the method that enables security teams to do more with less.

By reducing the day-to-day workload of team members through improved intelligence and reporting, streamlined workflows and playbooks for automated response actions, SOAR can enable those skilled cyber security professionals to put their talents and knowledge to better use. Empowered by accurate, relevant metrics that quantify the efforts of the SOC, SOAR not only highlights areas of improvement or shortfalls, but creates a compelling business case for greater investment within the security team.

About the author

As a vice president and managing director of EMEA, Ross Brewer has nearly 30 years of sales and management experience, with more than 20 years in the information security sector. At LogRhythm he leads the EMEA team and has been key in helping deliver consistent, rapid growth in the region. He is regularly quoted in national and trade press, including the BBC, The Times, Infosecurity Magazine and The Register. He is also frequently featured as a cyber security expert on TV and radio, with appearances on BBC News and BBC Radio to discuss breaking news and provide expert insight into cyber security incidents. Before joining LogRhythm, Brewer was a senior executive at LogLogic, where he served as vice president and managing director of EMEA. He's also held key leadership roles in Europe and the South-Pacific region at NetIQ, PentaSafe and Symantec.

References

1. 'Cyberthreat Defense Report'. CyberEdge, 2018. Accessed Sep 2019. https://cyber-edge.com/wp-content/uploads/2018/03/CyberEdge-2018-CDR.pdf.
2. 'The Role of Regulations in Enterprise Cyber security'. Aberdeen Group, 22 Mar 2017. Accessed Sep 2019. www.aberdeen.com/techpro-essentials/role-regulations-enterprise-cyber security/.
3. 'The Cost of Cloud Expertise Report'. Rackspace, 2017. Accessed Sep 2019. www.rackspace.com/resources/cost-cloud-expertise-report.

A source code perspective framework to produce secure web applications

Alka Agrawal, BBA University; Mamdouh Alenezi, Prince Sultan University; Rajeev Kumar, BBA University; and Raees Ahmad Khan, BBA University

Hackers and other cyber attackers remain fearless concerning the mitigation mechanisms that have evolved for addressing security over the past few years. Cyber attacks are on the rise and countless security breaches take place daily. It is believed that cybercrime in its various forms will cost the world $6tr per year by 2021.[1] It has become essential that software companies evaluate their businesses to identify application security needs, strategies and weaknesses. Establishing a security policy to safeguard their software applications has become an urgent need.


Alka Agrawal


Mamdouh Alenezi


Raees Ahmad Khan


Rajeev Kumar

Software companies need to make sure that their developers write more secure code in the first place. Executives should prioritise writing secure code up front. And the organisation should revise its lifecycle approach in order to include security professionals in the loop just after the project requirements get determined.

Vulnerable applications are the bane of the software industry. The only boon is to shift security from being exclusive to being inclusive. By integrating security teams within development teams, software companies will get earlier feedback on the security of their software or applications, thus reducing the costs associated with implementing these fixes. The budget to remove defects, including security flaws, can be hundreds of times higher after deployment. Therefore, there is a need to shift security from reactive to proactive, supported by appropriate techniques. Shifting security to the left will help to achieve the goal of releasing secure web applications. There is a need to embed security into the workflow.

Many existing secure coding practices are actually more focused on ethical hacking and penetration testing that is separate from software development. Developers do not have the time to learn

a whole new technical field. To even approximate bulletproof source code, analysts, designers, programmers and project leaders need to imagine everything that could go wrong with every aspect of the source code. However, while it is impractical to forecast all the curveballs the hacker will throw, developers can make an effort to minimise the attack surface, plug holes and prepare for the side effects of a probable breach.

"There is a need to shift security from reactive to proactive, supported by appropriate techniques. Shifting security to the left will help to achieve the goal of releasing secure web applications"

Given the rise in security breaches, software companies should consider development as synonymous with secure software development. Designing for security and the use of secure practices and standards does not guarantee security, but it helps in delivering secure software. Many vulnerabilities stem from relatively negligible coding errors and many research findings have shown that the majority of vulnerabilities are related to programming errors that are fairly well understood.

Why secure coding?

As the world becomes more interconnected through the use of faster and larger digital networks, the software industry is continuously trying to enhance mechanisms to protect applications against cyber attacks. Recently, Facebook discovered a cyber attack where attackers exploited a vulnerability in its code that might have potentially impacted 50 million accounts.

Despite many innovative and advanced cyber security mitigation techniques, everyday hackers and attackers are finding ways to push data breach statistics to a new height. According to a report by E&T editorial staff on 4 January 2019, hundreds of high-profile German politicians and other public figures have been affected by a major data breach, as a result of which their personal data were published online. The leaked data included personal phone numbers, email addresses, work correspondence, family conversations, holiday photos, photos of ID cards, bank account information and copies of identity cards.[2,3]

In another example, a criminal attack affecting bookings made on British Airways' website and app resulted in financial and personal data being stolen from potentially hundreds of thousands of customers in 2018.[4] In the same year, online fashion store Shein announced a security breach that affected around 6.42 million of its customers by compromising their email addresses and encrypted passwords for online store accounts. This year has seen an inordinate number of cyber security meltdowns.

"There have always been several schools of thought about the need to securely code, but consistent, tangible ways have not yet been clearly established"

Many of 2017's data breaches occurred at the hands of cyber criminals who leveraged security issues with data storage, misconfigured security settings, and the overall lack of security solution in place to protect data.[5,6] Data breaches in 2016 were making national headlines every other week with two of the largest breaches in history, including a massive hack at the Democratic National Committee as well as breaches in healthcare, point-of-sale systems and the federal sector. In the same year, Yahoo! captured the record for the largest breach, in September 2016 when 500 million customer records were exposed – and then went on to break its own record by revealing that the true figure was double that amount.[7,8] In the first half of 2015, a total of 880 data breaches were recorded, which was a 10% increase on 2014's record. The biggest data breach of 2015 was against Anthem, the health insurer, in which the data of more than 80 million people was compromised.[9]

Code reviews

Initially, the solution to securing source code was simply the use of code reviews. But a code review is a process without a specific deliverable to a customer, and it often becomes a collaborative effort without a leader or an owner. The process of code review finds bugs, rather than finding security flaws. There have always been several schools of thought about the need to securely code, but consistent, tangible ways have not yet been clearly established.[10,11] Secure coding should be as important to software industry culture as it is to the overall software development process. The only aim of this cultural shift is to instil security practices so intensive that they become second nature. Poor code equals insecure code. During the development process, insecure coding practices that stem from behaviours and bad habits might lead to vulnerabilities in the source code.

Source code is a sort of digital genome that defines the properties of software and elaborates how it functions. Compromised sensitive data is a consequence of a security failure. Often, sensitive data is compromised through vulnerable source code. Addressing source code security has become exceedingly unmanageable. It has been revealed from various studies that 20% of vulnerabilities pose 80% of the risk for source code. There will be a debate about the actual causes of such a severe security breach. The tremendous rise in the number of software vulnerabilities has been exploited in recent years.

A significant number of vulnerabilities have been traced back to coding errors. Source code vulnerability is highly technical. The US Department of Homeland security noted that 90% of security breaches happen because of vulnerabilities in the code. More importantly, a time has come when the

software industry should think about having security heavily integrated into its core culture.

Bad and insecure coding practice remains prevalent to this day. Software industries should take charge and prioritise security in the development process. By eradicating bad code, the industry will not only help software developers do a better job, but also potentially secure their reputation, data and ongoing survival.

Where is the gap?

The adoption of web application security solutions by the software industry has left much to be desired. For intruders, web application attacks have become one of the most frequent and successful patterns in confirmed breaches. Software companies are spending billions of dollars on securing the network, perimeter and hardware. In today's race to build cutting-edge business solutions, the inclusion of a fast development cycle using third-party software or open source software introduces a new layer of risk that needs to be addressed immediately to secure the application from a data breach. There has never been a better or more necessary time to invest in protecting web applications than right now. Therefore, the highest security standards should be the key highlight of the web application development process in any software company.

"Source code vulnerability is highly technical. The US Department of Homeland Security noted that 90% of security breaches happen because of vulnerabilities in the code"

Application-level security is increasingly coming under fire. Software firms are making efforts with regards to security breach mitigation. But in spite of investing huge budgets for securing applications, data continues to be compromised at an alarming rate. Recurrently, sensitive data is compromised through insecure source code. With data breaches, the discussion begins with the stolen data and industry efforts to cover up the breach but nobody talks about the reason for the breach. The reason is very simple. If the company that suffered the breach admits that it lacked important security features, which is why the breach occurred, it might lose customers. Instead of discussing what has happened, there is an urgent need to demonstrate what is wrong with the application software and efforts should be made to provide a solution. This allows a positive feedback loop between security and developers, demonstrated by clear recommendations to improve application security.

"The basic problem with the software industry is that secure development practices have not been accepted yet as a revenue-generating function. This is one of the strongest reasons why software firms don't bother to train their developers to write secure code"

In the literature, several security initiatives and methodologies have been proposed to support the integration of security with the development lifecycle.[12-16] But, unfortunately, vulnerabilities persist.[17] The reasons cited in the literature are conflicting. The persistence of vulnerabilities in software applications might be because of the lack of proper security guidelines or the ignorance of these guidelines by software companies. Another school of thought believes that developers lack knowledge or they might lack the ability or proper expertise to identify vulnerabilities despite having security knowledge.[18-21] Tools are another pain point for developers who want to write secure code.

Protecting software applications from theft and attack has been a time-proven practice. The basic problem with the software industry is that secure development practices have not been accepted yet as a revenue-generating function. This is one of the strongest reasons why software firms don't bother to train their developers to write secure code. As a result, software firms do not allocate budget for security technologies until after a successful attack. Unfortunately, fixing bugs and flaws post-attack is very expensive and reputations have already been damaged.

Recent developments

A review of research journals revealed few articles covering the area of writing more secure web applications. Many books, journals, research articles and online materials cover allied topics, but not this exact one. The works cited in the literature span from the simplest web page on the topic to full courses. The most specific problem during a review of the work on the topic was to navigate the vast size of the fragmentary information and to find what is relevant to the subject of writing secure source code, as the material does not use a uniform approach. Some of the most relevant work and influences cited in the literature are as follows.

Nunes et al in 2018 proposed a benchmark for assessing and comparing static analysis tools in terms of their capability to detect security vulnerabilities.[22] The benchmark proposed was implemented and assessed experimentally using a set of 134 WordPress plugins. The authors advocated the classification of vulnerabilities. In the same year, Smith et al carried out a study on the defect resolution process to build better security tools and subsequently help developers resolve defects more accurately and efficiently.[23] The authors reported on an exploratory study with novice and experienced software developers, equipping them with a security-oriented static analysis tool.

In 2017, Awan et al proposed a security framework that identifies vulnerabilities and observes the traffic between the browser and the

server.[24] This also takes control of the request and its response. The authors implemented advanced discovery and fuzzing technologies to discover the vulnerability. The objective of the framework was to enhance the security of important national ID databases. In the same year, Holík and Neradova presented a process of penetration testing of web applications.[25] Their goal was to detect application flaws and vulnerabilities and to propose a solution to mitigate them. The authors analysed current penetration testing tools and subsequently tested them on a use case web application, build specifically with current security flaws.

"Several security tools and suites of tools are available today to assist the design and development of secure code. However, no formal evaluation of any of these tools has been undertaken"

In 2016, Alenezi and Yasir tested several open source web applications against common security vulnerabilities categorised as 'dodgy code vulnerabilities', 'malicious code vulnerabilities' and 'security code vulnerabilities' on seven different web applications built in Java.[26] The results obtained revealed the fact that hasty programming or lack of developer knowledge concerning security causes major vulnerabilities in source code. In the same year, Alenezi and Yasir worked on educating developers and helping them to produce more secure code, and they proposed a framework that can be integrated into any development environment.[27]

An exhaustive review of recent developments reveals that, over the years, security experts' efforts have been invested in specific methodologies and techniques for delivering secure software. Several security tools and suites of tools are available today to assist the design and development of secure code. However, no formal evaluation of any of these tools has been undertaken. A dedicated process in the form of a framework for writing secure code with support in order to meet security requirements is urgently and genuinely required.

The framework

It is a well-established fact that source code will always have vulnerabilities, irrespective of time, effort and the techniques used to develop a secure software application. But it is always possible to evolve a mechanism that, when followed, will minimise the overall vulnerabilities in the source code and make those that remain harder to exploit. Writing secure code is challenging and more demanding because a large proportion of security incidents result from flaws in the source code. There is still room for the software industry and state of the art to evolve to provide a better standard mechanism that can be applied across software organisations more uniformly.

To encourage developers to write secure source code, there is a need to integrate the whole process of scanning, detecting and mitigating security vulnerabilities and flaws during source code analysis. Taking into account the need and significance of a roadmap or framework for developing secure source code with essential and desirable security features, an integrated and prescriptive framework is hereby proposed. We have attempted to make the proposed framework highly implementable and prescriptive in nature.

The development process for secure source code is comprised of three phases together with prescriptive steps for each. Such a framework has been proposed on the basis of integral and basic components for writing secure source code for a web application. Figure 1 shows a general overview of the framework. The first phase starts with 'execute and monitor' to produce the vulnerabilities and flaws data repository. The classification of vulnerabilities and flaws and the prioritisation of identified vulnerabilities is treated as an important task and has been put forth as a second phase, 'classify and control'. In the third phase of 'refine and manage', all data repositories of source code will be merged into a single repository, and a suggested measure in the form of prioritised secure source code writing guidelines is produced for ready reference by web application developers. An attempt has been made to symbolically represent the concept of writing secure source code and make the framework prescriptive in nature followed by a brief description of each of the phases comprising the depicted steps.

Phase I: Execute and monitor

This phase starts with scanning the source code using analysers. A dataflow analyser detects the flow of malicious data. The semantic analyser searches vulnerable functions used in the source code. The control flow analyser tracks the sequence of operations to detect

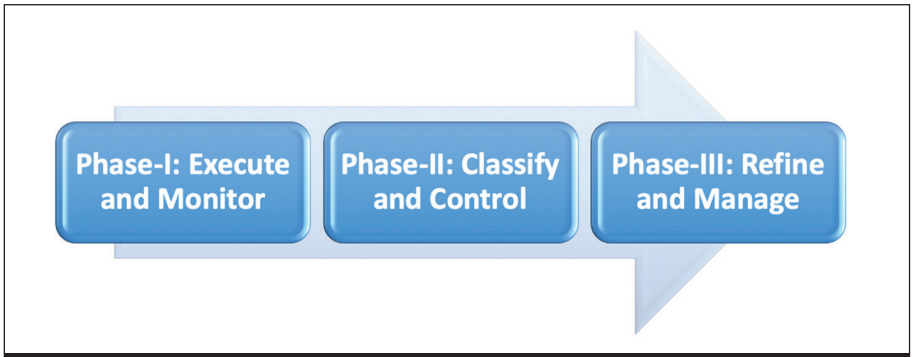

Figure 1: A general overview of the proposed framework.

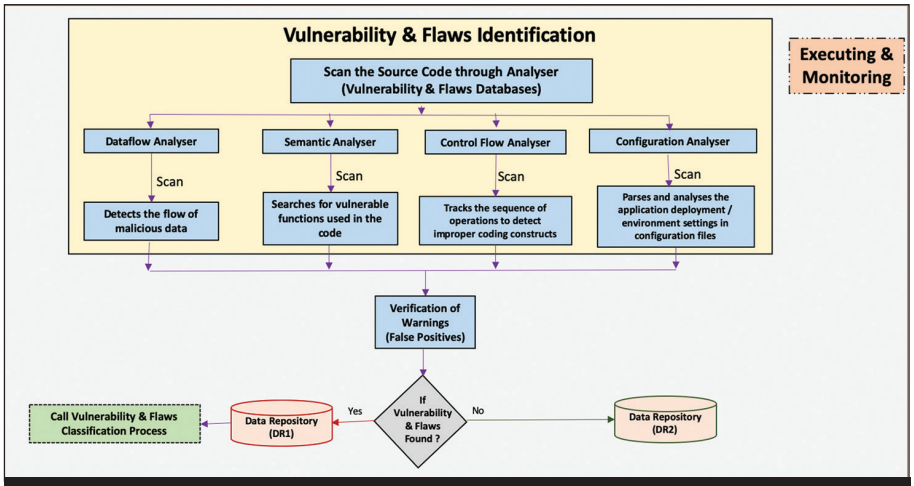

Figure 2: Phase I – execute and monitor.

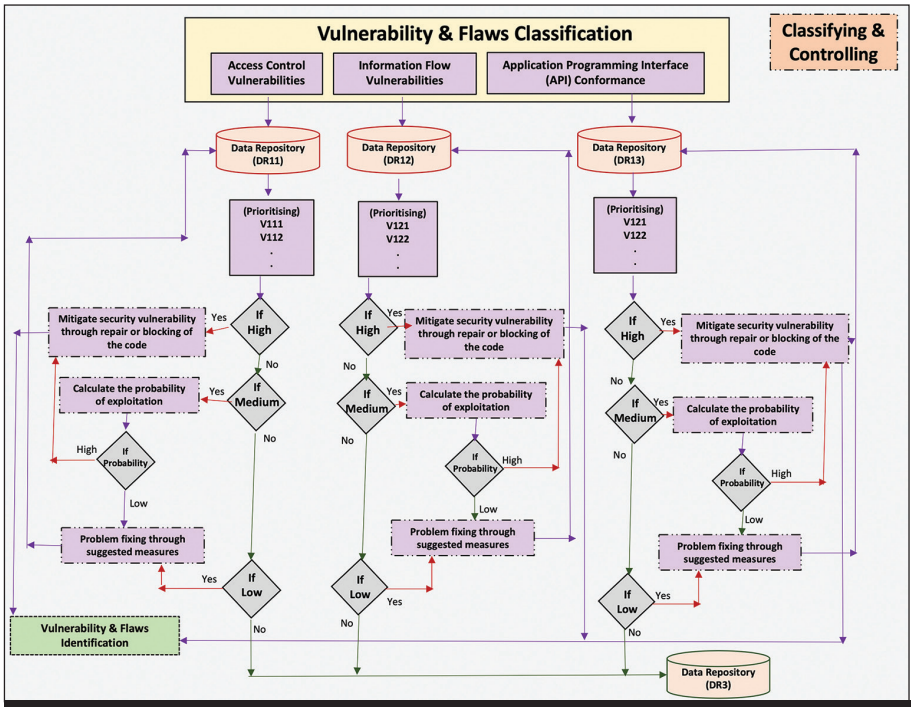

Figure 3: Phase II – classify and control.

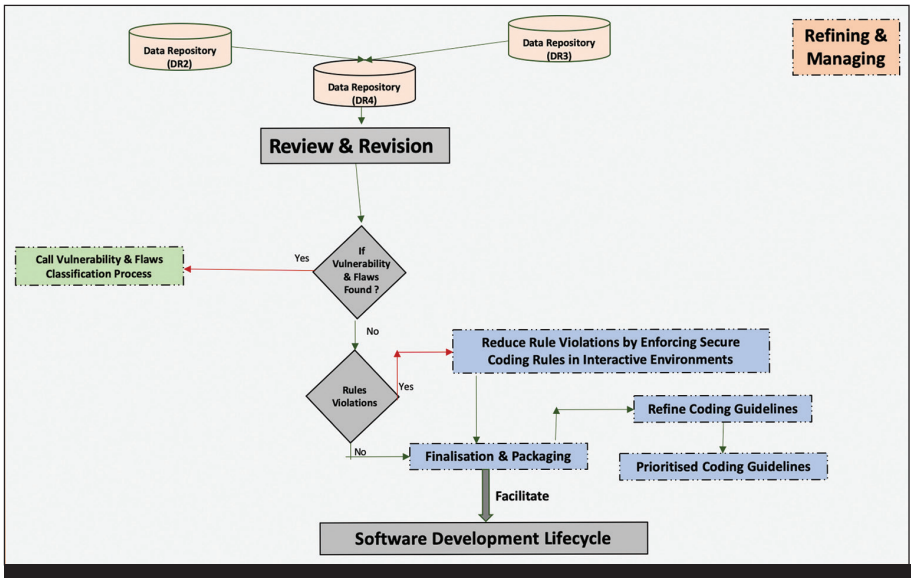

Figure 4: Phase III – refine and manage.

improper coding constructs. Finally, the configuration analyser parses and analyses the application deployment. Practitioners will verify identified vulnerabilities and flaws. Identified blacklist code and whitelist code will be documented. Prescriptive steps involved in executing and monitoring the source code are shown in Figure 2.

Phase II: Classify and control

This phase classifies the identified security vulnerabilities and flaws into three categories – namely access control vulnerabilities, information flow vulnerabilities and application programming interface (API) conformance. Now, these classified vulnerabilities will be prioritised according to their severity levels to reduce the cost and time during mitigation as per the priority list.

Prioritised vulnerabilities will be checked with three indexes – high, medium or low. In order to mitigate vulnerabilities bearing a high severity level, the code will be repaired or blocked. Vulnerabilities with a medium severity level will be passed through the procedure to calculate the probability of exploitation. If the probability is high, it will be treated as a vulnerability with a high severity level and will be mitigated accordingly. If the probability of exploitation is low, it comes under the category of vulnerabilities with low severity and will be addressed by fixing through suggested measures. Finally, an analysis summary report will be prepared summarising the actions associated with the source code. The prescriptive steps in classifying and controlling the source code analysis process are shown in Figure 3.

Phase III: Refine and manage

After successful implementation of Phase II, all the repositories of source code will be merged into a single

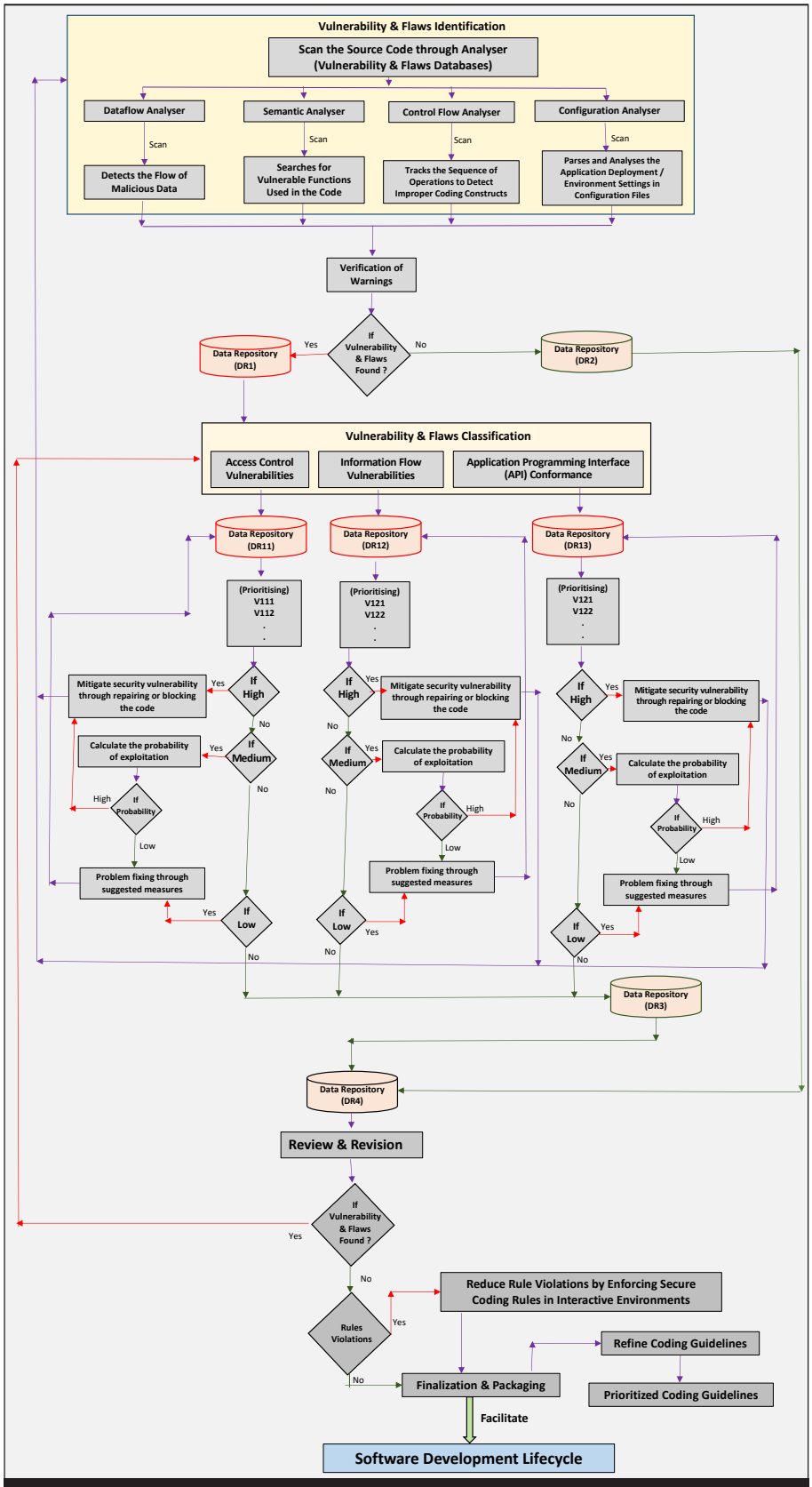

Figure 5: A framework to secure web application source code.

repository. Again, source code will be analysed manually. Identified logical errors and flaws will be mitigated through the suggested measures provided. Further, rule violations will be identified and reduced by enforcing secure coding rules in interactive environments. Prescriptive steps in refining and managing the source code analysis process are shown in Figure 4.

Once the source code analysis process meets the exit criteria based on time, cost and objectives, source code analysis will be finalised and will start facilitating a secure software development lifecycle. An integrated and prescriptive framework for securing web application source code is shown in Figure 5.

Significance of the framework

One of the most pertinent approaches for delivering secure code is vulnerability scanning of source code. This practical approach is currently adapted by most security practitioners. In essence, the integration of security strategies as a security framework while writing the source code would allow any security anomalies to be detected and fixed well before the software application is released. The framework will also allow the code to be audited for conformance which, as a result, will not only provide greater security but will also save time, cost and resources that might be incurred on redevelopment or patching of the software application once it is released.

"Experimental deployments and statistical analyses on a large scale with typical representative samples may be needed to standardise the framework"

The objective of the framework proposed is to be relentlessly practical. The framework will not only enable the developer to make the source code more secure but also to make the code more robust and reliable. We are confident that implementing all the three phases of the proposed framework will ensure commercial and public trust in the secure web application development process, reducing time, costs and effort.

Implementation of the conceptual framework proposed will help security experts and programmers identify flaws

in source code and find the best mitigation approach. In the absence of any other framework, it may be used by web application source code developers across the community and, if it becomes a standard, may be improved in terms of security. Further, experimental deployments and statistical analyses on a large scale with typical representative samples may be needed to standardise the framework. A close look at the components constructing the theoretical framework related to secure code writing led to the following observations:

- The framework proposed avoids writing secure code with a subjective rating such as 'very low', 'low', 'average', 'high', 'very high' etc.
- The framework helps to evaluate secure source code and provides cost estimates of writing secure code, which facilitates the estimation and planning of new activities.
- The framework will be able to identify faulty and vulnerable code early to decrease the amount of reworking.
- Viable experiments should be designed to validate the proposed framework.
- Pre-deployments and deployments should be conducted on the proposed framework and the results gained from these uses should be analysed and interpreted.
- Informal reviews and revisions should be carried out throughout entire phases of the secure development process.

Conclusion

This article proposes a three-step framework to produce secure web applications. The framework helps in identifying the types of security vulnerabilities that arise due to programmers' mistakes. It also finds the reason for the occurrence of these mistakes. Successful implementation of the proposed framework will identify and mitigate the vulnerabilities in source code and give suggested methods for writing secure code. In future work, the implemented framework will be empirically validated through the implementation of different web application projects.

About the authors

Dr Alka Agrawal earned her doctoral degree from Babasaheb Bhimrao Ambedkar University, Lucknow, India and she is currently working as an assistant professor in the same department. She is a passionate researcher and has also published a number of research papers in national and international journals. She has research and teaching experience of more than eight years. Her areas of research include software security and software vulnerability. She is currently working in the fields of big data security and genetic algorithms.

Dr Mamdouh Alenezi is currently dean of educational services and chief information & technology officer (CITO) at Prince Sultan University, Riyadh, Saudi Arabia. He received his MS and PhD degrees from DePaul University and North Dakota State University in 2011 and 2014, respectively. He has extensive experience in data mining and machine learning, where he applied data mining techniques to solve several software engineering problems. He conducted research in several areas and worked on the development of predictive models using machine learning to predict fault-prone classes, comprehend source code and predict the appropriate developer to be assigned to a new bug.

Dr Rajeev Kumar obtained his PhD in information technology at Babasaheb Bhimrao Ambedkar University, Lucknow and he completed his Master's in information technology from the same university in 2014. He has worked on a full-time major project funded by the University Grants Commission, New Delhi, India. He has also published and presented papers in refereed journals and conferences. His research interests are software security, software durability and security risk.

Prof Raees Ahmad Khan is currently working as a professor and head of department in the Department of Information Technology and as dean of the school for information science and technology at Babasaheb Bhimrao Ambedkar University, Lucknow. Khan has more than 18 years of teaching and research experience. His areas of interest are software security, software quality and software testing. He has published a number of national and international books (including Chinese language), technical articles, research papers, reviews and chapters on these topics.

Acknowledgment

The authors are thankful to the College of Computer and Information Sciences, Prince Sultan University, for providing the funding to carry out this work.

Resources

- Koussa, Sherif. 'Why don't developers write more secure code?'. Software Secured, 20 Jun 2016. Accessed Jan 2019. www.software-secured.com/why-dont-developers-write-more-secure-code/.
- Kanat-Alexander, Max. 'Why Programmers Suck'. Code Simplicity, 1 Dec 2009. Accessed Jan 2019. www.codesimplicity.com/post/why-programmers-suck/.

References

1. Kobek, Luisa Parraguez. 'The State of Cyber security in Mexico: An Overview'. Wilson Centre's Mexico Institute, Jan 2017. Accessed Jan 2019. www.wilsoncenter.org/sites/default/files/cybersecurity_in_mexico_an_overview.pdf.
2. 'Top German politicians affected in major data breach'. The Institution of Engineering & Technology, 4 Jan 2019. Accessed Jan 2019. https://eandt.theiet.org/content/articles/2019/01/top-german-politicians-affected-in-major-data-breach/.
3. 'The most infamous data breaches'. Techworld, 4 Jan 2019. Accessed Jan 2019. www.techworld.com/security/uks-most-infamous-data-breaches-3604586/.

4. Venkat Raman, Rama. 'Data stolen from hundreds of thousands of British Airways customers in major breach'. Business Insider, 7 Sep 2018. Accessed Jan 2019. www.businessinsider.com/british-airways-customer-data-stolen-2018-9?IR=T.
5. Hay Newman, Lily. 'The biggest cyber security disasters of 2017 so far'. Wired, 7 Jan 2017. Accessed Dec 2018. www.wired.com/story/2017-biggest-hacks-so-far/.
6. Rubens, Arden. 'Recap: The biggest data breaches of 2017'. Checkmarx, 31 Dec 2017. Accessed Jan 2019. www.checkmarx.com/2017/12/31/recap-biggest-data-breaches-2017/.
7. Kuranda, Sarah. 'The 10 biggest data breaches of 2016'. CRN, 28 Dec 2016. Accessed Jan 2019. www.crn.com/slide-shows/security/300083246/the-10-biggest-data-breaches-of-2016.htm.
8. Fischer, Thomas. 'The biggest and most impactful data breaches of 2016'. Data Insider, 19 Jan 2017. Accessed Jan 2019. https://digitalguardian.com/blog/biggest-and-most-impactful-data-breaches-2016.
9. Bihary, Chris. 'Cyber security year in review: major data breaches of 2015'. TAP into Technology, 15 Dec 2015. Accessed Dec 2018. www.garlandtechnology.com/blog/cyber-security-year-in-review-major-data-breaches-of-2015.
10. Grover, Mark; Cummings, Jeff; Janicki, Thomas. 'Moving beyond coding: why secure coding should be implemented'. Journal of Information Systems Applied Research, Vol.9, Issue 1, pp.38-46, April 2016.
11. Hentzen, Shil. 'The Software Developer's Guide'. Whitefish Bay: Hentzenwrke Publications, 2002.
12. Assal, Hala; Chiasson, Sonia; Biddle, Robert. 'Cesar: Visual representation of source code vulnerabilities'. In IEEE Symposium on Visualization for Cyber Security, 2016.
13. Backes, Michael; Rieck, Konrad; Skoruppa, Malte; Stock, Ben; Yamaguchi, Fabian. 'Efficient and flexible discovery of PHP application vulnerabilities'. In IEEE European Symposium on Security and Privacy, 2017.
14. Chess, Brian; McGraw, Gary. 'Static Analysis for Security'. IEEE Security & Privacy, Vol.2, Issue 6, pp.76-79, 2004.
15. Antunes, Nuno; Vieira, Marco. 'Defending against web application vulnerabilities'. Computer, Vol.45, Issue 2, pp.66-72, 2012.
16. Smith, Justin; Johnson, Brittany; Murphy-Hill, Emerson; Chu, Bill; Richter Lipford, Heather. 'Questions developers ask while diagnosing potential security vulnerabilities with static analysis'. In Joint Meeting on Foundations of Software Engineering. ACM, 2015.
17. Oliveira, Daniela; Rosenthal, Marissa; Morin, Nicole; Yeh, Kuo-Chuan; Cappos, Justin; Zhuang, Yanyan. 'It's the psychology stupid: how heuristics explain software vulnerabilities and how priming can illuminate developers' blind spots'. In 30th Annual Computer Security Applications Conference, pp.1-10, 2014.
18. Assal, Hala; Chiasson, Sonia. 'Motivations and amotivations for software security', Usenix Symposium on Usable Privacy and Security (SOUPS) 2018. August 12-14, 2018, Baltimore, MD, US.
19. Green, Matthew; Smith, Matthew. 'Developers are not the enemy!: The need for usable security APIs'. IEEE Security Privacy, Vol.14, Issue 5, pp.1-9, 2016.
20. Greenberg, Andy. 'Hackers remotely kill a Jeep on the highway – with me in it'. Wired, 21 Jul 2015. Accessed Jan 2019. www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.
21. Grieco, Gustavo; Grinblat, Guillermo Luis; Uzal, Lucas; Rawat, Sanjay; Feist, Josselin; Mounier, Laurent. 'Toward large-scale vulnerability discovery using machine learning'. In ACM Conference on Data and Application Security and Privacy, 2016.
22. Nunes, Paulo; Medeiros, Ibéria; Fonseca, José C.; Neves, Nuno; Correia, Miguel; Vieira, Marco. 'Benchmarking static analysis tools for web security'. IEEE Transactions on Reliability, Vol.67, Issue 3, pp.1159-1175, 2018.
23. Smith, Justin; Johnson, Brittany; Murphy-Hill, Emerson; Chu, Bei-Tseng; Richter, Heather. 'How developers diagnose potential security vulnerabilities with a static analysis tool'. IEEE Transactions on Software Engineering, Early Access, 2018.
24. Awan, Jawad Hussain; Memon, Shahzad; Khan, Shariq Mahmood; Usman, Muhammad; Khan, Rahat Ali; Abbasi, Shazia; Noonari, Abdul Qudoos; Hussain, Zahoor. 'A user friendly security framework for the protection of confidential information'. International Journal of Computer Science and Network Security, Vol.17, Issue 4, pp.215-223, 2017.
25. Holík, Filip; Neradova, Sona. 'Vulnerabilities of modern web applications'. 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2017.
26. Alenezi, Mamdouh; Yasir, Javed. 'Open source web application security: a static analysis approach'. Engineering & MIS (ICEMIS), International Conference, IEEE, 2016.
27. Alenezi, Mamdouh; Yasir, Javed. 'Developer companion: a framework to produce secure web applications'. International Journal of Computer Science and Information Security, 2016.