# George Kaline III

02/23/25

Project Two: Security Policy Presentation

<https://youtu.be/dYw616K0qFc>

Complete this template by replacing the bracketed text with the relevant information.

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Welcome everyone. Today I'll be presenting Green Pace's Security Policy Framework. I'm George Kaline III, and this presentation was prepared in February 2025. Let's dive into how we can enhance our security measures. |
| **2** | The foundation of our security approach is Defense in Depth. Think of it as a castle with multiple walls - an attacker who breaches one layer still faces several more before reaching valuable assets. We've implemented a comprehensive 10-layer strategy, each addressing different security aspects. Let me walk you through how these layers work together. |
| **3** | Here we have our threat assessment matrix, categorizing various security standards by likelihood and priority. As you can see, we have several high-priority concerns we need to address, including STD-003-CPP, STD-004-JAV, and STD-005-CPP. These represent our most pressing security challenges that require immediate attention. |
| **4** | These ten fundamental principles form the backbone of our security strategy. I'd like to highlight a few key ones: First, input validation is crucial and affects multiple standards including STD-001-CPP and STD-002-CPP. We also emphasize keeping systems simple, as complexity often breeds vulnerability. Each principle maps to specific coding standards, ensuring comprehensive coverage. |
| **5** | Let's examine our coding standards in detail. Each standard addresses a specific security concern. For instance, STD-001-CPP ensures correct data type handling, while STD-004-JAV specifically prevents SQL injection attacks. These standards aren't just guidelines - they're mandatory requirements for all our development work. |
| **6** | Our encryption strategy operates on three fronts: at rest, in flight, and in use. Data at rest encryption protects stored information, while encryption in flight secures data during transmission. Perhaps most importantly, encryption in use ensures data remains protected even during active processing. This comprehensive approach ensures continuous protection throughout the data lifecycle. |
| **7** | The Triple-A framework - Authentication, Authorization, and Accounting - forms our access control backbone. Authentication verifies identity, authorization determines access rights, and accounting tracks all activity. Together, these create a robust security system that not only prevents unauthorized access but also maintains detailed audit trails. |
| **8** | Moving to our unit testing framework, we've implemented four critical tests: Collection Clearing, Erase Verification, Capacity Reserve Testing, and Null Checking. Each test ensures specific aspects of our collection handling remain secure and efficient. |
| **9** | Moving to our unit testing framework, we've implemented four critical tests: Collection Clearing, Erase Verification, Capacity Reserve Testing, and Null Checking. Each test ensures specific aspects of our collection handling remain secure and efficient. |
| **10** | Moving to our unit testing framework, we've implemented four critical tests: Collection Clearing, Erase Verification, Capacity Reserve Testing, and Null Checking. Each test ensures specific aspects of our collection handling remain secure and efficient. |
| **11** | Moving to our unit testing framework, we've implemented four critical tests: Collection Clearing, Erase Verification, Capacity Reserve Testing, and Null Checking. Each test ensures specific aspects of our collection handling remain secure and efficient. |
| **12** | Automation plays a crucial role in maintaining our security standards. Our DevSecOps pipeline integrates security at every stage of development, making it everyone's responsibility rather than just the security team's concern. |
| **13** | We've implemented several key tools in our security framework, with special emphasis on CPPCheck for vulnerability detection. This tool has proven invaluable during our verification and testing phase, helping us identify potential issues before they become problems. |
| **14** | Let's consider the trade-offs of immediate action versus waiting. While acting now offers immediate problem resolution and competitive advantages, waiting can provide more time for analysis. However, the risks of delay often outweigh the benefits, particularly in security matters. |
| **15** | Our key recommendations focus on maintaining current security policies and establishing standardized policy sets for different project types. This approach allows for both consistency and flexibility in our security implementation. |
| **16** | To conclude, our success depends on four key factors: comprehensive staff training, current standards adoption, consistent use of security checkers, and regular internal audits. These elements together create a robust security framework. |
| **17** | For those interested in diving deeper, we've referenced authoritative sources including the SEI CERT C++ Coding Standard and CPPCheck documentation. These resources provide additional context for our security framework. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |