Project Two: Security Policy Presentation

Kevin Bristow

12/15/24

<https://youtu.be/0ckaHZqYi-g>

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

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Kevin and this is my presentation about security policy for the Green Pace company. |
| **2** | Green Pace uses the Triple-A security framework that uses a defense-in-depth as its foundation. Defense-in-depth is a strategy that implements multiple layers of security to protect the company’s data and assets. The thought process behind it is that if one layer fails, another layer will back it up. The security policy being presented here wants to implement 10 core security principles in an effort to build security into the original DevOps pipeline. There are also 10 standards that provide directions on how to adhere to the principles while developing. This policy in conjunction with other concepts such as encryption, unit testing, and standardization provide the defense-in-depth strategy. |
| **3** | This threat matrix shows where each coding standard exists in terms of likeliness and priority. Failing to implement a standard can result in a threat of varying magnitude.  STD-001 – Unlikely – P3  STD-002 – Likely – P6  STD-003 – Likely – P18  STD-004 – Likely – P18  STD-005 – Likely – P8  STD-006 – Unlikely – P1  STD-007 – likely – P4  STD-008 – Likely – P12  STD-009 – Likely – P6  STD-010 – Likely – P18  Automation such as static testing with CppCheck can locate such vulnerabilities before the code is put into production. |
| **4** | The 10 principles are: ValidateInput Data, Heed Compiler Warnings, Architect and Design for Security Policies, Keep it Simple, Default Deny, Adhere to the Principle of Least Privilege, Sanitize Data Sent to Other Systems, Practice Defense In Depth, Use Effective Quality Assurance Techniques, and Adopt a Secure Coding Standard. The corresponding standards are listed next to each principle which means that the standard can be used to adhere to the security principle. |
| **5** | Here are the 10 coding standards in their order of priority from threat level based on likeliness of occurring and severity. Level 1 is associated with high likelihood and high severity and Level 3 is associated with low likelihood and low severity. This basically means how dangerous is it if a developer fails to implement the standard such as accessing a string element out of range. |
| **6** | The encryption strategy for the policy comes in 3 forms: rest, flight, and in use. Encryption at rest is about having stored data encrypted so that in the event that a hack occurs, encryption protects that sensitive data. Encryption in flight encrypts data in transmission so it is not easily read by someone watching the network. Encryption in use refers to keeping data encrypted, even while we are using it. This strategy is computationally taxing but the data is always encrypted. |
| **7** | There are three facets to the Triple-A policy. Authentication – Confirms the identity of the user. It asks, “who are you?” and matches credentials to determine whether the user is valid or not. Authorization – Determines what a user is allowed to do. Once a user is authenticated, authorization takes over and permits what the user can and cannot do. Accounting – Tracks the resources used by users, session information, and data sent and received. Records all that goes on within the system. |
| **8-12** | In slides 8-12, I have included screen shots of unit test testing for the string correctness standard. Slide 8 shows the testing framework code on the right and the testing outcome on the left, with 4 passing tests. The point of this unit test is to test the behavior of strings.  Slide 9 tests whether the string does in fact exist after creating one.  Slide 10 tests what happens when accessing an out-of-range index.  Slide 11 tests if the string exists in memory after deleting it.  Slide 12 tests if we can access the memory of a deleted string.  The code presented in these unit tests should be used for implementing the string correctness standard because the unit tests prove that we can assure correct string behavior. |
| **13** | The DevSecOps pipeline is the continuous integration/continuous deployment lifecycle of producing production level software. Security has been added to DevOps in recent times because of the importance of security. |
| **14** | In the DevSecOps pipeline diagram there are a few areas where automation tools can be applied. In the build and Verify and test phases of pre-production, we can utilize tools such as dependency checkers or source code analyzers such as Cppcheck or Polyspace Bug Finder to find potential errors in the source code or to acknowledge potentially vulnerable dependencies our code may have.  On the production side with the Transition and health check and Monitor and detect phases, tools catering to penetration testing and reporting can be utilized. In this phase of the pipeline, automation tools such as LDRA tool suite can be used to automate testing of the code and to provide reports for audits and quality assurance. |
| **15** | Software is a double-edged sword: we can accomplish truly amazing feats with computers and software but non-secure software jeopardizes these feats.  Because humans are prone to creating software bugs, it is vital that security measures are used proactively instead of reactively to prevent the hack from ever occurring.  Waiting until the hack has occurred to fix the problem is expensive, breaks trust, and exposes sensitive information.  The solution to this problem is to implement preventative tactics such as Static tests, threat modeling, thinking like a hacker, unit testing, penetration testing, and policy creation. |
| **16** | For my recommendations, I would start with adopting a standard. Having a standard ensures that everyone is following the same set of rules and principles which will keep the code base uniform, simple, and traceable.  My second recommendation is to think like a hacker/threat modeling. Actively thinking about the system and potential threats can go a long way when protecting a system. In The Cuckoo’s Egg by Clifford Stoll, Clifford catches the hacker by thinking like them. He plants a file in his system that he believes the hacker will want, which will allow for the hacker to be traced which is what ultimately happens.  Thirdly, designing and planning programs with a security mindset builds that preventative layer of security into the program.  Lastly, social engineering attacks will cost the most secure systems from failing so it is very important to educate the team on attacks such as phishing and baiting. |
| **17** | In conclusion, this security policy provides a standard for Green Pace to follow and a way to integrate security into the existing DevOps pipeline.  The 10 standards and principle talked about in this presentation will be essential to follow for addressing security concerns and preventing future problems. This policy combined with the other security concepts like the Triple-A framework and encryption will provide a robust defense-in-depth security strategy. |
| **14** | [Insert text.] |