1. Introduction
   1. Personal
   2. Why “Security By Design”
      1. Cybercrime on the rise - “This represents the greatest transfer of economic wealth in history, risks the incentives for innovation and investment, and will be more profitable than the global trade of all major illegal drugs combined.” (Morgan, 2017)
      2. Average cost of a data breach - $3.86 million (McCarthy, 2018)
      3. Consequences of not doing it – cost of breach -
      4. If not done by design, increased likelihood of vulnerabilities.
      5. Beauty, Excellence, Quality… something that is exactly as it should be.
2. An Overall picture
   1. The Software development lifecycle
   2. The basics of web application security
      1. Strong Authentication
      2. Strong Authorization
      3. Protect Data
      4. Validate All Input
      5. Sanitize All Data
      6. Basic Security Headers
3. Analysis
   1. Answer “What are we going to do?”
   2. Interactive demo – developing an address database
   3. Security must be as important as functionality
   4. Training/Negotiating with the customer
   5. Security requirements
      1. What is expected in terms of security
      2. Do not put those fundamentals into a template
4. Design
   1. Answers “How are we going to do it?”
   2. Environment
      1. Personnel – Security awareness
      2. Language/Framework
   3. Layout of the application
      1. defense in depth
         1. We do not control the front end
         2. Every module, every layer
      2. Separation of data access – service accessible by whitelist only
   4. Threat modeling!
      1. Purpose: “Build a secure design”
      2. A slight disagreement with OWASP – “Threat modelling in the SDLC can help… bring Security and Development together to collaborate on a shared understanding, informing development of the system” – can help? Yes, but integration of security and development should already be in place.
      3. Answers
         1. “What are we building?” – but more in depth – architecture diagrams, dataflow transitions, data classifications
         2. “What can go wrong?”
         3. “What are we going to do about that?”
         4. Are we doing a good enough job about that – answered on iterations back from later phases.
      4. Whiteboard threat modelling – constructing a model, with varied stakeholders, fluid, changing to meet identified threats.
   5. Do use a template or sprint planning checklist
   6. And a note on designing/planning your project, as it pertains to personnel… team must be…
      1. A bit paranoid
      2. Open to input, scrutiny
      3. Learner!
5. Implementation
   1. Basics
      1. Require Secure Transport, even for all external resources
      2. Strong Authentication – 2FA, Central Authentication
      3. Validate and Sanitize input and output - use whitelisting  
         “The most common web application security weakness is the failure to properly validate input from the client or environment. This weakness leads to almost all of the major vulnerabilities in applications” (OWASP, 2019)
      4. Basic headers – X-Frame-Options, X-Content-Type-Options, X-XSS-Protection, HSTS, etc.
   2. Defense in Depth
      1. Across every module and down every layer
      2. Use base page
         1. every page derives from it
         2. every page asserts authorization
      3. Separate data access to Service or API
         1. Only allows access from web application server
         2. Uses strong authentication
         3. Again, validation and sanitization
         4. Uses type-safe parameters
         5. Uses Parameterized queries ONLY!
         6. Uses least privileged account
      4. Understand and use identity protection in the page life cycle
         1. How is the session id and other session information retained?
         2. Is it encrypted?
         3. Does it carry a signature hash for verification on postback?
6. Testing (Beaver, 2019)
   1. Define scope of testing
   2. Make use of OWASP testing checklist (OWASP, 2019)
   3. Use best tools
      1. Vulnerability scanner
      2. Pen testing tools (Zap, Burp, Metasploit, etc)
      3. Possible automated source code analysis
   4. Scan for vulnerabilities
      1. do a credentialed scan, with a web server and web app savvy profile/policy
      2. Potential for multiple scans with multiple tools – ZAP, Burp Suite.
   5. Manual testing
      1. Verify results of the scan
      2. Do additional penetration testing – Owasp Top 10 in mind
   6. Document all findings
      1. Start with summary of strengths
      2. Summarize and document score of the vulnerabilities
   7. Process remediation.
7. Deployment
   1. Run an audit
      1. Scan
      2. Pen testing
      3. Code review
   2. Separate test from production
      1. Separate application and database servers
      2. Do not include test accounts in production server
   3. Consider the use of a Web Application Firewall
8. Maintenance
   1. Iterate the software development lifecycle - Circling back to analysis, design, etc.
   2. On user maintenance requests
   3. On regular basis
      1. Regular scans and demand remediation
      2. Example – Public IP’s scans

# References

Attwood, J. (2015, April 23). *Your Password is Too Damn Short.* Retrieved from Coding Horror: https://blog.codinghorror.com/your-password-is-too-damn-short/

Beaver, K. (2019). *5-Step Checklist for Web Application Security Testing.* Retrieved May 2019, from SearchSecurity.TechTarget: https://searchsecurity.techtarget.com/tip/5-step-checklist-for-web-application-security-testing

Kubenka, K., & Munoz, G. (2019). Assessing You Web Server and Application's Security. *Tech Summit 2019.* Galveston.

McCarthy, N. (2018, July 13). *The Average Cost Of A Data Breach Is Highest In The U.S.* Retrieved May 9, 2019, from Forbes: https://www.forbes.com/sites/niallmccarthy/2018/07/13/the-average-cost-of-a-data-breach-is-highest-in-the-u-s-infographic/#3efad0882f37

Microsoft Security Development Lifecycle Resources. (2019). *Resource List.* Retrieved March 2019, from Microsoft - Security Engineering - Security Development Lifecycle: https://www.microsoft.com/en-us/securityengineering/sdl/resources

Morgan, S. (2017, October 16). *Cybercrime Damages $6 Trillion By 2021.* Retrieved May 9, 2019, from Cybersecurity Ventures: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/

Munoz, G., & Tarpley, J. (2017). Addressing Web Application Security. *Tech Summit 2017.* Galveston.

OWASP. (2018, December 28). *Clickjacking Defense Cheat Sheet.* Retrieved from Github: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Clickjacking\_Defense\_Cheat\_Sheet.md

OWASP. (2019). *Application Threat Modeling.* Retrieved April 2019, from Open Web Application Security Project: https://www.owasp.org/index.php/Application\_Threat\_Modeling

OWASP. (2019). *Data Validation.* Retrieved May 2019, from Open Web Application Security Project: https://www.owasp.org/index.php/Data\_Validation

OWASP. (2019, March 31). *SQL Injection Prevention Cheat Sheet.* Retrieved from Github: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/SQL\_Injection\_Prevention\_Cheat\_Sheet.md

OWASP. (2019). *Testing Guide.* Retrieved Mar 2019, from Open Web Application Security Project: https://www.owasp.org/index.php/OWASP\_Testing\_Guide\_v4\_Table\_of\_Contents

Romeo, C. (2018, January 23). *Secure Development Lifecycle: The essential guide to safe software pipelines.* Retrieved from TechBeacon: https://techbeacon.com/security/secure-development-lifecycle-essential-guide-safe-software-pipelines

Sucuri. (2018). *Website Hack Trend Report 2018.* Retrieved May 9, 2019, from Sucuri: https://sucuri.net/reports/2018-hacked-website-report/