r.withKeySpec(DataKeySpec.AES_128);
    } else {
        r.withKeySpec(DataKeySpec.AES_256);
    }
    ...
}
```
public SecretKey getKMSKey(int keyLength) {
    GenerateDataKeyRequest r = new GenerateDataKeyRequest();
    if (keyLength == 128) {
        r.withKeySpec(DataKeySpec.AES_128);
    } else {
        r.withKeySpec(DataKeySpec.AES_256);
    }
    ...
AWS case study 1: auditor acceptance

- key-length verifier
- verified in CI for 7 core AWS services
- replaced existing manual compliance workflow
- auditors accepted output of tool: all services compliant

“It eliminates [the need for] a lot of trust”

- external auditor
AWS case study 1: auditor acceptance

- key-length verifier
- verified in CI for 7 core AWS services
- replaced existing manual compliance workflow
- auditors accepted output of tool: all services compliant

“It eliminates [the need for] a lot of trust”

- external auditor

“This has saved my team 2 hours every 6 months and we also don’t have to worry about failing an audit control.”

- developer
AWS case study 1: auditor acceptance

- key-length verifier
- verified in CI for 7 core AWS services
- replaced existing manual compliance workflow
- auditors accepted output of tool: all services compliant

“*It eliminates [the need for] a lot of trust*”

- external auditor

“*This has saved my team 2 hours every 6 months and we also don’t have to worry about failing an audit control.*”

- developer

per team, per audit, per control
Evaluation

1. Run all verifiers on 492 open-source projects

2. Compare verifiers to existing tools

3. Case study of a verifier in a real, industrial compliance workflow

4. Case study of two verifiers as part of industrial security scans
AWS case study 2: security scanning

- key-length and crypto-algorithm verifiers
- scan all security-relevant (not just compliance relevant) code
Industrial projects

Real violations:

verified
(37,315)

true positives (173)

False warnings: false positives (1) true and false positives (0)
Industrial projects

Real violations:
99.94% required no annotations

Verified: (37,315)

True positives: (173)

False warnings:
false positives: (1)
true and false positives: (0)
Industrial projects

Real violations:

- All validated by security engineers;
  none compliance relevant

- true positives (173)

False warnings:

- false positives (1)
- true and false positives (0)
Lessons learned
Lessons learned

1. Verification is a good fit for **compliance**
   a. auditors require soundness (no false negatives)
   b. most controls are local and simple (human-checkable)
Lessons learned

1. Verification is a good fit for compliance
   a. auditors require soundness (no false negatives)
   b. most controls are local and simple (human-checkable)

2. Verification is useful for stakeholders other than programmers
   a. auditors, managers, security reviewers, etc.
   b. research impact from focusing on other stakeholders
Lessons learned

1. Verification is a good fit for compliance
   a. auditors require soundness (no false negatives)
   b. most controls are local and simple (human-checkable)

2. Verification is useful for stakeholders other than programmers
   a. auditors, managers, security reviewers, etc.
   b. research impact from focusing on other stakeholders

3. Verification can save time for developers
   a. don’t add a new task, replace an existing task
   b. verification is easier than tasks developers already do
Contributions

- **Idea**: verification is a good fit for compliance
- **Engineering**: we built verifiers for five compliance controls
- **Experimental**: open-source experiments and comparisons
- **Experiential**: verifiers in the compliance process at AWS

Tools and data are publicly available: see paper for links
Problems with traditional audits

- **Cost**: lost engineering time, paying auditors, failed audits, etc.
Problems with traditional audits

- **Cost**: lost engineering time, paying auditors, failed audits, etc.
- **Judgment**: humans can make mistakes
Problems with traditional audits

- **Cost**: lost engineering time, paying auditors, failed audits, etc.
- **Judgment**: humans can make mistakes
- **Sampling**: not a proof that there is not a violation
Problems with traditional audits

- **Cost**: lost engineering time, paying auditors, failed audits, etc.
- **Judgment**: humans can make mistakes
- **Sampling**: not a proof that there is not a violation
- **Regressions**: only checked at audit-time
void makeCipher() {
    Cipher.getInstance("AES");
}

Compliance code example
void makeCipher() {
    Cipher.getInstance("AES");
}

String
void makeCipher() {
    Cipher.getInstance(“AES”);
}

@StringVal(“AES”) String
Compliance code example

```java
void makeCipher() {
    Cipher.getInstance("AES");
}

@StringVal("AES") String

Type qualifier
```