Secure Coding.
Practical steps to defend your web apps.

Interested in learning more?
Check out the list of upcoming events offering "Defending Web Applications Security Essentials (DEV522)"
at http://software-security.sans.org/events/
Introduction

There was a time when the security of web applications was considered of value only in keeping an organization compliant with industry payment standards or government regulations. But web app security is now considered mission-critical for most organizations. User demand drove organizations of all kinds to shift development resources to web-based, customer-facing applications, whose features and content can change quickly to keep pace with changes in a company’s market, its competition or its strategy. Today’s web-based applications are complex, depending on a variety of programming languages, web servers and databases, and are accessed by a variety of client browsers, each with known security issues. Cloud platforms make it easier to scale infrastructure to match demand, but they make the security model even more complex, especially for enterprises that must manage hundreds, or even thousands, of web apps that are often highly sophisticated and reside on platforms not necessarily under the control of IT or enterprise security operations.

In this whitepaper, SANS lays out an approach to help organizations gain maturity in establishing and maintaining web application security at scale across the enterprise. This approach incorporates advice from industry leadership: Capability Maturity Model Integration (CMMI) for maturity models, the Open Web Application Security Project (OWASP) for testing frameworks, and SANS for security best practices.

Vulnerability management plays an indispensable role in maintaining security throughout the web application life cycle. Vulnerability management depends on vulnerability scanning for evaluating an app in situ, where developers may not have direct access to all source code or administrators to the underlying server configuration.

As organizations look to gain maturity in web application security, how should they approach the challenges inherent in improving their vulnerability management posture? And how should they approach the selection of automated tools to make scanning more efficient and effective?
**Maturity in Web App Security**

How can an enterprise evaluate how mature its overall *organizational* processes are or how effective they are at achieving a desired state of security for its web-based applications? A capability maturity model provides a reference framework of mature practices around a discipline, be it development, service delivery or procurement. The organization can compare its current practices to the framework to identify potential areas for improvement.

Perhaps the best known of these frameworks, the CMMI, was developed at Carnegie Mellon University in 1987. Today, administered by the CMM Institute, it is recognized globally—the program is used in 101 countries and by 11 national governments, with translations into 10 languages.¹

CMMI for Development (CMMI-DEV) is a framework for building systems that allow organizations to improve their processes for product and service development, maintenance and acquisition, and, as a result, to systematically improve customer satisfaction. CMMI-DEV provides criteria for evaluating process capability and process maturity.

Security touches almost every aspect of an organization. Purposeful, planned security is more reliable than ad hoc security. “Security by Design” was introduced to seamlessly integrate security processes into the related process categories of CMMI-DEV. Its four process areas are shown in Table 1. Organizations following the CMMI program in those four areas gradually adapt (or design) their workflows, processes and process assets to ensure that security measures are introduced by design, not by accident, into the entire product development life cycle.²

---

### Table 1. Overview of the Security Process Areas

<table>
<thead>
<tr>
<th>Security Process Area</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizational Preparedness for Secure Development (OPSD) (process management)</td>
<td>Establish and maintain capabilities to develop secure products and react to reported vulnerabilities.</td>
</tr>
<tr>
<td>Security Management in Projects (SMP) (project management)</td>
<td>Establish, identify, plan and manage security-related activities across the project life cycle and to manage product security risks.</td>
</tr>
<tr>
<td>Security Requirements and Technical Solution (SRTS) (engineering)</td>
<td>Establish security requirements and a secure design to ensure the implementation of a secure product.</td>
</tr>
<tr>
<td>Security Verification and Validation (SVV) (engineering)</td>
<td>Ensure that selected work products meet their specified security requirements, and demonstrate that the product or product component fulfills the security expectations when placed in its intended operational environment.</td>
</tr>
</tbody>
</table>

---

¹ CMMI Institute, [http://cmmiinstitute.com](http://cmmiinstitute.com)

While CMMI-DEV provides goal-level definitions and attributes for specific processes (such as software, systems and security engineering), it is neither an engineering development standard nor a development life cycle framework. It does not provide operational guidance for performing the design and development work. That’s where looking at what the OWASP defines as best practices comes in.

The OWASP helps organizations develop, purchase and maintain software applications that can be trusted. It provides a comprehensive resource that ranges from best practices for development and testing—its Top 10 is generally acknowledged as the indispensable list of the most dangerous current web application security flaws—and the most effective methods of dealing with those flaws.

So the equation to establish a mature approach to web application security looks something like this:

\[
\text{Organizational process maturity} + \text{Development and operational best practices} = \text{Better application security across the web app life cycle}
\]

Now let’s apply this formula to web application security, vulnerability management and the use of vulnerability scanning.

**Focus: Vulnerability Management Across the Web App Life Cycle**

*Vulnerability management* is the ongoing process of identifying, evaluating and remediating weaknesses in an organization’s information systems and applications, going beyond basic scanning, by categorizing assets and classifying defects according to risk level. It needs to work closely with other key enterprise processes, such as change/configuration management or patch management. A single insecure application (or component of an application) can compromise the security of an entire network.

At the heart of effective vulnerability management programs is vulnerability scanning. SANS has long advocated that organizations strive toward continuous scanning to pick up on the inevitable and constant change in the production network and computing environment. A malicious actor can discover a serious software defect that compromises the security of the application. SANS also advocates scanning (a.k.a. dynamic application security testing, or DAST) as part of software testing, especially as applications move from unit testing to system and integration testing.
Table 2 provides a crosswalk of how the elements of CMMI-DEV, the OWASP Testing Framework and SANS advice on the use of vulnerability management, including vulnerability scanning, come together to support secure web applications, regardless of the development approach taken (e.g., Agile, waterfall, spiral) or the use of commercial versus custom code.

<table>
<thead>
<tr>
<th>Application Life-Cycle Stage</th>
<th>Drivers</th>
<th>CMMI Security by Design (Process Area)</th>
<th>OWASP Testing Framework</th>
<th>SANS Advice on Vulnerability Management Aspect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept development (pre-design, development and integration (pre-DD&amp;I))</td>
<td>Strategic needs and investment</td>
<td>Organizational Preparedness for Secure Development (OPSD): Following CMMI-DEV objectives; these capabilities would be established at the organizational (or enterprise) level.</td>
<td>Pre-development: 1) Define the software development life cycle approach (e.g., waterfall, Agile), with security inherent at each stage; 2) establish policies, standards and documentation in place (e.g., Java secure coding standard if app to be developed in Java); 3) develop measurement and metrics, ensure traceability.</td>
<td>Determine tools and processes to be used in vulnerability management around web applications, looking toward more comprehensive and automated testing tools that can support continuous testing as development methods become more agile (a.k.a. DevOps).</td>
</tr>
<tr>
<td>Project planning and project management</td>
<td>Meet scope, schedule and resource demands; contain costs.</td>
<td>Secure Management in Projects (SMP)</td>
<td>Policy and standards development</td>
<td>Integrate vulnerability management into key project management activities (e.g., risk assessment and management); develop plans and approach for threat modeling.</td>
</tr>
<tr>
<td>DD&amp;I</td>
<td>Maintain secure development and engineering practices throughout DD&amp;I.</td>
<td>Secure Requirements and Technical Solutions (SRT); Security Verification and Validation (SVV)</td>
<td>Development steps: • Define/design • Develop • Deploy</td>
<td>Implement vulnerability management in development; use vulnerability scanning for both code and infrastructure testing; establish vulnerability management as part of the overall change management and acceptance processes for web apps.</td>
</tr>
<tr>
<td>Maintenance and operations (M&amp;O) (production)</td>
<td>Keep the investment secure; close the window of vulnerability in deployed applications.</td>
<td>Establish Security by Design processes, interfacing with key CMMI-DEV ones: configuration management (CM), measurement analysis (MA), process and product quality (PPQ).</td>
<td>Vulnerability management activities triggered by: • Health checks • Change verification • Changes in operations • Identification of new threats and/or vulnerabilities in the landscape</td>
<td>Repeat DD&amp;I vulnerability management process as needed: • Maintain (CM). Did a change in the infrastructure create a vulnerability? • Improve (PPQ). How can you improve the time to patch? • Assess (MA). How do you measure what is needed to combat new exploits or threats?</td>
</tr>
</tbody>
</table>
Security by Design emphasizes establishing the organizational capabilities for the development and maintenance of secure products that can be used across multiple projects and engagement. These processes and tools can then be drawn upon for various projects. Vulnerability management is one of the specific goals for the OPSD process area. (See sidebar titled “OPSD Specific Goal and Practice Summary.”)

**Planning for Process**

How should an organization go about investing in the processes and tools needed prior to engaging vulnerability management in any project? SANS suggests the following four-step approach based on the specific objectives under OPSD SP 1.5.

**Step 1: Standardize Processes and Tools for Handling Vulnerabilities**

Smaller organizations work within basic budget constraints; larger organizations may face the same issues because of limited funding due to lack of management commitment. But one should not despair! Organizations can take an evolutionary approach to establishing and maintaining processes and supporting assets (tools and repositories). Start small and grow over time.

The first four families of the Center for Internet Security’s (CIS) Critical Security Controls, together with CIS Control 18, can be considered a foundational set of requirements for web app security:

- CSC 1: Inventory of Authorized and Unauthorized Devices
- CSC 2: Inventory of Authorized and Unauthorized Software
- CSC 3: Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers
- CSC 4: Continuous Vulnerability Assessment and Remediation
- CSC 18: Application Software Security

Don’t neglect that the organizations should also invest in necessary training and awareness for all stakeholders—end users, operations staff and management—and don’t forget the developers!

---

3 Center for Internet Security, [www.cisecurity.org/critical-controls/Library.cfm](http://www.cisecurity.org/critical-controls/Library.cfm)
Step 2: Prepare Vulnerability Records and Reports (Standards, Formats and Content)

Data on vulnerabilities needs to be collected in one place, both from internal systems and external sources, as well as vulnerability databases, such as the National Vulnerability Database, and advisories such as US-CERT alerts. Once various data sources have been collected, the challenge becomes presenting the derived information. A presentation that would convince a security analyst of the severity of a problem is not the same as what is needed to convince management. Metrics are key and should be ideally based on industry-accepted measures. The “Measurement Companion to the CIS Critical Security Controls,” for example, provides suggested metrics that can clearly communicate to management the need for those operational processes that, when approved, will help rapidly close windows of vulnerability.

Also, take into account the role that vulnerability management plays in the overall management of a web application, during both development and M&O. Managers like reports that track the status of vulnerability management, summarizing the number of vulnerabilities found, the number of resulting bug fixes or patches and the number of problems resolved.

Step 3: Create a Communication Plan around Vulnerability Management

Outcomes from vulnerability scans, especially critical issues, need to be communicated to stakeholders in an organized, effective and timely fashion. So why not establish a formal communications plan specific to vulnerability management?

The vulnerability management plan should at least address the following:

• Type of communication (alert, text, email, standard report, security advisory)
• Priority/urgency of communication (critical, high, medium, low)
• Primary party to contact (internal, external)
• Distribution list (if required)
• Method of contact (will vary between internal and external)
• Frequency of communication (daily, weekly, monthly for routine issues; within the hour for alerts)

The communication plan will be based on the infrastructure to which vulnerabilities can be reported (e.g., email address, hotline or intranet website). Be sure to consider out-of-band communications, which may be necessary for emergency outages. Be responsive to the media communication aspects of incident response.
Step 4: Ensure that Vulnerability Management Processes Have a Path to Continuous Improvement

Don’t stop once processes and tools have been defined and established as in Step 1. Expand the use of vulnerability management into other areas of the organization. Look to embed vulnerability management in other operational areas of the organization such as change/configuration management, policy compliance and risk management, as well as its more traditional role in security operations.

Automation: Not a Necessary Evil

Clear ownership advances security by providing a single point of decision making or control. Ownership of a web application, however, is often complicated by organizational structure. Major gaps often exist between managers of the business units on whose behalf an application operates and the development or operations teams responsible for building and maintaining it. Those gaps multiply in production environments that contain a mix of in-house and commercially developed applications that may be based on-premises or in the cloud. Best practices recommend that applications be designed with at least a three-tier architecture, with the presentation, application and persistence layers appropriately segmented and protected from one another. Servers in each tier should be hardened according to appropriate guidelines for the services they provide, including patching the operating system in a timely manner, turning off nonessential services, disabling default accounts and establishing appropriate logging and alerting. Failure to maintain a secure configuration across the tiers may not only compromise the web app, but also other organizational back-end systems and the entire enterprise network as well.

Web application security demands a holistic approach that identifies and remediates vulnerabilities in both software and the underlying infrastructure. Organizational gaps need to be identified and closed, especially between development and operations. Automation can help bridge these gaps, especially in large organizations.

A web application vulnerability scanner is “an automated tool that scans web applications to look for … security vulnerabilities such as cross-site scripting, SQL injection, command execution, directory traversal and insecure server configuration,” many of which can be a result of insecure or incorrect coding and design. General-purpose (network or system) scanners, on the other hand, can identify open ports, active IP addresses and logons, host operating systems and software types, revision and patch levels, and running services.

Figure 1 shows how these two scanner types can be integrated into a production environment to support CIS Critical Security Control families 4 (Continuous Vulnerability Assessment and Remediation) and 18 (Application Software Security). This can enable an organization to move toward mature vulnerability management with support for continuous monitoring and metrics that allow visibility into the state of enterprise security.5

Figure 1. Tools Called Out by CIS Controls Related to Vulnerability Scanning.

This also allows a natural progression from integration and end-to-end system testing done during development, where DAST is the critical methodology. Web application vulnerability scanners can quickly analyze the application in a test environment that is designed to simulate production as closely as possible. General-purpose (ideally, Security Content Automation Protocol-compliant) tools can be used to survey the test network and computing infrastructure, allowing developers and operators to develop expectations for monitoring the application in production.

5 Center for Internet Security, www.cisecurity.org/controls
Plan in Advance! (CONTINUED)

All indicators of vulnerabilities should be stored in a central repository, where the information can be correlated, prioritized and acted upon according to established rules. The generic “alerting/reporting analytics system,” shown in Figure 1, should include interfaces to the organizational security information and event management (SIEM) system, the help desk/trouble ticketing systems, and external development problem-reporting and tracking repositories such as GitHub. Visualization (e.g., reports and dashboards) reinforces enterprise decision making around prevention and protection. Data in this repository must also be available to organizational analytics and reporting platforms.

**What to Look for in a Vulnerability Scanner**

Scanners differ in the protocols they support, the authentication and session management methods they use, and the crawling and parsing capabilities they possess. Make sure that the scanner addresses the types of problems for which you need to test, including both architectural weaknesses and vulnerabilities in the web app. The OWASP Top 10 is a mandatory starting point for defining vulnerabilities, because it reflects the major ones being exploited in the real world, including injection flaws (e.g., SQL, LDAP, XPath), lack of input sanitization, cross-site-scripting, insufficient authentication and session weaknesses.

Consider how the scanner identifies vulnerabilities. Does it look mainly for known flaws as with a signature-based approach? Or does it leverage an extensive set of heuristic web vulnerability checks, backed by frequent updates from both research and the field that allows for improved identification of issues, including zero-day vulnerabilities, in custom-built applications? Does it also support sources of known vulnerabilities, such as the Common Vulnerability Scoring System (CVSS), to help your organization understand and prioritize remediation needs?

A robust set of technical requirements can also be derived from CIS Control families 1 through 4 (with an emphasis on 3 and 4), plus Control 18.

Next, explore how the scanner is rated for accuracy—the most important metric—as well as reliability, scalability and quality of reporting. Conduct a proof of concept against a benchmark, a test suite designed to evaluate the speed, coverage and accuracy of automated software vulnerability detection tools and services. Benchmarks can be based on industry-defined standards or defined by your organization, but you need to understand how the benchmark test presents its outcome.

Understanding the scanner tuning process is imperative. A benchmark test, for example, may present its results in terms of false/true positives and false/true negatives. If not configured correctly, an automated scanner might not crawl the target properly and can easily report false positives, which require manual verification—a time-consuming and often difficult process—to be done by security experts. This defeats the purpose of automation!
Therefore, during the benchmarking testing process, both the pre-scan and post-scan operations should also be evaluated. Does the scanner require a lot of configuration, or can it automatically fine-tune itself? Can it verify its own findings without the involvement of security experts? Does it have a reputation for reporting large numbers of false negatives?

If accuracy is lacking, you may end up running two different scanners, hoping that one picks up vulnerabilities that the other misses. This adds time, cost and effort to the scanning process, because an IT staffer has to spend double the time on the scanning process and also comb through two sets of scanning results to see what’s what.

Pay attention to reporting functionality—especially the flexibility the scanner offers in defining display and sort criteria—and the methods of presentation for best visualization. Does the tool provide a dashboard? Can it show how the results compare to compliance standards such as the Gramm-Leach-Bliley Act, the Health Insurance Portability and Accountability Act or the Payment Card Industry Data Security Standard? Can the tool group and tag scanned assets by location, business unit or priority?

Is the output from the scanner easily mapped to the common terminology adopted by your organization so it can be integrated with a common analytics platform for alerting and status reporting, as shown in Figure 1?

Finally, match desired features to budget and available resources. Don’t neglect getting a realistic estimate of the time and resources a tool demands for administration and analysis. Automation should reduce demands on both, allowing organizations with hundreds of web applications to concentrate on remediating, not just identifying, prioritized vulnerabilities.

Low-end vulnerability scanners may simply scan a network or system and provide remedial reporting. Higher-end products can further automate processes such as configuration auditing and enforcement, target profiling, penetration testing and detailed vulnerability analysis. It may come down to balancing the cost of a tool against the manpower needed to run it.

For small and mid-size companies, a vulnerability scanner that is a cloud-based, software-as-a-service (SaaS) tool can be beneficial. An on-premises setup would require a hefty investment in hardware and staff.

Cloud-based scanning may be even more ideal for large enterprises. It allows such organizations to easily scale up their operations without a significant investment in dedicated hardware and staff. Cloud-based scanning tools can provide continuous, hands-free monitoring of all of an organization’s assets on all network segments (perimeter to internal), as well as cloud services. A real-life example is an organization with 1,500 domains. An on-demand solution allows it to scan 1,000 websites in 24 hours. Installation, manual integration and maintenance aren’t required—the enterprise simply subscribes to the service, configures the scans and goes.
Vulnerability management supports the overall management of security-related activities and risks across the web application life cycle. The key objective is to obtain actionable information that can prevent any recurrence under the same or similar conditions. Vulnerability management project management processes should focus on Specific Practice 1.4 (identifying, analyzing and resolving the underlying causes of vulnerabilities) listed in the sidebar “Security Management in Projects: Specific Goal and Practice Summary.”

Vulnerability management is also an ongoing process. The organizational processes, discussed previously, can be incorporated into a specific project plan that is aligned with a plan-do-check-act cycle of key activities, as shown in Figure 2.

Figure 2. Vulnerability Management Follows a PDCA Cycle.
The engineering process area under CMMI-DEV aligns with the life-cycle phases designated as DD&I. There are two key sets of activities: The first is the creation of the technical solution, starting with requirements, and the other is testing, which encompasses both verification and validation.

Security Requirements and Technical Solutions: Specific Goal and Practice Summary

SG 1: Develop customer security requirements and secure architecture and design.
SP 1.1: Develop customer security requirements.
SP 1.2: Design the product according to secure architecture and security design principles.
SP 1.3: Select appropriate technologies using security criteria.
SP 1.4: Establish standards for secure product configuration.
SG 2: Implement the secure design.
SP 2.1: Use security standards for implementation.
SP 2.2: Add security to the product support documentation.

Security Verification and Validation: Specific Goal and Practice Summary

SG 1: Perform security verification.
SP 1.1: Prepare for security verification.
SP 1.2: Perform security verification.
SG 2: Perform security validation.
SP 2.1: Prepare for security validation.
SP 2.2: Perform security validation.

Security Verification: Ensure web app meets the specified requirements and standards (e.g., for architecture, code, configuration).

Security Validation: Identify vulnerabilities in the web app without explicitly testing against requirements (e.g., resists attacks when exposed to potential threats).

The CMMI-DEV engineering processes, overlaid with the build steps outlined by the OWASP, are shown in Figure 3. Technologies such as protocols, platforms or programming languages will all have an effect on web application security, especially when an application is placed into the final production environment. During definition and design, a systematic walk-through of the proposed system architecture, which takes into account known vulnerabilities (such as the OWASP Top 10) and threats, should be performed to validate any security-related design decisions. These decisions should be documented for individual application components and for the integrated components in the production environment, as a reference that becomes part of the system baseline.

Figure 3. CMMI-DEV Engineering and OWASP Development Processes Aligned
DAST plays a prominent role during development. Traditionally, it is the main testing method during integration and end-to-end system testing. However, DAST can also be used to uncover vulnerabilities that result from insecure coding practices. It can work with static application security testing (SAST) to more effectively expose and remediate flaws than either testing approach alone. In a well-instrumented test environment, DAST demonstrates how various application components play together under controlled conditions, providing visibility into how the app will behave in production.

Finally, the completely integrated web app is positioned for final testing and acceptance prior to deployment into production. The entire web app should undergo vulnerability scanning as if it were deployed into production. The more robust this testing is—such as determining susceptibility or identification of potential zero-day exploits—the greater the confidence will be that the application can handle both the known knowns and the unknown unknowns.

**Maintenance and Operations: Rinse and Repeat**

Once web applications are deployed to production, vulnerability management truly becomes an ongoing process. The M&O phase becomes a series of mini-development projects triggered by routine health checks of the applications, changes to the technical structure of the deployed system, changes in security or IT operations, and new threats and/or vulnerabilities in the attack landscape.

How vulnerability management might support post-production security of web applications is shown in Figure 4. At the heart of the discovery process is vulnerability scanning, the results of which support the three key processes associated with CMMI-DEV: configuration management (CM), measurement analysis (MA) and process and product quality (PPQ).
Because security is a process, establishing an effective vulnerability management program is one as well. Initially, an organization might not have the maturity—processes, infrastructure or human resources—needed to continuously scan and analyze its web applications. But it needs to work toward that goal.

And at the heart of this program is the effective integration of vulnerability scanning. Continuous scanning helps organizations both determine their success in designing in security from the start and also in remediating flaws uncovered in production. It can help identify trends in the performance of the vulnerability management program, information that security managers and other executives can use to justify budget allocations to the organization’s board of directors.

What should the next steps be? Simply to build your approach based on the maturity levels in CMMI-DEV as adapted to vulnerability management. Advice on what maturing your vulnerability program entails and the use of vulnerability scanning is shown in Table 3.

<table>
<thead>
<tr>
<th>Level</th>
<th>Characteristics</th>
<th>Vulnerability Management Emphasis</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. Optimizing</td>
<td>Processes focused on continuous improvement</td>
<td>Information security processes are constantly improved through feedback from existing processes. Metrics are used to lead improvement and continuously reduce attack surface. Vulnerability scanning is integrated with asset and configuration management across the enterprise.</td>
</tr>
<tr>
<td>4. Quantitatively Managed</td>
<td>Processes that are measured and controlled</td>
<td>Information security processes can be measured quantitatively using metrics such as defined by CIS Measurement Companion or the OWASP. Results are used to inform organizational stakeholders for further review and action on results.</td>
</tr>
<tr>
<td>3. Defined</td>
<td>Processes that are characterized for the organization and are proactive</td>
<td>Information security processes are well defined and meet minimum standards for CIS Control families such as 4 and 18. Scans are authenticated at least weekly, maybe daily. Results inform operational processes.</td>
</tr>
<tr>
<td>2. Managed</td>
<td>Processes that are characterized for projects and are often reactive</td>
<td>Information security processes are established, defined and documented. Successful outcomes are repeatable. Vulnerability scanning is brought in-house, but only monthly or yearly. Characterized by lower-end solutions, limited budget and lack of prioritization by management.</td>
</tr>
<tr>
<td>1. Initial</td>
<td>Processes that are unpredictable, poorly controlled and reactive</td>
<td>Information security processes are unorganized and unstructured. Vulnerability scanning is done once or twice a year (compliance-based) or ad hoc (reactionary). Infosec depends on third-party services and tools. Characterized by limited exposure throughout the organization.</td>
</tr>
</tbody>
</table>

Each maturity level provides a necessary foundation for effective implementation of processes on which the next level builds. Maturity levels should not be skipped, because higher-level processes—such as CM, MA and PPQ—have less chance of success without the discipline provided by achieving success at the lower levels.

Barbara Filkins, a senior SANS analyst who holds the CISSP and SANS GSEC (Gold), GCIH (Gold), GSLC (Gold), GCCC (Gold), GCPM (Silver) and GLEG (Gold) certifications, has done extensive work in system procurement, vendor selection and vendor negotiations as a systems engineering and infrastructure design consultant. She is deeply involved with HIPAA security issues in the health and human services industry, with clients ranging from federal agencies (Department of Defense and Department of Veterans Affairs) to municipalities and commercial businesses. Barbara focuses on issues related to automation—privacy, identity theft and exposure to fraud, as well as the legal aspects of enforcing information security in today’s mobile and cloud environments.
## Upcoming SANS App Sec Training

<table>
<thead>
<tr>
<th>Course</th>
<th>Location</th>
<th>Dates</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS 2020</td>
<td>Orlando, FL</td>
<td>Apr 03, 2020 - Apr 10, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Amsterdam May 2020</td>
<td>Amsterdam, Netherlands</td>
<td>May 11, 2020 - May 18, 2020</td>
<td>CyberCon</td>
</tr>
<tr>
<td>SANS Silicon Valley - Cupertino 2020</td>
<td>Cupertino, CA</td>
<td>Jun 22, 2020 - Jun 27, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Copenhagen August 2020</td>
<td>Copenhagen, Denmark</td>
<td>Aug 24, 2020 - Aug 29, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Network Security 2020</td>
<td>Las Vegas, NV</td>
<td>Sep 20, 2020 - Sep 27, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Online</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
<tr>
<td>SANS SelfStudy</td>
<td>Books &amp; MP3s Only</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>