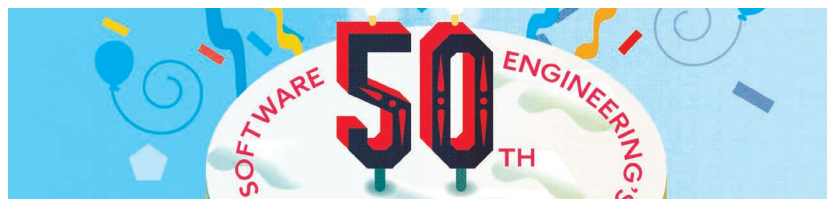


Engineering Security Vulnerability Prevention, Detection, and Response

Laurie Williams, North Carolina State University

Gary McGraw and Sammy Migues, Synopsys

// This article aims to aid software engineers, software engineering educators, and security researchers understand opportunities for education and research through an analysis of current software security practices. //



A PREPONDERANCE OF the software engineering development practices used by teams over the past 50 years guide the delivery of high-quality functionality to enable users, actors, and stakeholders to do what they want to do with a

software system. Around the turn of the 21st century, practices began to emerge to guide teams toward engineering software to stop attackers and users from utilizing unintended functionality by violating the system designer's assumptions to cause

a security breach.¹⁻⁴ Yet, breaches are reported daily in the news in all domains—from the casual to the critical. For example, in 2017 sensitive and personal data from 143 million consumers was exposed in the Equifax breach, and the WannaCry ransomware attack crippled medical institutions in the UK and other organizations around the world. The attackers will press on, unceasingly.

It's up to software engineers and security researchers to take control of the situation. We need to build software that is engineered to thwart intentional attackers and protect users from exposing data through their unintentionally insecure actions. Unfortunately, the demand for trained cybersecurity professionals far exceeds the supply; the number of unfilled cybersecurity jobs is predicted to rise to 1.8 million by 2022.⁵ Professional and university educators would benefit from understanding the most important software security practices to teach all software engineering students, particularly those desiring to focus on cybersecurity. Additionally, the need for a foundational science for cybersecurity has infiltrated security research⁶ and motivated a focus on the hardest problems in cybersecurity.⁷

The goal of this article is to aid software engineers, software engineering educators, and security researchers understand opportunities for education and research through an analysis of the software security practices currently in use by software professionals. The analysis is conducted on data on the use of a subset of 113 software security practices by 109 firms over 42 months, as reported in the *Building Security In Maturity Model (BSIMM) Version 8 (BSIMM8)* study.⁸



Software Security Practice Use

Software engineers use software security practices to

- *prevent* the introduction of vulnerabilities into a product,
- *detect* vulnerabilities that have been injected during development, and
- *respond* to the discovery of vulnerabilities in a deployed product by attackers and researchers.

These vulnerabilities are generally classified as *implementation bugs* and *design flaws*, which are considered to appear in equal proportions.³ Implementation bugs are implementation-level problems that can be more easily discovered and remedied, such as buffer overflow or SQL injection. Bugs may exist in code but never be executed, and therefore remain dormant.³ Design flaws are deeper and represent a systemic problem in the product that can be corrected only with redesign. For example, a number of classic flaws exist in error-handling and recovery systems that fail in an insecure or inefficient fashion.⁹

In the next three subsections, we discuss 55 software practices that are used to prevent, detect, and respond to both implementation bugs and design flaws. We utilize data obtained via the BSIMM8 study. The BSIMM is a multiyear empirical study of the current state of software security initiatives in industry. The BSIMM assessments are conducted through in-person interviews by software security professionals at Cigital (now Synopsys) with security leaders at a firm. Via the interviews, the firm obtains a scorecard on which of the 113 software security practices the firm uses. The 113 practices were

enumerated by Cigital professionals on the basis of the practices they had observed in their work with companies. The 58 practices not related to prevention, detection, and response deal with governance, intelligence, and deployment.

After the firm completes the interviews, they are provided information comparing themselves with the other organizations that have been assessed. BSIMM assessments have been conducted since 2008 (BSIMM1 to BSIMM8). In the BSIMM8 report in 2017, all measurements older than 42 months were excluded to ensure the relevance of the data, bringing the distinct measurements to 109 firms.

Vulnerability Prevention

The best case is when the development team is able to prevent the injection of vulnerabilities. A vulnerability that is never injected never needs to be discovered, mitigated, or responded to.

Table 1 shows the percentage of the 109 firms that use the 30 software security practices that can be used to prevent the injection of vulnerabilities. Two practices are used to prevent the injection of implementation bugs. These two practices are used, on average, by 18% of the firms. Twenty-eight practices are used to prevent the injection of design flaws. These practices are used, on average, 28% of the time. Overall, the 30 practices are used 27% of the time, with more firms using design flaw prevention practices than implementation bug prevention practices.

As observed in Table 1, only seven practices are used by more than half of the firms. The more-used practices that appear at the top of the table often deal with policies

and regulations. The least-used practices that appear at the bottom of the table often deal with the development of patterns and “top-N” lists. These results indicate that organizations may be motivated more by compliance than by systemic eradication of vulnerability types.

The software security group (SSG) in an organization can play a key role in preventing design flaws. The SSG is the internal group charged with carrying out and facilitating software security for the organization. The results in Table 1 indicate the firms could better use their SSG for vulnerability prevention.

Vulnerability Detection

Vulnerability detection practices are used to find implementation bugs and design flaws in a product prior to its deployment to a customer. Detecting vulnerabilities is less desirable than preventing them, but better than deploying the product with vulnerabilities. Detection is more reactive; prevention is more proactive.

As shown in Table 2, the 10 implementation bug practices are used, on average, by 42% of the firms. The 11 design flaw detection practices are used, on average, by 30% of the firms, indicating that more resources are focused on detecting the smaller implementation bugs. Overall, the 16 detection practices are used 36% of the time, on average, compared with the prevention practices, which are used 26% of the time. These results indicate that firms tend to be more reactive than proactive when dealing with vulnerabilities.

Highly used detection practices include the use of external (87%) and internal (62%) penetration testers to find problems. While penetration testing is a highly effective practice, the software product is late in the

Table 1. Vulnerability prevention practices.

Problem	Practice	Usage (%)
Bugs (2)	Use a top- <i>N</i> bugs list (real data preferred).	21
	Use secure coding standards.	14
	Average (bugs)	18
Flaws (28)	Build and publish security features.	78
	Translate compliance constraints to requirements.	65
	Engage a software security group (SSG) with architecture.	64
	Create a data classification scheme and inventory.	62
	Unify regulatory pressures.	61
	Create security standards.	61
	Create (security) policy.	51
	Gather and use attack intelligence.	46
	Create an SSG capability to solve difficult design problems.	38
	Identify potential attackers.	33
	Implement and track controls for compliance.	32
	Use application containers.	27
	Identify a personally-identifiable-information data inventory.	25
	Create standards for technology stacks.	23
	Identify open source in apps.	23
	Define and use an architectural-analysis process.	13
	Build and maintain a top- <i>N</i> possible attacks list.	13
	Standardize architectural descriptions (including dataflow).	11
	Require use of approved security features and frameworks.	10
	Build attack patterns and abuse cases tied to potential attackers.	8
	Create technology-specific attack patterns.	7
	Build a capacity for eradicating specific bugs from the entire code base.	5
	Form a review board to approve and maintain secure design patterns.	5
	Have a science team that develops new attack methods.	4
	Make the SSG available as an architectural-analysis resource or mentor.	2
	Have software architects lead design review efforts.	2
	Find and publish mature design patterns from the organization.	2
	Drive analysis results into standard architecture patterns.	0
Average (flaws)	28	
Average usage of all 30 practices	27	

development process when this testing is done. Conversely, code review can be done earlier in the process but has a lower usage rate at 31%.

Fuzzing tools and automated static-analysis tools are available to detect implementation bugs. Software teams would be well served by automation tools to detect design flaws. Few firms indicated the use of automation in detecting vulnerabilities. Security researchers can consider the low usage of automation an opportunity. Software development teams would be well served by accurate automation tools that can fit into their current workflow without excessive disruption.

Vulnerability Response

Six software security practices are used to detect a breach or to respond to the detection of vulnerabilities once the product is deployed. As shown in Table 3, these six practices are used, on average, by 48% of the firms. The three practices used most often deal with emergency responses and bug fixing. The lowest-used practices are focused on proactive actions, such as fixing all occurrences of bugs. The lack of use of application behavior monitoring can be a signal for the need for research in effective behavior-monitoring tools.

Recommendations

Increasingly, security breaches occur—from nation-state actions, to the disruption of federal elections, to the release of millions of personal records, to hackers talking to babies through Internet-connected baby monitors. Software security practices for building secure products have emerged only in the last 20 years of the 50-year life of software engineering.

The 10 years of data obtained in the BSIMM1 through BSIMM8

studies have shown that organizations are increasingly adopting software security practices. However, the BSIMM data indicated that firms most often adopt response practices, followed by the use of detection practices, followed by prevention practices. Software development organizations have no choice but to respond to a breach once the product is deployed, but the damage has been done. Organizations could benefit from a more proactive approach to building secure products through continued growth in the use of the prevention and detection software security practices.

Software engineers and security researchers must continue to rise to protect society from the attackers. Engineers should explicitly consider the bad actors for their systems and what these actors want to do, such that the system can stop them in their tracks using practices such as abuse cases and threat models. Engineers should also consider the unintentional mistakes that users can make, such as clicking on suspicious links, and design systems to protect the user from his or her own actions.

Over the past 10 years, research conferences have shown a growth in work at the intersection of security and software engineering. Through scientific security research programs, security researchers are making explicit the principles that underlie attackers' actions and that underlie fundamentally secure systems. The BSIMM data indicates that researchers should continue to develop tools to enable development teams to efficiently prevent and detect implementation bugs and design flaws.

Providing tools to aid in software security is not enough.¹⁰ Students and practitioners need to be trained. Educators of software engineers

Table 2. Vulnerability detection practices.*		
Problem	Practice	Usage (%)
Bugs (10)	<i>Use external penetration testers to find problems.</i>	87
	Ensure that quality assurance (QA) supports edge or boundary value condition testing.	80
	Have the SSG perform an ad hoc review.	63
	<i>Use penetration testing tools internally.</i>	62
	Use automated tools along with a manual review.	60
	Make code review mandatory for all projects.	31
	<i>Integrate black-box security tools into the QA process.</i>	23
	Perform fuzz testing customized to application APIs.	9
	<i>Include security tests in QA automation.</i>	8
	<i>Create and use automation to do what attackers will do.</i>	1
Average for bugs	42	
Flaws (11)	<i>Use external penetration testers to find problems.</i>	87
	Perform a security feature review.	83
	<i>Use penetration testing tools internally.</i>	62
	Perform a design review for high-risk applications.	28
	<i>Integrate black-box security tools into the QA process.</i>	23
	Have the SSG lead design review efforts.	22
	Use automated tools with tailored rules.	15
	<i>Include security tests in QA automation.</i>	8
	Build a factory. (Multiple analysis techniques feed into one reporting or remediation process.)	3
	Automate malicious-code detection.	3
	<i>Create and use automation to do what attackers will do.</i>	1
	Average for flaws	30
Average use of all 16 practices (duplicate practices removed)	36	

* Italics indicate practices that appear for both bugs and flaws.


should ensure that students learn the importance of and the practices for designing and developing secure systems.

Thwarting the attackers goes beyond the software engineering

practices discussed in this article. Indeed, the BSIMM contains 58 other practices related to governance; organizing, managing, and measuring a software security initiative; staff development and training; and

Table 3. Vulnerability response practices.

Practice	Usage (%)
Create or interface with incident response.	84
Track software bugs found in operations through the fix process.	76
Have an emergency code base response.	72
Use application input monitoring.	45
Use application behavior monitoring and diagnostics.	4
Fix all occurrences of software bugs found in operations.	4
Average	48

We all play a role in securing our world. 

References

1. M. Howard and D. LeBlanc, *Writing Secure Code*, Microsoft Press, 2001.
2. M. Howard and S. Lipner, *The Security Development Lifecycle*, Microsoft Press, 2006.
3. G. McGraw, *Software Security: Building Security In*, Addison-Wesley, 2006.
4. J. Viega and G. McGraw, *Building Secure Software: How to Avoid Security Problems the Right Way*, Addison-Wesley, 2001.
5. D. Kawamoto, “Cybersecurity Faces 1.8 Million Worker Shortfall by 2022,” *Dark Reading*, 7 June 2017; <https://www.darkreading.com/careers-and-people/cybersecurity-faces-18-million-worker-shortfall-by-2022/d/d-id/1329084>.
6. D. Evans and S. Stolfo, “The Science of Security,” *IEEE Security and Privacy*, vol. 9, no. 3, 2011, pp. 16–17.
7. D. Nicol et al., “Science of Security Hard Problems: A Lablet Perspective,” 27 Nov. 2012; <https://cps-vo.org/node/6394>.
8. G. McGraw, S. Miguez, and J. West, *Building Security in Maturity Model (BSIMM) Version 8*, 2017; <https://www.bsimm.com>.
9. I. Arce et al., *Avoiding the Top 10 Software Security Design Flaws*, IEEE Center for Secure Design, 2014; <https://www.computer.org/cms/CYBSI/docs/Top-10-Flaws.pdf>.
10. J. Witschey et al., “Quantifying Developers’ Adoption of Security Tools,” *Proc. 10th Joint Meeting Foundations of Software Eng.* (FSE 15) 2015, pp. 260–271.

ABOUT THE AUTHORS



LAURIE WILLIAMS is the interim department head and a professor of computer science at North Carolina State University. She’s also the codirector of a US National Security Agency–sponsored Science of Security lablet. Her research focuses on software security; agile software development practices and processes, particularly continuous deployment; and software reliability, software testing, and analysis. Williams received a PhD in computer science from the University of Utah. She’s an IEEE Fellow. Contact her at laurie_williams@ncsu.edu.



GARY MCGRAW is the Vice President Security Technology at Synopsys. He also produces the monthly Silver Bullet Security Podcast for Synopsys and *IEEE Security & Privacy*. McGraw received a dual PhD in cognitive science and computer science from Indiana University. Contact him at gem@synopsys.com.



SAMMY MIGUES is the principal scientist at Synopsys. He currently focuses on management consulting that helps executives build holistic and realistic security programs. Miguez received a master’s in information security from Eastern Michigan University. Contact him at sammy.miguez@gmail.com.

the collection of corporate knowledge used in carrying out software security activities throughout the

organization. Management and product owners must evaluate and see the value in developing secure products.