# CS-405 Project Two: Security Policy Presentation Script

Ed Morrow

Southern New Hampshire University

Professor Mike Prasad

20th April 2023

Link to Presentation Video on YouTube: [**https://youtu.be/wm4nauOgiak**](https://youtu.be/wm4nauOgiak)

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, and welcome. This presentation will outline the proposed new security policy for Green Pace. While many of you may have already been implementing many of the proposals described within the policy, it is important to be sure that these processes are standardized going forward to create a coherent and manageable Defense in Depth (DiD) strategy which can be monitored and adapted to meet the ever-evolving challenge presented by hackers the world over. |
| **2** | Our policy will be based upon a Defense in Depth Strategy. DiD involves multiple, redundant layers of overlapping security measures, ranging from physical security for all sites and for all devices, through encryption, Authentication, Authorization and Accounting (AAA) policies and coding standards to address specific security vulnerabilities.  The thinking behind DiD is that no system can be truly secure, but by creating a multilayered strategy we can create the most secure system possible because if one measure should fail to protect it, others which overlap may nevertheless do the job.  While DiD does carry certain drawbacks in terms of cost and manpower requirements, the initial outlay required to train developers to work to the required standards pays dividends in the long run because it leads to coherent and easy-to-manage processes which create a system with multiple layers of safeguards in place. |
| **3** | As a preliminary to creating the policy, we considered possible threats to the system and attempted to come up with a provisional ranking for them. The threat matrix shows that potential threats were measured along two axes, the likelihood that they would occur and the severity if they should do so.  While the ranking would certainly be subject to constant review, the method we used was to consider the potential severity as the primary factor, and the perceived likelihood as a secondary factor. This is because, in part, it is easier to quantify the severity of a security vulnerability than it is the likelihood of it existing and being exploited; and also because, in an imperfect world with limited resources, it makes sense to tackle the most dangerous issues first.  An example of a very severe threat would be that of SQL injection which could, potentially, expose entire databases to attackers. This would thus represent one of the highest priorities in our security policy. |
| **4** | In addition to the threat matrix, we have also proposed ten principles which will guide our security policy. As you can see, at least one of our ten proposed coding standards maps to each of the principles, and in fact most principles map to more than one standard. Ensuring that our coding standards are aligned with the principles helps to keep our focus on code security throughout the development process. |
| **5** | Here we have the list of coding standards ranked according to their priority. Again, this ranking is provisional, but has been guided in the first place by considering potential severity first and perceived likelihood second. |
| **6** | Encryption will form a hugely important part of our DiD strategy, and should be implemented for all data, be it at rest (*i.e.* stored but not in use), in flight (being transmitted) or in use.  While encryption should be utilized across the board, another layer of our DiD strategy will be to attempt to limit storage of the most sensitive data to as few locations as possible to ease the task of implementing additional, physical security at those sites. |
| **7** | The triple-A framework refers to authentication, authorization and accounting: authentication, or the process of ensuring that all users have to present valid login credentials to access our systems; authorization, which deals with finer controls over which users, after authentication, may or may not access particular data or systems and perform certain actions; and accounting, which deals with monitoring access and actions to ensure that the authentication and authorization elements are doing their job and to take action if not. This last task can be aided by AI security analytics which frees up our people’s time, which might have been spent on analysis, to take action where needed instead. |
| **8** | Unit testing is an essential element of security because it allows us to focus on specific elements of code and to check that they work correctly across multiple use-cases before integrating them with existing the codebase.  The examples that follow were written with Google Test and show how we might test the behaviour of a particular data structure (a collection, in this case) in line with coding standard STD-003-CPP, which deals with the valid range of container indices and iterators. |
| **9** | The first test checks to see that the collection is empty upon creation, and thus has a size of zero, which is the behaviour that we would expect. |
| **10** | The next test adds an element to the newly created collection. Since the new collection would have had a size of zero (per the previous test) we check to see that the size has increased to one after the addition of an element. |
| **11** | This tests the behaviour of the resize() method and confirms that it has indeed increased the size of the collection, as we would expect. |
| **12** | The final example considers the case of an attempted pop\_back() call on an empty collection. While a range of responses would be possible here, we would like such an action to raise a fatal error, and that is what this test seeks to verify. |
| **13** | Of course, these are just a few examples of the kinds of tests we would have to run against the Collection object. The examples you just saw formed part of a test suite which gives us a very comprehensive picture of how the object would perform. The screenshot shows the console readout after running a test suite (which included the four examples just discussed) and, as you can see, in this case all of the tests were successful! |
| **14/15** | We would seek to automate security measures at several points in the DevSecOps process. Not only can this reduce the possibility of human error and ease the process of auditing security standards, but it can also free up developers’ time to work on addressing any security issues we might identify and to continue to adapt the policy as necessary.  Stages of the process that would particularly lend themselves to automation would include the *build* stage, where we can use tools from the Open Web Application Security Project, among others, to ensure the libraries our code depends on are secure and regularly maintained; unit testing, which we just discussed, during the *verify and test* stage, using tools like Google Test; penetration testing as part of the *transition and health check* stage, where resources like Astra Pentest could aid in automation; and in the monitor and detect stage, using tools like the ManageEngine EventLog Analyzer.  As with all other aspects of the policy, the specific tools we use would be subject to review: these are just suggestions. And while we would continue to include automation at the stages just discussed, we would also be on the lookout for other areas where automation might be employed in the future. |
| **16** | So, what are the risks and benefits of acting now versus waiting until later to overhaul our security policies?  As we can see, the major drawback in acting now is the time and manpower requirements to train developers to adopt the new policies in a rigorous and systematic way. However, given the increasing frequency of major attacks (and we’ll discuss an example in a moment), we cannot postpone the issue of security indefinitely — we will have to act sooner or later — and we risk serious financial and reputational damage in the meantime if we do not make security central to everything we do.  Another drawback to delaying is that we will have ever more ground to cover in securing our legacy code to the same standard as new code; and we risk falling behind in the never-ending race against would-be attackers and finding ourselves in a situation where we’re struggling just to keep our heads above water. On the other hand, the sooner we adopt a comprehensive security policy, the sooner we might be in a position to actively preempt attacks rather than dealing with them as they happen. |
| **17** | Our recommendation, then, is to implement the security policy as quickly as possible and employ automation and AI analytics to speed its adoption and to broaden its scope. We should also roll out training for all developers and testers to ensure that we are all on the same page when it comes to security, which will make the accounting side of things much easier.  At the same time, we should recognize that this policy is just the beginning, and that we should continually reevaluate it in light of feedback from our development teams and of the data we get from the accounting and analysis processes. |
| **18** | In conclusion: security is an issue that we simply cannot afford to ignore, and the sooner we act to implement a comprehensive security policy that we can build upon going forward, the better placed we will be to preempt and neutralize possible attacks.  I mentioned earlier that we would look at an example of a ‘successful’ attack, and I wanted to discuss the web scraping operation against the Alibaba Group. This was perpetrated by an employee who had access to their systems as a developer, but abused the trust placed in him to scrape data from a company API to share with his business partner in their own commercial venture. It is estimated that they stole over 1 billion data items before their ruse was discovered, and the company suffered very serious reputational damage as some of the data stolen (including mobile telephone records) is considered to be sensitive in the PRC as by law it must be linked to other personal information.  Looking at our proposal for a DiD security policy, we can see that the authorization and accounting facets of the triple-A policy could have helped to prevent such an attack. While an ‘inside job’ such as this would be particularly difficult to protect against, proper accounting processes could have detected this issue within days or even hours, rather than the months it took Alibaba to realize what was happening.  If we want to avoid an embarrassing incident of this sort — and remember, it could have been much worse if the company had also incurred financial losses due to data being sold on to competitors — we need to act now to take a proactive stance against possible security threats. |
| **19** | Thank you very much for listening, and please let me know if you have any questions! |