Southern New Hampshire University

CS 405 Project Two Script Template

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

Slide Number	Narrative
1	Hi. I'm Matt Jackson, a bachelor's student at Southern New Hampshire University studying computer science with a concentration in software engineering. This presentation outlines the security policy for Green Pace, a software engineering company focusing on projects for environmentally responsible entrepreneurs. At Green Pace, the security mission is defense in depth.
2	Green Pace developed this security policy to ensure that the systems and information both we and our customers use are safe and limit the impact of any compromise. By layering security principles and standards, a defense-in-depth approach will help ensure there is no one critical point of failure.
3	According to the cybersecurity company Crowdstrike's 2024 Global Threat Report, eCrime accounted for 84% of intrusions. (Crowdstrike, 2024) The information Green Pace and its customers use, and store includes personally identifiable and financial information, thus we have characterized eCrime as likely. Our priority threat is Social Engineering as the European Union Agency for Cyber Security has named it as a most important initial vector for cyber-attacks. (European Union Agency for Cyber Security, 2024) As a low priority, we list hacktivism, as our customers' environmental goals may have political ramifications. We do not see hacktivism as a threat to our customers' sensitive information. An unlikely but high impact threat is sabotage. Our customers may service environmental infrastructure like water treatment plants or pipelines. For example, in 2011, a group of hackers took control of a networked drinking water pump and broke it by turning it off and on repeatedly. (Cyberseurity Guide Contributors, 2024)
4	Without Secure Coding Standards, there is no benchmark against which to judge the security of our products. Defense in Depth ensures no one critical point of failure. Quality Assurance happens throughout the design and development process. Building security principles into our designs is more effective than bolting it on at the end. By limiting access to necessary processes and systems, a compromise of one shouldn't lead to deeper problems. Specifying what actions and processes a given user has rather than listing those they don't ensures risks aren't forgotten. Untrusted input can lead to unauthorized or unexpected behavior. Only send the information needed for functionality of downstream products. Warnings are there for a reason. Ensure you understand them and work to avoid them. Simple systems are easier to understand, update and defend.

Slide Number	Narrative
5	Coding standards are based on security principles. Multiple principles can apply to standards. Properly deallocating objects, handling exceptions and making declarations unambiguous are standards derived from the principle of Architect and Design for Security Policies. Ensuring data values don't wrap around, access values are within the container range, parameterizing SQL input and not using assertions to validate input are part of the input validation principle.
6	Encryption at rest is all data not currently being used or transmitted. This typically requires the strongest form of encryption as the duration of this state may be prolonged. In flight data is moving from one location to another, be it through an email, file transfer protocol, web upload/download or the like. The best way to secure data in transit is through shared key encryption, most often asymmetrical, which solves the key distribution problem. When data is opened by an application or user, it is considered in use. Data in use can be protected by access controls and permissions.
7	Authentication is identifying a user. Logins with passwords and unique usernames are a form of authentication. The in-use policy can be applied here because the user will be accessing various forms of data while they are authenticated. These are the tasks and data the authenticated user is allowed to perform or access. Once the user is authenticated, their authorization policy is applied. Database changes must only be made by authorized users. Levels of access and files accessed are governed by authorization policies. Accounting logs information about a user's session that can be used to verify authentication and authorization policies are being properly applied. It can also be used for analysis of resource usage to compare against a baseline to look for abnormalities or for design planning. New user creation should be logged and reviewed regularly. By monitoring levels of access and files accessed by users, authorization policies can be checked for compliance and enforcement.
8	These four tests demonstrate how crucial unit testing is to software security and Quality assurance.
9	Here we create a collection with a random number of entries and ensure adding five more entities increased the size that much. This helps validate input data.

Slide Number	Narrative
10	This test takes a random sized collection, decreases it by a random amount, reduces the size of the collection and verifies the size has decreased by the appropriate amount.
11	Here we attempt to access a collection element beyond the size range resulting in an off-by-one error (because the first element is zero, and the last is size -1) and an exception is thrown. This is an example of a negative test because our attempt to access the element fails.
12	We create a collection of unique values, pop the last one, then search for it. Again, we expect this to fail, and it does.
13	All of the tests have passed, meaning the code is running as expected, but it does not guarantee the code is threat-proof.
14	During the Assess and Plan and Design portions, automation should be considered alongside the design of the program or system. Researching and evaluating the types of tools available to help ensure security requirements are met should be done as soon as the requirements are established. During the Build and Verify and Test portions of the DevOps process, automated tests can be written with each portion of code as necessary. This also helps ensure that changes to the code still result in correct behavior. The push from pre-production to production environments can be enhanced by Continuous Integration (CI) and Continuous Deployment (CD) methods. CI can automate building, testing and regression testing of products and provide critiques to developers. CD can automatically push changes once they've passed testing. In the production environment, log collection and event alerting are best automated because of the sheer volume of data that can be gathered. Some responses, like cutting off access to an IP address that is trying to login with many different usernames can be automated as well.
15	Some tools that can be used at various stages of the pipeline include: CODESonar, which checks for tainted data, buffer issues, dangerous memory access, integer and floating-point overflow, and other common security coding errors. It can be especially useful during the Development and Testing phases of the DevSecOps cycle. (CODESonar) Sonar Cube is a static analyzer that offers a free and open-source IDE-plugin. (SonarQube, 2024)PVS-Studio is a static analyzer that can review commits and pull requests, integrate into other

Slide Number	Narrative
	tools like SonarQube during verify and test and quality control stages, and in continuous integration services. (PVS-Studio, 2024) SpotBugs focuses on Java code and is another freeware solution that can be integrated with automated build products like Ant or Maven. (SpotBugs, 2024) Polyspace BugFinder dynamically checks software at runtime and statically before compilation. Because of its dynamic capabilities, it may be good for checking live code during the deployment phase. (MathWorks, 2024) GCC is another open-source static analysis tool that supports the GNU project and can check for common bugs like double free problems or information leaks. (Free Software Foundation, Inc., 2024)
16	The question is whether to implement the security policy now or wait until something goes wrong to address the specific problems then. The cost of acting now is increased production time which may lead to delayed product delivery which may be outside the agreed upon terms of current contracts. However, having identified that threats to our and our client's systems do exist, it would cost more in terms of response time and money to respond after the fact.
17	The strategy only considers historic and anticipated threats. As technology evolves and systems expand, so do tactics and vectors. Security must evolve with it and be continuously evaluated throughout the DevOps cycle. We have only identified 10 secure coding principles and standards to address them. Many more threats exist than they address. Once our product is deployed, who is responsible for security maintenance, us or the client? We do not currently have a training program established. Like a security policy, we need to develop a data recovery plan and incident response plan.
18	Splunk, who provides a platform for data analysis with an emphasis on cybersecurity, has written an entire book on just the top 50 cyber security threats including Account Takeover, Phishing, SQL injection, web session cookie theft, and cloud storage. (Splunk, 2023) Partnering with a third party that provides security as a service would provide consistent protection across our systems and our clients'. At the very least, responsibilities at each stage of the DevSecOps cycle should be outlined during requirements analysis and contracting development. Training of employees, especially regarding current security policy and our priority threat of social engineering, should be done at onboarding and annually. Education of clients on threats to deployed products is also important.

Slide Number	Narrative
	Some guidelines to create a disaster recovery plan include asset inventory, identification of critical resources (and data), define recovery objectives, assess risks (which this document starts to do), establish a plan, identify key team members, develop a communication plan, document infrastructure, set procedures, *test the plan*, and keep it updated. (Kreisa, 2024)
19	More information can be found at these resources. **References** CODESonar. (n.d.). *CODESonar Datasheet.* Retrieved from codesonar.com: https://codesecure.com/wp-content/uploads/2024/12/CodeSonar-8.3-Datasheet.pdf Crowdstrike. (2024). *Global Threat Report.* Retrieved from go.crowdstike.com: https://go.crowdstrike.com/rs/281-OBQ-266/images/GlobalThreatReport2024.pdf Cyberseurity Guide Contributors. (2024, 04 15). *Safeguarding the environment: Cybersecurity in environmental protection*. Retrieved from cybersecurityguide.org: https://cybersecurityguide.org/industries/environmental-protection/ European Union Agency for Cyber Security. (2024, 06). *ENISA Threat Landscape.* Retrieved from ensia.europa.eu: https://www.enisa.europa.eu/sites/default/files/2024-11/ENISA%20Threat%20Landscape%202024_0.pdf Free Software Foundation, Inc. (2024). *GCC, the GNU Compiler Collection.* Retrieved from gcc.gnu.org: https://gcc.gnu.org Kreisa, M. (2024, 07 24). *How to create a disaster recovery plan*. Retrieved from pdq.com: https://www.pdq.com/blog/how-to-create-a-disaster-recovery-plan/ MathWorks. (2024). *Polyspace Bug Finder.* Retrieved from mathworks.com: https://www.mathworks.com/help/bugfinder/index.html PVS-Studio. (2024, 11 15). *PVS-Studio manual.* Retrieved from pvs-studio.com: https://pvs-studio.com/en/docs/ SonarQube. (2024). *SonarQube for IDE Documentation.* Retrieved from sonarsource.som: https://docs.sonarsource.com/sonarqube-for-ide/intellij/ Splunk. (2023). Top 50 Cybersecurity Threats. Splunk. Retrieved from https://www.splunk.com/en_us/pdfs/gated/ebooks/top-50-cybersecurity-threats.pdf SpotBugs. (2024). *SpotBugs.* Retrieved from spotbugs.github.io: https://spotbugs.github.io/