Enterprise Security: Why Do We Make It So Difficult?

Bob Duncan
Computing Science
University of Aberdeen
Aberdeen, UK
Email: bobduncan@abdn.ac.uk

Mark Whittington
Business School
University of Aberdeen
Aberdeen, UK
Email: mark.whittington@abdn.ac.uk

Victor Chang
International Business School Suzhou
Xian Jiaotong Liverpool University
PR China
Email: ic.victor.chang@gmail.com

***Abstract*—Achieving information security and privacy is not a trivial exercise. This becomes much more challenging in the cloud, due to the multi-tenancy nature of cloud ecosystems. We are concerned that the traditional legacy compatible approach to software development is holding enterprises back from achieving effective security and privacy, particularly in the cloud. In this paper we discuss the implications of the traditional approach to software development and question why we stick to this approach, despite the fact that this approach makes the job of security and privacy far more difficult.**

***Index Terms*—Cloud security and privacy; legacy software**

I. Introduction

In previous work [1], we questioned whether we should be using a different approach to achieving information security in the cloud, and concluded that there were two major conflicts that needed to be addressed. The first concerns the ever increasing technical complexity of software systems, and the ever more technically complex solutions being developed to address these problems. The second concerns the presence of technically simple flaws, which continue to be exploited year after year by attackers. We now take a slightly different approach to this work in order to try to understand what may be making life much more difficult than it need be for a step change in security.

Before looking at the first issue, we need to consider the evolution of software development, and in Section II, we run through the traditional approach to software development, in order to understand how this situation arose. Thus informed, in Section III, we will start to address the first conflict, by considering why software appears to be becoming more and more complex. In Section IV, we ask why this complexity presents such a problem for security and privacy, in Section V, we identify some simple flaws which consistently fail to be addressed, and in Section VI, we question why this should be. The rest of the paper is laid out as follows: in Section VII, we consider how we might approach resolving these issues, and in Section VIII, we draw our conclusions.

II. The Traditional Approach to Software Development

To consider how this approach evolved, we need to go back in time to when computers first came on the scene. They had far less memory than we are accustomed to have today. Hard disk sizes were minuscule compared with today's mega sized drives. Consequently, operating systems and programmes were necessarily compact. Early desktops, such as the first IBM PC [2], introduced in 1981, were capable of being run without a hard drive, with only one or two floppy drives of 360KB capacity. One floppy disk for the operating system, and one for both programmes and data. Why, with two drives, you could have a floppy for each, swapping between operating system and programme floppies, leaving a "vast" space for data alone.

Needless to say, this state of affairs did not last for long. Thanks to the benefits of Moore's Law [3], hardware grew in both size and performance, relentlessly, year on year, and costs kept falling. Operating systems very quickly grew in size and complexity, as did programmes. Data files grew larger in size and volume. Often, this rapid expansion in software capability and size would even outpace Moore's Law, leading to hardware bottlenecks. When Microsoft had developed the Windows XP operating system, it had an installed size of around 1.5GB. Once Vista was released, the installed size increased to around 19.5GB — a massive jump. This additional "bloat" meant that Vista would not run on some of the older hardware which had happily run XP for years.

While the core operating system components of Vista were only around 25% larger than XP, many of the other programmes were first loaded into memory, to ensure a more rapid response if the user wished to use them. Unfortunately, there simply was not enough RAM to make this a viable approach. There were also security issues with XP, whereby all users were created with administrator privileges, thus unwittingly exposing them to hackers and malware. As Jerome Saltzer said about security [4], "Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job". A point rather lost with XP.

This endless quest to pack more and more features into both operating systems and software resulted in monolithic systems capable of delivering considerably more programmes than any user actually wanted or needed. In the constant quest for user "ease of use", most systems needed to have root permission to run, otherwise they were troublesome to install and operate. Not such an issue in the days before the internet, but once

it arrived, soon criminals were having a field day, breaking in to systems and stealing confidential data, or even cash. For decades, ease of use had the upper hand, meaning that default installation settings were geared to making the software deploy effortlessly. Security and privacy were very much an afterthought, due to the difficulties associated with configuring such needs [4], and this only served to fuel the opportunity for criminals.

Early software development was based on the "engineering" approach, where the development was broken down into phases such as analysis, planning, design, build and deployment. These were carried out in sequence, which became known as the "waterfall" method, whereby each phase was completed before moving on to the next, first presented by Herbert D. Benington at a Symposium [5], on advanced programming methods for digital computers on 29 June 1956. Analysis would try to elicit what the requirements were, and planning would attempt to schedule the workload properly. The design phase was very rigorous, and took the lion's share of the effort. Once all the details were understood, the build time was generally very rapid. Deployment, could be fairly rapid, providing everything else had been properly specified and carried out to standard. This system either would work very well, particularly for large projects, or it would fail spectacularly.

Incremental software development methods can be traced back to 1957 [6]. However, it was not until around the turn of the century [7], that the waterfall method really started to give way to the "agile" process. Here, the process is highly adaptive, involving less bureaucracy, whilst retaining a high level of commitment. Client feedback is encouraged throughout the project, leading to a faster overall solution. This process is very well suited to dynamic projects.

The common factor to both these methodologies is that the main focus of each is on properly designing the system around system processes. The key driver here is information flow control, where most of the emphasis is on capturing the information flow, with far less interest paid to information control. Legacy compatibility is another driver, arising from the desire to re-use code wherever possible. The trouble with this approach is that the code being re-used was in all likelihood developed some considerable time previously, meaning the code gets re-used "warts and all", including all the security and privacy deficiencies brought forward.

III. The Increasing Complexity of Systems

The fundamental concepts of information security are confidentiality, integrity, and availability (CIA), a concept developed when it was common practice for corporate management to run an enterprise under agency theory. We have seen how agency theory has failed to curb the excesses of corporate greed. The same is true for cloud security, which would suggest a different approach is needed [8]. The business environment is constantly changing [9], as are corporate governance rules and this would clearly imply changing security measures would be required to keep up to date. More emphasis is now being placed on responsibility and accountability [10], social conscience [11], sustainability [12][13], resilience [14][15] and ethics [16]. Responsibility and accountability are, in effect, mechanisms we can use to help achieve all the other security goals. Since social conscience and ethics are very closely related, we can expand the traditional CIA triad to include sustainability, resilience and ethics.

Does this expansion of security needs account for the increasing complexity of software alone? No, it does not. There are many other reasons behind this increase. First, as already stated in Section II, the long running benefits offered by Moore's Law [3], have continued to offer ever more capable and powerful computing systems. Every step of the way, evolving technology has brought more and more innovations, with the capability and quality of software increasing apace to take full advantage of this evolving technology. The evolution of distributed systems removed the potential bottleneck of high cost mainframe systems, which could only be afforded by the largest enterprises, and opened up more opportunities for smaller enterprises. This increase in hardware demand brought with it the demand for better, and cheaper software. The evolution of corporate computing models such as Business Process Management, Service Oriented Architecture, Grid Computing, Utility Computing, Virtualization, Corporate Outsourcing, and Cloud Computing have influenced the development of radical change in how enterprises do business.

These changes, however, have not come without an increase in complexity. Apart from the complexity of more and more complex hardware, software systems, too, have increased in complexity. More complex systems mean more inter-connected relationships being developed between systems, enterprises, and third party organisations, such as agents, facilitators and services providers. While this can lead to better systems being available to enterprises, it can also increase the risks they face in installing, configuring and maintaining such systems, due to the additional configuration needed to ensure the systems become secure. All too often, the default settings are designed to ensure complete ease of installation, yet this is usually at the expense of good security and/or privacy. Successful configuration for security and privacy purposes is usually a highly complex task, which generally proves very difficult to achieve in practice. We therefore need to consider why this should be so.

IV. Why Should This Complexity Present Such a Problem for Security and Privacy?

Achieving effective information security is not a trivial problem to address successfully, and this becomes much more difficult in a cloud setting. The business architecture of an enterprise comprises a combination of people, process and technology [17], and we have long argued over the weaknesses presented by a purely technical approach [18][19][20][1][9][8], where such approaches ignore the impact of people and process on security. However, we equally should not lose sight of possible issues presented by technology. While computing hardware is generally robustly made these days, and has few

weaknesses, nonetheless, we still need to keep an eye out for possible issues. However, of far greater concern is the software used to run on the hardware we all use today. By far the greatest number of security vulnerabilities arise from the weaknesses present in software, and it is this aspect that we shall focus on in this paper.

For example, if we wanted to build a house, one of the fundamental requirements would be to build it upon solid foundations. Yet, we happily persist in missing this point entirely when it comes to modern software, building systems on poor foundations time and time again. We refer, of course, to the practice of building in legacy compatibility for software. This has been common practice for decades, with banks particularly keen on encouraging the practice to ensure legacy hardware can continue to be used within their overall system. Microsoft have been instrumental in using this practice for their operating systems for decades. While, in theory, this is a laudable goal, in practice, this can give rise to some serious security issues, particularly in the case where early Microsoft operating systems relied extensively on root access for all programmes to ensure smooth running of the operating system for even novice users. Unfortunately, this is not an approach which is compatible with good security.

However, this approach is not confined to operating systems alone. Virtually all software systems in use today have their origins in the dim and distant past, when security was much less of an issue (think of pre-internet days). However, security, and privacy, have become much more of an issue in the corporate business world of today. But with the built-in backward compatibility option being a fundamental part of the sales pitch for many software products, there is no doubt that this approach can help perpetuate bugs and vulnerabilities year after year.

Antoine de Saint-Exupery [21], once said, "Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away." He was an engineer, but his statement could equally apply to the writing of software code. Vulnerabilities exist in a very high proportion of all software in use today. Most source lines of code reviews and estimates suggest that a very conservative guess would place the number of of bugs at the rate of 1 per 1,000 lines of extremely well written source code with great attention to security detail. The reality is far from this. Capers Jones [22], suggested that the reality is more like 5 bugs per Java method, with a method taking 50 - 55 lines of code, i.e. approximately a 10% error rate. When you consider that the old Microsoft XP had 40 - 45 million lines of source code, that would suggest around 4.5 million bugs were in the first released version. Probably not too far from the truth, since it took over 12 years to fix the bugs before support was discontinued. And that is just an operating system. Add some complex applications to this, and the number of lines will increase radically.

A quick look at the annual security breach reports issued by many security companies [23][24][25] will clearly demonstrate the security and privacy problems still faced today, including the fact that the same attacks continue to be successful year on year, as demonstrated by the five year summary of the Verizon reports shown in Table I below.

Threat	2010	2011	2012	2013	2014
Hacking	2	1	1	1	1
Malware	3	2	2	2	2
Misuse by company employees	1	4	5	5	5
Physical theft or unauth. access	5	3	4	3	4
Social Engineering	4	5	3	4	3

TABLE I: Verizon Top 5 Security Breaches — 2010-2014 (1=Highest) [26][27][28][29][23]

This raises a number of obvious questions. Why should we need this increasing complexity? Why does increasing complexity make security and privacy more challenging? Why is it that the same attacks are successful year on year?

V. The Simple Flaws Under Attack Year on Year

Before addressing the questions raised in Section IV above, we will consider what these weaknesses might be in a little more detail. An obvious place to start is the public face of an enterprise — web services. These provide the main point of entry for adversaries trying to breach the enterprise systems. For this, we turn to the Open Web Application Security Project (OWASP) Top Ten Vulnerabilities list. OWASP is an international organisation which engages in the conception, development, acquisition, operation, and maintenance of applications that can be trusted. Every three years, they produce a Top Ten list of the most dangerous vulnerabilities.

2013 Code	Threat
A1	Injection Attacks
A2	Broken Authentication and Session Management
A3	Cross Site Scripting (XSS)
A4	Insecure Direct Object References
A5	Security Misconfiguration
A6	Sensitive Data Exposure
A7	Missing Function Level Access Control
A8	Cross Site Request Forgery (CSRF)
A9	Using Components with Known Vulnerabilities
A10	Unvalidated Redirects and Forwards

TABLE II: OWASP Top Ten Web Vulnerabilities — 2013 [30]

Looking at this list, it seems obvious that these flaws are not particularly technically complex. Yet they are successful year, after year, after year.

VI. Why might this be?

In this section, we will attempt to address the three questions raised in Section IV, namely:

- Why should we need this increasing complexity?
- Why does increasing complexity make security and privacy more challenging?
- Why is it that the same attacks are successful year on year?

A. Why should we need this increasing complexity?

Starting with this question, why? Granted, the hardware we have today is infinitely more powerful than previous versions were decades ago. Thus, this greater hardware capacity allows us to run ever more complex software, capable of carrying out more detailed and ever more difficult tasks for us. We can perform every more detailed analysis on everything we collect through these complex systems, allowing us to become more efficient at management of the business as a consequence.

But the wonder of these enhanced capabilities comes at a price. We need ever more capable staff to operate these complex systems. We create more and more data than we ever did decades ago, which needs to be stored securely. Staff therefore need more and more training. Enterprises also are now subject to greater legislation and regulation than ever before. Large public enterprises in the UK, for example, also have to comply with corporate governance guidance. This means there is a much greater need to demonstrate compliance to a range of interested parties.

This new ability to be able to "Micro-manage" an enterprise, can also work against efficient management, as there is a balance to be struck between the effort required in (and costs associated with) maintaining these complex systems, which can often far exceed the benefits to be obtained.

B. Why does increasing complexity make security and privacy more challenging?

As software becomes more complex, so more and more source code is required for this more complex system to function. As we mentioned in Section IV, the more source code we write, the more the possibility of introducing bugs and errors expands. Of course, many enterprises do not write their own code, but instead outsource this task to professional organisations.

However, perhaps more importantly, since much of this software has evolved from earlier versions, (or decades), it may well be that security and privacy were less of a priority in previous years, and thus this practice allows a poorer level of security and privacy to be achieved.

There is also no doubt that the greater complexity of this software means there is now a vastly increased range of settings that can be configured to optimise the software for each enterprise. Sadly, this often results in many of the default settings being set up in such a way as to make installation, especially by unfamiliar users, much more quick and simple to execute. This is usually achieved at the expense of robust security and privacy, as these requirements tend to be far more complex to set up properly. Often, an enterprise fails to make clear at the beginning of such a contract that security and privacy are high priority for them, often resulting in the message not getting through. This usually results in a far from satisfactory security and privacy level being achieved.

Should the reader not wish to believe this state of affairs could possibly exist in a modern enterprise, we refer the reader to the results shown in TABLE I in Section IV, which clearly demonstrate how these trivial breaches continue to be an issue, year on year, conveniently leading us to the next question.

C. Why is it that the same attacks are successful year on year?

The simple reason for that is that they work. Often, an enterprise that has been breached may not even know it has happened, and the breaches in that case will keep on happening. Sometimes, having been breached, this is frequently brought to the enterprise's attention by an outside party, rather than having discovered the breach themselves [26].

The attackers are often quite lazy in their approach. They simply perpetrate the same attacks, time after time, and once a victim enterprise closes the vulnerability, they simply move on to the next potential victim. Generally all enterprises are being attacked all the time, 24/7, for 365 days a year. Where the enterprise has good defences in place, it is not long before the attacker moves on to easier pickings.

As a general rule, the larger enterprises tend to be attacked by the more serious attackers, while the SME sized enterprises are considered fair game, as they generally are likely to be less able to afford the right calibre of technical staff to properly defend their systems. Often, sysadmins in these smaller enterprises don't install security patches regularly, thus leaving vulnerabilities wide open to attack.

As far as the attackers are concerned, If it ain't broke, don't fix it. If it works, keep doing it. And they do, with considerable success, year, on year, on year.

VII. How do we tackle these challenges?

Recalling the long history, and therefore the greater lessons learned, we can refer to some lessons from accounting. Separation (also known as segregation) of duties is a standard factor considered in the design of accounting workflow and checked by any internal audit. The central idea is that if tasks within a workflow are given to different people (separating responsibility for ordering from invoicing, for example) then firstly the opportunity for fraud is much reduced and secondly a second person may spot an unintentional error. In the years immediately following the introduction of the Sarbanes-Oxley Act (2002) [31], in the USA (which required a greater examination of internal problems), Ge and McVay [32], found that complex, smaller and profit-challenged companies were more likely to report problems including inadequate separation of duties.

The relevance of this to software development is also obvious, with the Information Systems Audit and Control Association (ISACA) [33], including the following information in their glossary: "A basic internal control that prevents or detects errors and irregularities by assigning to separate individuals the responsibility for initiating and recording transactions and for the custody of assets".

The appropriate separation of duties between individuals clearly needs to be matched by a separation of processes within information technology. So, for example, an audit trail that records the person, time and nature of each sequential transaction should be separate, i.e. on a different instance or

machine, depending on the architecture of the enterprise, so that an attacker would not be able to cover their tracks by cleansing the audit trail following an intrusion. This important aspect of separation of duties is often overlooked in information systems.

Of course, there are many other things that can, and need, to be done to tackle these challenges. The first thing to do is to recognise that something needs to be done. Until the enterprise accepts that a problem exists, no real improvement in the status quo is possible.

The UK government provide a passport to good security on the website of the Centre for the Protection of National Infrastructure [34], which provides advice to enterprises involved in national infrastructure and encourages them to adopt good practices. They also offer specific advice on cyber security. Naturally, these enterprises are at higher risk than many other enterprises, but it would do no harm to consider following their recommendations. The UK government also provide specific advice on implementing the cloud security principles [35].

Other major organisations also offer good advice in this respect. On their website, OWASP [36], offers a full range of recommendations to help mitigate the security vulnerabilities they review. A good fist step would be to pay careful attention to this advice, and set about implementing all the recommended advice they provide. For every vulnerability that they discuss, and this is not limited to just the top ten vulnerabilities, they provide good practical advice on how to mitigate all the problems they discuss. It is clear from looking at the security reports that many enterprises are failing to implement these, often simple, recommendations in order to secure their systems.

In the EU, ENISA [37], provide a report on cloud computing benefits, risks and recommendations for information security. In the US, NIST [38], provide extensive advice on cloud computing on their website. The Cloud Security Alliance (CSA) [39], one of the main cloud standards organisations, provide a report on security guidance for critical areas of focus in cloud computing. One of the largest database software providers, Oracle, provide a report [40], on their optimized solution for secure enterprise cloud infrastructure. Cisco [41], one of the largest communications manufacturers, provide a report on their Cisco cloud enablement services.

All the major cloud service providers, such as Amazon, Google, HP, IBM, Microsoft, and specialised providers such as Salesforce, provide detailed suggestions and recommendations on how to set up secure cloud systems on their resources. Ramachandran and Chang [42], provide comprehensive recommendations and best practices for cloud security. Also most governments throughout the globe now offer advice on appropriate steps to take in order to set up secure cloud applications.

It is certainly vital for enterprises to realise the dangers of installing software while placing blind reliance on default settings. Default settings are geared to ensure the installation of their software is easy to carry out. Achieving good security and privacy is not a trivial exercise, and the configuration effort needed is challenging, and requires a great deal of effort. There are no short cuts to a successful and secure implementation.

A great many software application systems were designed for traditional distributed systems, and the systems architectures involved were necessarily highly complex. Many of these software applications have been moved to a cloud environment without giving any consideration to the additional relationships involved in a cloud setting, nor any consideration to the impact of running parts of the systems out-with the secure firewall surrounding a normal distributed system.

When it comes to any web based application featuring a database back end, it is important to give consideration to system logging for forensic examination purposes [8]. After a successful injection attack, one of the first goals of the attacker is to delete, or modify the audit trail and system logs, in order to cover their tracks. A fairly obvious solution to this possibility would be to ensure that all system logs are logged to a completely separate database, not hosted on the same cloud machine. Particular attention should be paid to the logging instance, to ensure that it is not accessible from the main web application. Further steps should be taken to ensure that the database is properly configured, by first ensuring no direct access is available to users of the web application. Second, to configure the logging database to specifically prevent modifications or deletions to be performed by the web application. While this will not stop an attacker if they can gain root access to the database application, if there is no direct access available from the web application, this will make their job far more difficult.

VIII. Conclusions

While these measures will help to provide a short term solution for an enterprise, it is clearly far from satisfactory. Most software systems can trace their ancestry back to the times of pre-internet early enterprise systems. Here, the focus was on usability, not on security nor privacy.

When it comes to software implementations, enterprises would do well to learn the lessons of separation of duties, both for staff, process and technology, particularly in the case of audit trail logging.

As to the development of new software, there is a clear need to develop secure software systems from the ground up, rather than try to add security to an already complex piece of software as an afterthought. There is no doubt that there is a place for the development of an immutable database system, which would be particularly suitable for system logging and would be especially useful for deployment in accounting systems.

References

[1] B. Duncan and M. Whittington, "Information Security in the Cloud: Should We be Using a Different Approach?" in *2015 IEEE 7th Int. Conf. Cloud Comput. Technol. Sci.*, Vancouver, 2015, pp. 1–6.

[2] IBM, "IBM PC release," 1981. [Online]. Available: https://www-03.ibm.com/ibm/history/exhibits/pc25/pc25_intro.html

[3] G. Moore, "Cramming More Components Onto Integrated Circuits," *Electronics*, vol. 38, no. April 19, pp. 114–117, 1965.

[4] J. H. Saltzer, "Protection and the control of information sharing in Multics," *Commun. ACM*, vol. 17, no. 7, pp. 388–402, 1974.

[5] US Navy Mathematical Computing Advisory Panel, "Symposium on advanced programming methods for digital computers." Washington, DC: Office of Naval Research, Dept. of the Navy, OCLC 10794738, 1956.

[6] R. Victor, "Iterative and incremental development: A brief history," *IEEE Comput. Soc.*, pp. 47–56, 2003.

[7] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, and Others, "Manifesto for agile software development," 2001.

[8] B. Duncan and M. Whittington, "Enhancing Cloud Security and Privacy: The Power and the Weakness of the Audit Trail," in *Cloud Computing 2016: The Seventh International Conference on Cloud Computing, GRIDs, and Virtualization*. Rome: IEEE, 2016, pp. 125–130.

[9] B. Duncan and M. Whittington, "Enhancing Cloud Security and Privacy: The Cloud Audit Problem," in *Cloud Computing 2016: The Seventh International Conference on Cloud Computing, GRIDs, and Virtualization*. Rome: IEEE, 2016, pp. 119–124.

[10] M. Huse, "Accountability and Creating Accountability: a Framework for Exploring Behavioural Perspectives of Corporate Governance," *Br. J. Manag.*, vol. 16, no. S1, pp. S65–S79, mar 2005.

[11] A. Gill, "Corporate Governance as Social Responsibility: A Research Agenda," *Berkeley J. Int'l L.*, vol. 26, no. 2, pp. 452–478, 2008.

[12] C. Ioannidis, D. Pym, and J. Williams, "Sustainability in Information Stewardship: Time Preferences, Externalities and Social Co-Ordination," in *Weis 2013*, 2013, pp. 1–24.

[13] A. Kolk, "Sustainability, accountability and corporate governance: Exploring multinationals' reporting practices." *Bus. Strateg. Environ.*, vol. 17, no. 1, pp. 1–15, 2008.

[14] F. S. Chapin, G. P. Kofinas, and C. Folke, *Principles of Ecosystem Stewardship: Resilience-Based Natural Resource Management in a Changing World*. Springer, 2009.

[15] V. Chang, M. Ramachandran, Y. Yao, Y. H. Kuo, and C. S. Li, "A resiliency framework for an enterprise cloud," *Int. J. Inf. Manage.*, vol. 36, no. 1, pp. 155–166, 2016.

[16] S. Arjoon, "Corporate Governance: An Ethical Perspective," *J. Bus. Ethics*, vol. 61, no. 4, pp. 343–352, nov 2012.

[17] PWC, "UK Information Security Breaches Survey - Technical Report 2012," London, Tech. Rep. April, 2012. [Online]. Available: www.pwc.comwww.bis.gov.uk

[18] B. Duncan, D. J. Pym, and M. Whittington, "Developing a Conceptual Framework for Cloud Security Assurance," in *Cloud Comput. Technol. Sci. (CloudCom), 2013 IEEE 5th Int. Conf. (Volume 2)*. Bristol: IEEE, 2013, pp. 120–125.

[19] B. Duncan and M. Whittington, "Compliance with Standards, Assurance and Audit: Does this Equal Security?" in *Proc. 7th Int. Conf. Secur. Inf. Networks*. Glasgow: ACM, 2014, pp. 77–84.

[20] B. Duncan and M. Whittington, "The Importance of Proper Measurement for a Cloud Security Assurance Model," in *2015 IEEE 7th Int. Conf. Cloud Comput. Technol. Sci.*, Vancouver, 2015, pp. 1–6.

[21] A. de Saint-Exupéry, *Airman's Odyssey*. Houghton Mifflin Harcourt, 1943.

[22] C. Jones, "Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies," 2010.

[23] Verizon, "2014 Data Breach Investigations Report," Tech. Rep. 1, 2014. [Online]. Available: http://www.verizonenterprise.com/resources/reports/rp_Verizon-DBIR-2014_en_xg.pdf

[24] PWC, "2014 Information Security Breaches Survey: Technical Report," Tech. Rep., 2014.

[25] Trustwave, "Trustwave Global Security Report," Tech. Rep., 2013. [Online]. Available: https://www2.trustwave.com/2013GSR.html

[26] W. Baker, M. Goudie, A. Hutton, D. Hylender, J. Niemantsverdriet, C. Novak, D. Ostertag, C. Porter, M. Rosen, B. Sartin, P. Tippett, T. Bosschert, E. Brohm, C. Chang, M. Dahn, R. Dormido, B. Van Erck, K. Evans, E. Gentry, J. Grim, C. Hill, A. Kunsemiller, K. Lee, W. Lee, K. Long, R. Perelstein, E. Telemaque, D. Todd, Y. Uzawa, J. A. Valentine, N. Villatte, M. Van Der Wel, P. Wright, T. Beeferman, C. Dismukes, P. Goulding, and C. Neal, "2010 Data Breach Investigations Report," Tech. Rep., 2010.

[27] Verizon, "2011 Data Breach Investigation Repeort: A study conducted by the Verizon RISK Team in cooperation with the United States Secret Service and Others," Verizon/USSS, Tech. Rep., 2011.

[28] Verizon, N. High, T. Crime, I. Reporting, and I. S. Service, "2012 Data Breach Investigations Report," Verizon, Tech. Rep., 2012.

[29] Verizon, "Verizon2013," Tech. Rep.

[30] OWASP, "OWASP Top Ten Vulnerabilities 2013," 2013. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[31] Sox, "Sarbanes-Oxley Act of 2002," p. 66, 2002. [Online]. Available: news.findlaw.com/hdocs/docs/gwbush/sarbanesoxley072302.pdf

[32] W. Ge and S. McVay, "The disclosure of material weaknesses in internal control after the Sarbanes-Oxley Act," *Account. Horizons*, vol. 19, no. 3, pp. 137–158, 2005.

[33] ISACA, "Monitoring Internal Control Systems and IT." [Online]. Available: http://www.isaca.org/knowledge-center/research/researchdeliverables/pages/monitoring-internal-control-systems-and-it.aspx

[34] CPNI, "Passport to Good Security," 2015. [Online]. Available: http://www.cpni.gov.uk/advice/Passport-to-Good-Security/

[35] HMG, "Implementing the Cloud Security Principles," 2016. [Online]. Available: https://www.gov.uk/government/publications/implementing-the-cloud-security-principles/implementing-the-cloud-security-principles

[36] OWASP, "OWASP Top 10 Web Vulnerabilities - Details." [Online]. Available: https://www.owasp.org/index.php/Top_10_2013-Top_10

[37] ENISA, "Cloud Computing: Benefits, Risks and Recommendations for Information Security." [Online]. Available: https://resilience.enisa.europa.eu/cloud-security-and-resilience/publications/cloud-computing-benefits-risks-and-recommendations-for-information-security

[38] NIST, "NIST Computer Security Resource Center (CSRC)," 2016. [Online]. Available: http://csrc.nist.gov/

[39] CSA, "Security Guidance for Critical Areas of Focus in Cloud Computing v3.0," Tech. Rep., 2011. [Online]. Available: https://downloads.cloudsecurityalliance.org/assets/research/security-guidance/csaguide.v3.0.pdf

[40] Oracle, "Oracle Optimized Solution for Secure Enterprise Cloud Infrastructure," Tech. Rep., 2015. [Online]. Available: http://www.oracle.com/technetwork/server-storage/hardware-solutions/o12-043-cloud-sparc-1659149.pdf

[41] Cisco, "Cisco 2011 Annual Security Report," Cisco, Tech. Rep., 2011.

[42] M. Ramachandran and V. Chang, "Recommendations and Best Practices for Cloud Enterprise Security," *2014 IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, pp. 983–988, 2014.