Measuring Application Security

Christopher Horn[1], Anita D'Amico[2]

[1] Secure Decisions, Clifton Park, NY, USA
chris.horn@securedecisions.com
[2] Code Dx, Northport, NY, USA
anita.damico@codedx.com

Abstract. We report on a qualitative study of application security (AppSec) program management. We sought to establish the boundaries used to define program scope, the goals of AppSec practitioners, and the metrics and tools used to measure performance. We find that the overarching goal of AppSec groups is to ensure the security of software systems; this is a process of risk management. AppSec boundaries varied, but almost always excluded infrastructure-level system components. Seven top-level questions guide practitioner efforts; those receiving the most attention are *Where are the application vulnerabilities in my software?*, *Where are my blind spots?*, *How do I communicate & demonstrate AppSec's value to my management?*, and *Are we getting better at building in security over time?*. Many metrics are used to successfully answer these questions, but one challenge stood out: there is no good way to measure AppSec risk. No one metric system dominated observed usage.

Keywords: Application security · Security management · Security metrics · Program management · Risk management

1 Introduction

Cybersecurity is receiving more attention than ever. With the ubiquity of computers, the list of possible harms that can result from malicious attacks on computers and computer networks is endless. Many, some say most, cyber incidents can be traced to exploitation of software vulnerabilities. The Software Engineering Institute estimates that 90% of cyber incidents are traceable to the attacker's exploitation of software defects [1]. The 2016 Verizon Data Breach report indicates that 40% of all data breaches were achieved using a web application attack [2]. Reducing software application vulnerabilities is a primary lever by which most organizations can reduce their risk exposure.

Application security is a specialty within cybersecurity that's focused on improving the security of application software. It focuses on understanding the threats posed to applications, the vulnerability to those threats, and the means of mitigating the resulting risk.

Assuring application security is much more than a technology problem – it requires coordinating the actions of numerous people, which means organization and process. Roles and responsibilities must be defined; budgets must be approved; people need to be hired, educated, and enabled to develop skills; culture needs to be created; tools need to be selected and acquired; and policies and processes must be defined [3–6]. Organizations orchestrate these activities under application security programs. Typically formed under a Chief Information Security Officer (CISO), these programs are directly staffed with a relatively small number of people (median of 4.3 per 1,000 developers [7]) who are charged with meeting the goals of the program.

Understanding the problem space that application security (AppSec) practitioners face and documenting and disseminating the methods that they have developed for measuring and tracking the effectiveness of their programs is a key part of maturing and refining AppSec practices. There is a significant amount of thought-intensive work required to discover effective metrics for assessing the performance of an AppSec program [8, 9].

To better understand how AppSec programs are managed, we set out to explore:

- The boundaries used to define the scope of an application security program
- The goals of the people responsible for assuring application security
- What information and metrics are being used to guide decision-making
- The tools being used to measure and track AppSec metrics

2 Methodology

In 2017, Secure Decisions investigated how people measure the progress and success of their application security tools and programs. We employed a two-pronged approach, combining a literature review with interviews of application security practitioners. This approach allowed us to sample multiple perspectives on security and management.

2.1 Literature Review

We identified and read over 75 research papers, technical reports, books, magazine articles, blog posts, and presentations during our investigation. Our primary line of inquiry was into metrics related to security and risk, including technical debt, vulnerability, security requirements and controls, threat probability, and process compliance. We also sought out information on security investment decision-making, including return on security investment and risk quantification.

2.2 Interviews

We interviewed 13 people who work in application security roles at both commercial and government organizations. The commercial organizations included healthcare insurers, software producers, and military/defense contractors. The government organizations were mostly federal-level independent verification & validation groups, but also included an IT group in a state agency with responsibility for the security verification of custom-developed application software.

Interviews were conducted over a voice & screen share Web conference using a semi-structured format. Interviews typically lasted for one hour.

Each interview was comprised of four major sections. First, we began with a standard introduction in which we introduced ourselves and our research objectives, and collected some basic, demographic-type information. Second, we presented several PowerPoint slides and asked for the level of agreement with each; the slides covered a description of a “metric”, the root goal of application security, and the technical scope of application security. By this point, interviewees were fully engaged as we began the third section – the bulk of the interview. We asked about their management goals and any metrics that they use to measure performance against these goals. For each metric, we probed about the audience, the frequency with which information is tabulated/reported, and the technical means by which the data are collected. Finally, in the fourth section, we reviewed a pre-formulated list of “management purposes” to prompt for forgotten metrics and affirm that we accurately captured management priorities. During this final section, we also gave interviewees an open-ended prompt to talk about anything on their mind and ask us questions.

We selected the semi-structured interview format because it is an effective means of exploring complicated research questions [10], enabling researchers to uncover new and unexpected findings that would not be possible using a structured survey instrument. The format allowed us to obtain detailed contextual information about each interviewee’s role and their organization’s mission and structure, to better understand their priorities and behaviors.

3 Observations

3.1 Types of Organizations

During the interviews, we noticed that practitioners’ focus varied based on the type of organization in which they worked. We saw two broad types of AppSec organization:

1. External reviewer
   An independent verification group, most commonly a legally separate third-party.
2. Internal department
   An application security group operating as a peer group in its parent organization.

While we observed that the primary goal of both types of organization is to ensure the security of software systems, we noted a subtle difference in the approach to that goal. External reviewers were more verification driven (that is, measuring compliance with formally-defined requirements) whereas internal departments considered a broader range of means to achieving security. As most of the external reviewer organizations were governmental, we believe that this reality is driven by the bureaucratic structures that drive U.S. federal procurement and mandate an external reviewer role.

A related distinction between these two types of organization was the different level of influence that each has on the product development process. External reviewers have less influence on things like developer training, architecture, and requirements that were often cited as levers of control by practitioners in internal departments.

3.2 Personnel Roles

While there were almost as many job titles as people in our interviews, practitioners in our sample generally fell into one of two broad roles: director and analyst.

In an internal department, the director role is broadly responsible for the application security program. This person champions secure development practices with the software development organization, establishes the structure, roles, and responsibilities of their team, defines testing policies and processes, selects testing tools, hires analysts, and manages the departmental budget.

In an external review organization, the director role has similar management responsibilities, but typically does not have a software development organization with which to champion secure development practices.

At the lower level, analysts are people who work directly with application security testing tools, screen findings for review with development teams, and serve as security subject matter experts who answer questions that arise during design and development.

Within the internal departments that we spoke with, AppSec analysts were almost exclusively centrally deployed to review applications in a "service bureau" model. By the same token, almost every organization was working to increase the security literacy of developers and develop at least one strong security champion who is embedded in each development team. Providing security input during the early, architecture-design phases of a project was another area of growth that internal departments were pursuing.

3.3 Application Security Boundaries

One of the first questions that we asked interviewees was, *Where is the boundary on what is an "application"?*. The definition of an application is somewhat vague, and can refer to both an individual software executable and a system of interdependent and networked executables running across multiple hosts [11]. Through this question we wanted to elicit the scope of responsibility that organizations have assigned to application security groups.

The practitioners in our sample work with high-complexity software systems. Due to their scale, the work to build and maintain these systems is often organizationally managed along functional lines of different specialties. For example, product regression testing is often the responsibility of a quality assurance (QA) group that is managed independently from the software development group that writes the source code of a product. Similarly, the group responsible for application security, a relatively new specialty, is charged with ensuring the security of software systems.

By asking practitioners about the boundary of "applications", we could learn about what types of security concerns they manage. For example, if practitioners focus more on application source code, they will necessarily pay more attention to flaws like SQL injection as opposed to the patch version of third-party applications and firmware. When asking this question, we presented four levels of system components:

- First-party source code
- Third-party library
- Third-party application (database, application server, operating system)
- Infrastructure (hypervisor, load balancer, router, SAN)

Responses for 11 of the 13 interviewees were split evenly between organizations that are responsible for only the top two levels (first-party source code and third-party libraries), and those responsible for the top three levels. There remaining two practitioners, working in external review organizations, noted that safety-critical systems, such as those found in naval and aviation contexts, are granted certifications at a whole-system level. In these cases, all levels of a system's software are considered during evaluation (hardware is reviewed separately).

It is well known that any component of a system (regardless of its level) can expose security vulnerabilities. The practitioners working in an internal capacity generally revealed that their motivation is to ensure the security of internally-deployed *systems*, thereby requiring attention to all levels of system components. Per their answers to the boundary question, however, most indicated that they focused on ensuring the security of higher-level components (source code & libraries), while others (e.g., an IT group) bore responsibility for lower-level infrastructural components.

We also asked if the boundary is shifting over time. Most practitioners saw a trend toward being responsible for ensuring the security of whole systems (i.e. all levels of system components).

3.4 Goals, Questions, Metrics, and Tools

Understanding, documenting, and disseminating the goals, questions, and metrics that AppSec practitioners use to define and measure their work is a key part of maturing and refining AppSec practices and designing tools that better support their needs.

Modern planning and management practices commonly define four levels of detail about how goals will be achieved. These levels of detail are important for building a shared understanding and encouraging accountability, but are not necessary to understand what practitioners believe to be important [12–15]. For this, we turn to a software engineering method called Goal Question Metric (GQM) approach. This approach was originally defined at the NASA Goddard Space Flight Center and has since been adopted more widely [16].

In the top-down GQM approach, a conceptual-level goal for a product, process, or resource is identified. Then, a set of questions are created to break down the goal into its constituent parts. Finally, metrics are created to define a means of reliably assessing or characterizing an answer to each question [16]. Technically, measures and metrics are separate concepts; from the ISO/IEC/IEE standard on systems and software engineering vocabulary, measures are a systematic way "of assigning a number or category to an entity to describe an attribute of that entity" and metrics are a "combination of two or more measures or attributes" [17]. However, both concepts can serve as metrics in the GQM approach, so we do not distinguish between them in this paper.

Goals. We observed that the overarching goal of application security organizations is to ensure the security of software systems. Security is defined as the state of being confident that "bad things" (i.e., adverse consequences, or losses) cannot happen.

More specifically, every practitioner agreed that the root goal of application security is to achieve the correct risk–cost balance. That is, to reduce expected losses attributable to undesirable behaviors of software applications to an acceptable level, given available risk mitigation and remediation resources.

In other words, application security is a form of risk management. The idea that software security is risk management is echoed in both a guide on software security from the Carnegie Mellon University Software Engineering Institute as well as Dr. Bill Young, a Senior Lecturer and Research Scientist in the Department of Computer Science at the University of Texas at Austin [18, 19].

Questions. Through our interviews, we heard approximately 30 distinct questions that practitioners asked while ensuring the security of software systems. For example, *Where are the defects in my software?*, *What does good performance look like?*, and *Where are my blind spots?*. Many times, the questions that we heard from different participants were variations on a theme; this section synthesizes those raw inputs.

While considering presentation strategies, we identified several possible groupings of questions; for example:

- Using the risk management process, which consists of seven steps: identify risk exposure, measure and estimate risk exposures, assess effects of exposures, find instruments to shift/trade risks, assess costs and benefits of instruments, form a risk mitigation strategy (avoid, transfer, mitigate, keep), evaluate performance [20].
- A list of "management purposes" that we formulated in preparation for the interviews to prompt discussion. This list included eight management categories: security (risk reduction), security control effectiveness, process compliance, engagement & adoption, cost, throughput capacity, staffing, and effect of an intervention.
- Another list that we developed, based on the object of measurement/consideration: an application (or a portfolio of applications), the things that create applications (e.g., people, processes, tools, and interim products), and organizations and policy-level constructs (e.g., roles, policies, and process definitions).
- The NIST Cybersecurity Framework [6] that several respondents volunteered had helped inform their AppSec programs. This framework identifies five areas of management of cybersecurity risk: identify, protect, detect, respond, and recover.
- Various forms of maturity model [4, 5, 21–23].

Each of these groupings has merits, but fails to provide a clear picture of the needs that we observed. The clearest way to convey this picture is a simple, hierarchical list. Ordered roughly in descending frequency of report, this list mirrors what practitioners believe to be important.

The hierarchy also partially conveys the relative sequence of need, which can correlate with maturity – less mature organizations typically develop coarser answers to higher-level questions while more mature organizations have refined their practices to the point where finer-grained answers of sub-questions are relevant. For example, we heard from several people that organizations first focus on onboarding all teams to the secure development lifecycle (SDL) [24]. Similarly, monitoring the frequency or type of defects being introduced by certain teams, or developers, is a later-stage concern.

1. Where are the application vulnerabilities in my software?
   - (a) What should I fix first?
     - (i) What are the highest risk vulnerabilities?
       - (1) What adverse consequences do I face?
2. Where are my blind spots?
   - (a) Is the AppSec program complete and meeting needs?
     - (i) Are policies and procedures documented?
     - (ii) Are roles and responsibilities defined?
     - (iii) Is AppSec group providing all relevant services and meeting needs (e.g., static analysis security testing (SAST) tools, dynamic analysis security testing (DAST) tools, manual code review, penetration testing, architectural analysis, software composition analysis, security guidelines/requirements)
       - (1) Does program need more/different staff/tools/procedures?
       - (2) Does testing cover all relevant types of weakness/vulnerability?
   - (b) Have all teams/projects been onboarded to the SDL?
     - (i) Have all staff had required training?
     - (ii) How do we persuade developers to adopt secure development practices?
   - (c) Are all teams adopting/practicing the SDL?
     - (i) Are teams using the security resources provided by the AppSec group?
       - (1) Are teams following security control requirements and guidelines?
       - (2) Are teams consulting with AppSec analysts?
       - (3) Are teams using the scanner tools that are provided?
   - (d) How much of a system is covered by testing?
   - (e) Have as-built systems drifted from modeled designs (e.g., threat models)?
   - (f) How is the attack surface changing?
   - (g) How do I make attacks/breaches more visible (i.e. increase the probability of detection)?
   - (h) Are the security controls being implemented effective?
3. How do I communicate & demonstrate AppSec's value to my management?
   - (a) What does good performance look like (i.e. benchmark)?
     - (i) Are we meeting the industry standard of care [25]?
   - (b) Is risk decreasing?
   - (c) How do we show that we react quickly to rapidly evolving needs?
   - (d) Are we slowing down "the business" (i.e. what is AppSec's effect on release cadence, or time to market)?
   - (e) What are the financial costs of AppSec?
     - (i) What is the cost of remediation?
     - (ii) How many AppSec employees are employed?
     - (iii) How much do AppSec testing tools cost?
4. Are we getting better at building in security over time?
   - (a) What percent of security requirements/controls are satisfied/implemented?
   - (b) How long do findings/vulnerabilities take to resolve?
   - (c) How long does it take to discover a vulnerability from its introduction?
   - (d) What mistakes are developers making?

        (i) Where is improvement needed?
            (1) On specific projects?
            (2) With certain teams/developers?
                (a) Which teams/developers are introducing defects?
                (b) Is each team/developer introducing fewer defects over time?
            (3) During specific phases of development?
            (4) With specific languages or technologies?
    (e) How much time is spent on security remediation?
    (f) How can software maintenance costs be reduced?
5. Demonstrate compliance with requirements (e.g., internal commitments, external standards such as NIST 800-53, OWASP Top 10 or Application Security Verification Standard (ASVS), and DISA STIGs [26–28])
    (a) Are all teams practicing the SDL?
    (b) What is the severity of the vulnerabilities in my software products?
    (c) Are vulnerabilities being resolved within required time periods?
6. How do I make attacks/breaches more difficult for adversary?
7. What is the AppSec team's input to the broader organization's acquisition decisions of systems/capabilities?
    (a) Is it less expensive to assure the security of software that is built in-house versus acquired from a third party?
    (b) What are the expected ongoing costs of security remediation for a system?
        (i) What system properties contribute most to the cost of maintaining the security of a system?

Metrics. Like with the questions that practitioners ask, there are many ways to group or characterize metrics. For example, we can differentiate between technical versus operational [9], leading versus coincident versus lagging, and qualitative versus quantitative [29] metrics. We can differentiate between metrics that measure processes believed to contribute to the security of a system and those that denote the extent to which some security characteristic is present in a system. We can also evaluate metrics according to how well they satisfy many goodness criteria [30]. As with the questions, though, presenting metrics using these characteristics does not clarify what practitioners measure.

These characteristics *can be* useful as cues on how to interpret metric data. For example, if a code quality metric is assessed qualitatively (i.e., judged by a human reviewer), there is likely variation between assessments of different reviewers. In this case, care should be taken when interpreting scores and one should consider statistical methods to assess inter-rater reliability [31].

Metrics must really be considered in the context of a question that is being asked. Due to space limitations, we cannot discuss each question individually. Instead, we will discuss specific questions that stood out across multiple interviews.

Every practitioner asks question 2.b, *Have all teams/projects been onboarded to the SDL?*. Their objective, often achieved, is to have 100% of development teams and projects in basic compliance with secure development processes. The metrics used to track this are the number and percent of teams with all members having received training, routinely have their application(s) scanned by the AppSec group, and/or been set up with access to centralized, automated security testing tools.

After this basic team coverage is achieved, several practitioners noted shifting their attention to question 2.c, tracking the degree to which development teams apply security tools and thinking to their projects. One practitioner assesses this using question 2.c.i.2 *Are teams consulting with AppSec analysts?*. This practitioner looks at the number of weeks since a team last posed a question to an AppSec analyst and proactively reaches out to teams beyond 4-8 weeks. Other practitioners monitor 2.c.i.3 with the number and frequency of scans run with security testing tools. Another practitioner monitors 2.c per project using the presence of a documented threat model.

For questions 3 and 3.b, *What does good performance look like (i.e. benchmark)?* and *Is risk decreasing?*, every practitioner is interested in the number of defects detected by the various forms of application security testing. Detected defects are the most readily available data to practitioners. These data are used to answer multiple questions, including comparing projects and approximating the security risk in applications.

Because the number of defects increases with the number of lines of code, equation 1 shows how some practitioners normalize defect count to lines of code to support comparing projects. Source lines of code usually doesn't include comments nor whitespace.

$$\text{defect density} = \text{defect count} \div \text{source lines of code} \ . \tag{1}$$

Also, defects are commonly filtered to include only those with the highest severity, as determined by automated security testing tools; most practitioners reported using only "critical" or "high" or "CAT 1" and "CAT 2" severity defects.

Finally, one of the key findings from our study is that practitioners want to measure AppSec risk, but there is no good way of doing so. Two areas that were very challenging for practitioners are assessing risk and communicating with management. These two challenges are likely related: risk is one of management's key areas of concern [32, 33] and risk is fiendishly difficult to assess.

Risk is the potential of gaining or losing something of value. Convention in the information security domain, however, is to use the word to mean potential losses. A risk is fundamentally about an uncertain event; a risk has two components: 1) an event with an expected probability, or likelihood, and 2) an outcome with a cost, or severity. The expected value of a loss associated with a risk is the cost of the event's outcome multiplied by the probability of that event [34–36].

Risk is said to be increased by:

1. Increasing the probability of the event (aka threat)
2. Increasing the probability of the event's success
   (a) Via properties of the attack
   (b) Via properties of the vulnerability
3. Increasing the cost/severity of the outcome

An example of a risk is that a cyber-criminal could exploit a command injection vulnerability on an Internet-facing service to open a remote shell through which they gain access to the internal network and exploit a SQL injection vulnerability on an internal web service and gain access to customer Social Security numbers. In this example, the event is a chained attack via vulnerabilities in two applications and the outcome is the unauthorized disclosure of customer personally identifiable information (PII).

Research in optimal information security investment strategy uncovered many practically infeasible ways to estimate the expected loss of risk. There is insufficient data to estimate the probability of most events [19, 30, 37–39], the complexity and information requirements of modeling system structures are very high [38], and the scope of outcome cost estimates (including things like liability, embarrassment, market-share and productivity losses, extortion and remediation costs [37]) is daunting.

We observed several organizations that make do with manual estimations of risk. One organization uses custom forms in the Atlassian JIRA issue tracker that prompts analysts to assign 8 probability factors that model threat and vulnerability and 6 outcome cost factors that model the financial and operational effects per defect finding. Another organization avoids the difficulty of estimating expected loss by substituting the amount that they would pay out if the bug was discovered in their bounty program. To capture the risk associated with chained attacks that move between different microservices, they developed a system that records trust relationships between services and can report manually-generated threat-risks that are "inherited" from other services [40].

Tools. We found a range of systems for measuring and tracking information to assess the effectiveness of AppSec programs. These systems included ad-hoc personal observations, manually-maintained spreadsheets, use of reporting features in commercial software security tools, basic in-house solutions (e.g., a relational database sometimes with a Web interface), and elaborate in-house solutions (e.g., multiple systems and databases, automated extract transform and load (ETL) jobs, one or more data warehouses, and sometimes third-party governance risk and compliance (GRC) software).

No one measurement and tracking system dominated observed usage. Even within a single organization, multiple solutions are often used to measure and track different metrics. All organizations relied heavily on commercial application security testing tools as the primary source of application vulnerability data, but most augmented these data with results from manual code reviews and manual penetration testing. All organizations correlated and normalized this raw testing data using a vulnerability management system (often Code Dx) to facilitate interpretation and triage of testing findings.

Acknowledgements. This work was made possible by Secure Decisions, the Department of Homeland Security, and AppSec practitioners, many introduced to us by Code Dx, Inc. We would sincerely like to thank these practitioners for their time and candidness during the interviews; this work would have not been possible without their participation.

This material is based on research sponsored by the Department of Homeland Security (DHS) Science and Technology Directorate, Cyber Security Division (DHS S&T/CSD) via contract number HHSP233201600058C. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Department of Homeland Security.

4 References

1. Davis, N.: Developing Secure Software. New York's Software & Systems Process Improvement Network. , New York, NY (2004).
2. 2016 Data Breach Investigations Report. Verizon Enterprise.
3. CISO AppSec Guide: Application Security Program - OWASP, https://www.owasp.org/index.php/CISO_AppSec_Guide:_Application_Security_Program.
4. About the Building Security In Maturity Model, https://www.bsimm.com/about.html.
5. OpenSAMM, http://www.opensamm.org/.
6. Framework for Improving Critical Infrastructure Cybersecurity, https://www.nist.gov/sites/default/files/documents/cyberframework/cybersecurity-framework-021214.pdf, (2014).
7. McGraw, G., Migues, S., West, J.: BSIMM8, https://www.bsimm.com/download.html, (2017).
8. Payne, S.: A Guide to Security Metrics, https://www.sans.org/readingroom/whitepapers/auditing/guide-security-metrics-55, (2006).
9. Sanders, B.: Security metrics: state of the Art and challenges. Inf. Trust Inst. Univ. Ill. (2009).
10. Miles, J., Gilbert, P.: A Handbook of Research Methods for Clinical and Health Psychology. Oxford University Press (2005).
11. Application software, https://en.wikipedia.org/w/index.php?title=Application_software&oldid=826560991, (2018).
12. Steiner, G.A.: Strategic Planning. Simon and Schuster (2010).
13. JP 5-0, Joint Planning, http://www.jcs.mil/Doctrine/Joint-Doctrine-Pubs/5-0-Planning-Series/, (2017).
14. Douglas, M.: Strategy and tactics are treated like champagne and two-buck-chuck, https://prestonwillisblog.wordpress.com/2015/05/15/strategy-and-tactics-are-treated-like-champagne-and-two-buck-chuck/, (2015).
15. Marrinan, J.: What's the difference between a goal, objective, strategy, and tactic?, http://www.commonbusiness.info/2014/09/whats-the-difference-between-a-goal-objective-strategy-and-tactic/, (2014).
16. Basili, V.R., Caldiera, G., Rombach, H.D.: The goal question metric approach. Encycl. Softw. Eng. 2, 528–532 (1994).
17. ISO/IEC/IEEE 24765:2010(E) Systems and software engineering — Vocabulary, https://www.iso.org/standard/50518.html, (2010).
18. Allen, J.: How Much Security is Enough, https://resources.sei.cmu.edu/asset_files/WhitePaper/2013_019_001_295906.pdf, (2009).
19. Young, B.: Measuring Software Security: Defining Security Metrics. (2015).
20. Crouhy, M., Galai, D., Mark, R.: The Essentials of Risk Management. McGraw-Hill, New York (2005).
21. Krebs, B.: What's Your Security Maturity Level?, https://krebsonsecurity.com/2015/04/whats-your-security-maturity-level/, (2015).
22. Richardson, J., Bartol, N., Moss, M.: ISO/IEC 21827 Systems Security Engineering Capability Maturity Model (SSE-CMM) A Process Driven Framework for Assurance.

23. Acohido, B., Sager, T.: Improving Detection, Prevention and Response with Security Maturity Modeling, https://www.sans.org/reading-room/whitepapers/analyst/improving-detection-prevention-response-security-maturity-modeling-35985, (2015).
24. Microsoft Security Development Lifecycle, https://www.microsoft.com/en-us/sdl/default.aspx.
25. Olcott, J.: Cybersecurity: The New Metrics, https://www.bitsighttech.com/hubfs/eBooks/Cybersecurity_The_New_Metrics.pdf?t=1509031295345&utm_source=hs_automation&utm_medium=email&utm_content=37546190&_hsenc=p2ANqtz--m-crOcN48EycaIJFVXnHInTyc_LOO2aQWbl5YHXd3Fz34z7w0EfMptTs1_XnOGjEH_6jM_g6FUJUgAMYFSjV06QDmyQ&_hsmi=37546190, (2016).
26. SP 800-53 Rev. 5 (DRAFT), Security and Privacy Controls for Information Systems and Organizations, https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/draft.
27. Wichers, D.: Getting Started with OWASP: The Top 10, ASVS, and the Guides. 13th Semi-Annual Software Assurance Forum. , Gaithersburg, MD (2010).
28. Application Security & Development STIGs, https://iase.disa.mil/stigs/app-security/app-security/Pages/index.aspx.
29. Jansen, W.: Directions in security metrics research, http://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7564.pdf, (2009).
30. Savola, R.: On the feasibility of utilizing security metrics in software-intensive systems. Int. J. Comput. Sci. Netw. Secur. 10, 230–239 (2010).
31. Hallgren, K.A.: Computing Inter-Rater Reliability for Observational Data: An Overview and Tutorial. Tutor. Quant. Methods Psychol. 8, 23–34 (2012).
32. The 15-Minute, 7-Slide Security Presentation for Your Board of Directors, https://blogs.gartner.com/smarterwithgartner/the-15-minute-7-slide-security-presentation-for-your-board-of-directors/.
33. Gaillard, J.C.: Cyber Security: Board of Directors Need to ask the Real Questions, http://www.informationsecuritybuzz.com/articles/cyber-security-board-of-directors-need-to-ask-the-real-questions/, (2015).
34. Risk, https://en.wikipedia.org/w/index.php?title=Risk&oldid=824832006, (2018).
35. Gordon, L.A., Loeb, M.P.: The economics of information security investment. ACM Trans. Inf. Syst. Secur. TISSEC. 5, 438–457 (2002).
36. Expected value, https://en.wikipedia.org/w/index.php?title=Expected_value&oldid=826427336, (2018).
37. Hoo, K.J.S.: How much is enough? A risk management approach to computer security. Stanford University Stanford, Calif (2000).
38. Schryen, G.: A Fuzzy Model for IT Security Investments. (2010).
39. Böhme, R.: Security Metrics and Security Investment Models. In: IWSEC. pp. 10–24. Springer (2010).
40. Held, G.: Measuring End-to-End Security. AppSecUSA 2017. , Orlando, FL (2017).