Creating an Immutable Database for Secure Cloud Audit Trail and System Logging

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

***Abstract*—Conventional web based systems present a multiplicity of attack vectors. One of the main components, the database, is frequently configured incorrectly, often using default settings, which leave the system wide open to attack. Once a system has been attacked, valuable audit trail and system log data is usually deleted to cover the trail of the perpetrator. Given the average industry time between breach and discovery, there is often little forensic trail left to follow. Of equal importance is that in cloud settings, where new instances are automatically spooled and shut down to follow the demand curve, any data stored on the running instance before shut down is lost. We demonstrate how the configuration of a simple immutable database, running on a separate private system can go a long way to resolving this problem.**

***Index Terms*—Cloud security and privacy; immutable database; forensic trail.**

I. Introduction

Achieving information security is not a trivial process, and in the context of cloud computing, it becomes increasingly more difficult. Because cloud technology is enabled by the Internet, one of the key weaknesses comes from web services, which invariably are structured with a database back-end. There are a host of well understood vulnerabilities surrounding the use of modern databases, and while there are a number of mitigating strategies that can be deployed, often they are not, as evidenced by their continual recurrence on annual security breach reports.

Duncan and Whittington [1] have written about the difficulties surrounding proper audit of cloud based systems. They have talked about the need for enterprises to maintain a proper audit trail in their systems, and about the weaknesses arising as a result of poor configuration of database systems, particularly in the context of cloud systems [2]. They have proposed addressing this problem through the use of an immutable database for the purpose of secure audit trail and system logging for cloud applications [3].

Some five years ago in 2012, Trustwave [4], were reporting an average time taken by enterprises of 6 months between breach and discovery. Discovery was often made by third parties external to the enterprise, rather than by the enterprise themselves. This time lag between breach and discovery has been significantly reduced, but nevertheless is a great concern, particularly in the light of forthcoming legislation, such as the ED General Data Protection Regulation (GDPR). Looking at the latest security breach reports, it is clear that many enterprises will be unable to comply with the requirement to report any breach within 72 hours. This would suggest that many firms are not monitoring their systems properly, do not maintain proper audit trails, thus leading to inadequacy in retaining a proper forensic trail to understand exactly what information has been accessed, modified or deleted.

In this paper, we outline how we might approach developing a solution to satisfy these issues and concerns. In Section II, we provide some background and discuss the motivation for this work, and in Section III, we discuss what an immutable database needs to be. In Section IV, where we outline how we can create and configure an immutable database using existing software, in this case we have chosen MySQL for illustrative purposes. In Section V, we discuss typical attack vectors against database systems. In Section VI, we discuss our conclusions.

II. Background and Motivation

In this paper, we use the MySQL database language to illustrate what is currently possible. While not all databases are exactly similar, most exhibit the same weaknesses, often arising through improper configuration. Equally, the software environment chosen to integrate with the database is often subject to the same poor configuration, thus leading to the ongoing success of attackers. These weaknesses in configuration are frequently exploited by attackers, and there is often a poor understanding of how proper use of the audit trail can help to improve security significantly. Thus, we shall first discuss the purpose of audit and the significance of the audit trail.

A. Audit and the Audit Trail

There are many areas of business activity that merit diligent checking and verification by an objective person or organization from outside the organization itself. Some of these may be undertaken voluntarily by the firm, others such as the audit of financial systems and results are mandated. Clearly cloud computing audit is a new, immature field and it would be surprising if there were not lessons to learn from the experiences — and failures — of audit processes and practices that have been honed over decades if not centuries [5].

Whenever a new technical area emerges it will be difficult to find people with the appropriate skillset — a technical knowledge of the area and competency in carrying out an audit. As commercial organisations, audit companies may seek to extend their audit competence into new technical areas, not just cloud audit, but perhaps environmental audit as another example. Over a century of experience in the development of audit tools and practices then needs to be applied to a new technical domain. Alternatively, computing specialists might pick up an audit skillset. A logical outcome would be for audit firms to recruit computer cloud experts and seek to harmonise their skills with those of audit already embedded in the firm. The culture clash between accountants and cloud experts would be a potential side effect from such a strategy.

One tool the accountants have used for decades is the audit trail and this is a phrase already in the cloud computing literature by the National Institute of Standards and Technology (NIST) [6], for example. However, the same phrase may not carry the same meaning in both settings. Quoting from the Oxford English Dictionary (OED) [7]: "(a) Accounting: a means of verifying the detailed transactions underlying any item in an accounting record; (b) Computing: a record of the computing processes that have been applied to a particular set of source data, showing each stage of processing and allowing the original data to be reconstituted; a record of the transactions to which a database or a file has been subjected". So, disparity of definition is recognized by the OED.

Accountants are members of professional bodies (some national, some global) that limit membership to those who have passed exams and achieved sufficient breadth and length of experience that they are deemed worthy to represent the profession. Audit is a key feature of these exam syllabi and the tracing back to the source each accounting activity (the trail) is a foundational aspect of audit.

Whilst NIST [6], gave a clear explanation of an audit trail in a computing security setting and in keeping with the OED definition (b), the use of the term in research in cloud audit seems less precise and consistent. For example, Bernstein [8], sees the trail including: events, logs, and the analysis of these, whilst Chaula [9], gives a longer, more detailed list: raw data, analysis notes, preliminary development and analysis information, processes notes, and so on. Indeed, Pearson and Benameuer [10] accept that the attaining of consistent, meaningful audit trails in the cloud is a goal rather than reality. More worryingly Ko et al. [11], point out that it is quite possible for an audit trail to be deleted along with a cloud instance, meaning no record then remains to trace back, understand and hold users to account for their actions and Ko [12], then details the requirements for accountability. Indeed, the EU Article 29 Working Party [13], highlights poor audit trail processes as one of the security issues inadequately covered by existing principles.

Whilst the audit trail might seem a long and tedious list of activities and interventions, it can be of enormous value in chasing down the root of a cyber-attack, in much the same way as an accountant might use it to trace the steps and individuals involved in enabling an inappropriately authorised payment. At root, the concept should be implemented in a way that it ought even enable the reconstruction of a system were it to have been completely deleted, not just trace an errant one transaction. The audit trail may be duplication, but it is necessary given the risk of manipulation, compromise or loss. Our discussions with IT professionals, who have asserted their confident reliance on data backups, show a level of unmerited trust as an inappropriate intervention will be repeated in every backup until it is discovered. Backups of a corrupted system will not achieve a rebuild to an uncorrupted one the audit trail gives this opportunity. Referring back to Ko et al. [11], establishing an excellent audit trail is worthless if it is only to be deleted along with a cloud instance. The establishment of an adequate audit trail often needs to be explicit as software can allow audit trails to be switched off in its settings.

Once an audit trail has been established, it contents need to be protected from any adjustment. As Anderson [14], points out, even system administrators must not have the power to modify it. Not only is this good practice even with well trained and ethical individuals, but it is always possible that a hacker might be able to attain administrator status. Therefore, the audit trail needs the establishment of an immutable database (i.e., one that only records new activities but never allows adjustment of previous ones). This is the primary goal of this first test for the successful development of a system to preserve both the audit trail and system logs. In the next section, we discuss the motivation for this work.

B. Motivation

Given how easily many enterprises unwittingly make life much easier for attackers, we are motivated to do something about it that should neither be expensive to implement, nor technically challenging. It is obvious from analysis of past successful attacks, that one of the key goals of the attacker is to attack both the audit trail and the system logs, in order to obfuscate, or delete all trace of their visit, and everything that they have done whilst inside the compromised system.

The lack of proper monitoring by enterprises, and the ease with which attackers can carry out this, important for them, exercise also makes it much harder for the enterprise to even know they have been breached, let alone understand what exactly has been read, modified, deleted, or ex-filtrated from their systems. Since this will form a cornerstone of forthcoming legislation, this requirement must be addressed.

We strongly believe that enterprises must make provision to ensure the maintenance of both a proper audit trail, and the preservation of as much forensic evidence as possible. For the reasons already discussed above, they must also take particular note of the need to preserve both audit trail data and systems log data when using the cloud. Thus we now take a look at one of the weakest links in this chain, the database.

The cloud paradigm is essentially web based technology, facilitated by a database back end. There are many well known web based vulnerabilities, yet it is clear from analysis of

security breach reports, that many enterprises are continually failing to implement even the simplest of preventative measures to mitigate these weaknesses. In addition, it is also clear that many enterprises are failing to monitor their systems properly to detect breaches, given the disparity in time between breach and discovery. As far back as 2012, Verizon [15] highlighted the fact that discovery of security breaches often took weeks, months or even years before discovery, with most discovery being advised by external bodies, such as customers, financial institutions or fraud agencies. While improvements have been made in the intervening years, the situation is far from perfect.

Thus it is appropriate to consider the work done by the Open Web Application Security Project (OWASP) carry out a survey every 3 years in which they collate the number of vulnerabilities which have the greatest impact on enterprises. In TABLE I, we can see the top ten list from 2013, 2010 and 2007:

TABLE I. OWASP TOP TEN WEB VULNERABILITIES — 2013 - 2007 [16]

2013	2010	2007	Threat
1	1	2	Injection Attacks
2	3	7	Broken Authentication and Session Management
3	2	1	Cross Site Scripting (XSS)
4	4	4	Insecure Direct Object References
5	6	-	Security Misconfiguration
6	-	-	Sensitive Data Exposure
7	-	-	Missing Function Level Access Control
8	5	5	Cross Site Request Forgery (CSRF)
9	-	-	Using Components with Known Vulnerabilities
10	-	-	Unvalidated Redirects and Forwards

Sitting at the top of the table for 2013, again for 2010, and in second place in 2007, we have injection attacks. It is very clear that enterprises are consistently failing to configure their database systems properly. Injection attacks rely on misconfigured databases used in dynamic web service applications, which allow SQL, OS, or LDAP injection to occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization. This can lead to compromise, or deletion of data held in enterprise databases.

But injection attacks are not the only attacks which involve databases, numbers 3 and 8 also are directly related to either missing input validation or output sanitation. Equally, databases might also be use in most of the other top ten vulnerabilities, which means database mis-configuration, or failure to configure systems which use database systems properly account one way or another for most of the successful attacks.

Attackers continue to use methods which continue to work, which is clear to see from the continued success of the same attacks, year after year. Thus, we consider this area to be of vital importance for ensuring that any enterprise may achieve a high level of security. And given the importance of the audit trail and system log data, we believe the best approach would be to use an immutable database to record this data properly, which we shall discuss in the next Section.

III. What is an Immutable Database?

We can describe an immutable database as a secure database implementation capable of meeting the criteria for a proper audit trail, namely, that it should only be capable of being read by a restricted number of authorised users. It must not permit the editing of any transactions, and must not allow any transaction to be deleted. Only new records can be added, no modifications are permitted, and no deletions may take place, thus preserving the original input for subsequent examination.

Looking at the fundamental requirements of the audit trail in Section II-A, it is clear that a conventional database structure fails to deliver on a number of these requirements. A conventional database structure allows any records to be seen, by anyone authorised, or an attacker able to gain adequate credentials to do so. Worse, there is nothing to prevent modification, or deletion of these records. Thus a conventionally set up database is totally unsuitable for an audit trail. The same argument holds for system logs, which should have the same characteristics as an audit trail.

Thus, an audit trail and system log database must have the same characteristics as the manual system, namely restricted access to view the audit trail, with NO option to add, modify or delete records [2]. Naturally, in a cloud setting, as there may be anything from a single instance up to many thousands of instances running at any given time, it would be sensible to host the logging systems on a completely different server or servers at a location remote from the cloud instances, such that all the instances will have their audit trail and system logging data stored in the remote system. This can reduce the probability that a successful attack on the cloud instance can be leveraged to attack the logging database. Ideally, the logging server or servers should be dedicated entirely to running a secure immutable database, with preferably no direct means of public access.

We accept that this means that the logging database is likely to become a prime target for attack. Thus the logging database should be protected with the highest level of security settings, and should be subject to special monitoring to provide instant warning of any attack.

We made the decision that there would be insufficient time to consider writing bespoke software for our purposes. Thus we would restrict ourselves in this work to evaluating what we could do with an existing system. In [2], we observed that short of writing new bespoke database software, or making serious modifications to existing database software, we would be left with three options we could use to meet our objective:

1) Remove all user access for all users to modifying or deleting records and the database itself;
2) Remove the Modify Record and Delete Record command from the software;
3) Use an Archive Database.

In the next section, we examine the pros and cons of each option, in order to come up with the best practical solution to this problem.

IV. Creating an Immutable Database

Having decided that we would not consider writing some bespoke software, but instead would see how we could configure something utilising existing software, we then evaluated the three options listed in Subsection III.

1) On the positive side, this option is the simplest to configure, does not involve any software modification, and will not impact on software updates. On the negative side, should an attacker gain access to the database and be able to escalate privileges, there would be nothing to prevent them from reversing the restrictions;
2) On the positive side, this option would take away the ability of an attacker, should they get in to the database and be able to escalate privileges, to reverse the restrictions. On the negative side, this could complicate software updates;
3) On the positive side, this presents an extremely simple solution, no software needs modifying, and there is nothing for the attacker to reverse. On the negative side, the Archive Database does not support key searching. This is likely to make searches cumbersome. However. in the short term, we could resolve this issue by extracting a copy of all the data into a conventional database with full key search capabilities for rapid examination.

Thus, we took the view that for the purposes of this work, we would use option 3, using the Archive Database option, in order to create the system logging and audit trail databases. We assume the application database will run using conventional settings, although it is important to take account of the following four weaknesses in conventional systems.

First, default logging options can result in insufficient data being collected for the audit trail. Second, since there is often a lack of recognition that the audit trail data can be accessed by a malicious user gaining root privileges, we recommend the audit trail and system logs should be sent to the external immutable database, set up using the Archive Database configuration, for this purpose. Third, failure to ensure log data is properly collected and moved to permanent storage can lead to loss of audit trail data, either when an instance is shut down, or when it is compromised. Sending all audit trail and system log data to the external immutable database/s will ensure that the data will not be lost when the instance is closed down. Fourth, the recommended mitigation techniques suggested by OWASP should be implemented in the main web application software.

Now, we consider the minimum audit trail data we would wish to collect. MySQL offers the following audit trail options:

- Error log — Problems encountered starting, running, or stopping mysqld;
- General query log — Established client connections and statements received from clients;
- Binary log — Statements that change data (also used for replication);
- Relay log — Data changes received from a replication master server;
- Slow query log — Queries that took more than long_query_time seconds to execute;
- DDL log (metadata log) — Metadata operations performed by Data Definition Language (DDL) statements.

By default, no logs are enabled, except the error log on Windows. Some versions of Linux send the Error log to syslog. Thus for a straightforward implementation, we would wish to collect the Error Log, the General query log, the Binary log and the Slow query log. Where replication is in use, adding the Relay log is recommended. Where DDL statements are used, then the DDL log should also be activated.

While Oracle offer an audit plugin for Enterprise (paid) editions of MySQL, which allows a range of events to be logged, by default most are not enabled. The MariaDB company, whose author originally wrote MySQL, have their own open source audit plug-in, and offer a version suitable for MySQL. It has the following functionality:

- CONNECTION — Logs connects, disconnects and failed connects (including the error code);
- QUERY — Queries issued and their results (in plain text), including failed queries due to syntax or permission errors;
- TABLE — Which tables were affected by query execution;
- QUERY_DDL — Works as the ‘QUERY’ value, but filters only DDL-type queries (CREATE, ALTER, etc);
- QUERY_DML — Works as the ‘QUERY’ value, but filters only Data Manipulation Language (DML) DML-type queries (INSERT, UPDATE, etc.).

Where a company falls under the provisions of the new EU GDPR regulations, using the MariaDB audit trail plug-in and turning on ALL 5 logging options would be a prudent move. Admittedly this would require a considerable increase in storage requirements for the log output. However, since they would then be in a position to provide full disclosure to the regulator of all records accessed, tampered with or deleted, this would go a very long way to mitigate the amount of fine they might be subject to, which could be as high as 4% of their global turnover.

Thus, this approach will address the first problem, that of insufficient audit trail and system logging data being collected. If the data is sent to a well protected external database, an attacker who has compromised the running instance will not be able to cover their trail. The system logs could be retained on the instance to make the attacker think that they have covered their tracks. Thus, the second point is addressed. By sending a copy of all log data to the secure immutable database, we can address the third point, thus ensuring no data is lost on shut down of the instance. Finally, if the OWASP mitigation techniques are used to harden the web application, there will be less likelihood of a successful breach taking place. Plus the immutable database on the secure external server satisfies the

requirements of a proper audit trail [14].

There is also no doubt that adding an Intrusion Detection system (IDS) is also a useful additional precaution to take, and again, this should be run on an independent secure server under the control of the cloud user.

Equally, where the MySQL instance forms part of a LAMP server, then it would also be prudent to make some elementary security changes to the setup of the Linux operating system, the Apache web server, and to harden the PHP installation.

There is one additional task that would be very worthwhile. That is to set up an additional control instance to monitor every new instance added to the application, which regularly checks whether the instance is still functioning as expected. This would allow this system to warn of instances unexpectedly being closed down, which might be a sign of an attack. In addition, the log files in the immutable database could be monitored for specific patterns, which might indicate the possibility of an attack.

One of the biggest issues is the fact that there is such a lag between breach and discovery, and this approach could provide much earlier warning of such an event. However, of greater interest, is the fact that a full forensic trail would be instantly available for immediate investigation. And it would be possible to disclose the extent of the breach well within the required disclosure time of 72 hours from the time of breach to disclosure.

As we see from [17], see Figure 1, that in 2015, 75% of breaches happened within days, yet only 25% of discoveries are actually made within the same time-frame. This still leaves a large gap where compromised systems may still be under the control of malicious users. Our proposed approach would go some way to reducing this problem.

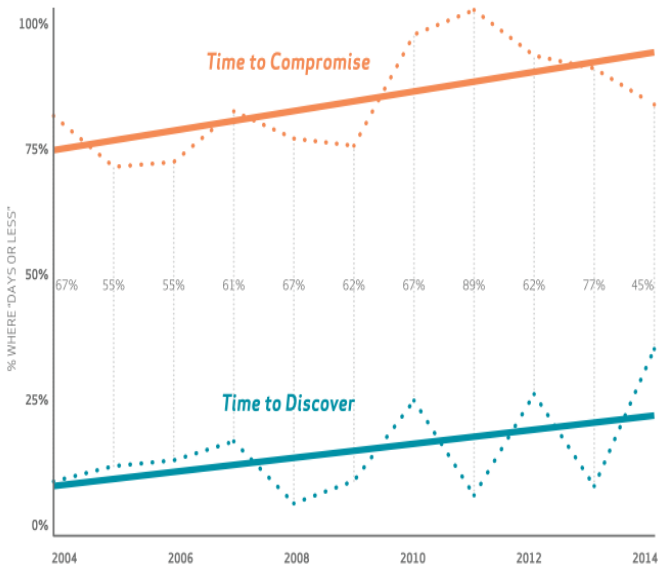

Fig. 1. The Lag Between Breach and Discovery © 2015 Verizon

This presents a clear indication that very few firms are actually scrutinising their server logs. We take a quick look at some typical database attacks and possible mitigation for these attacks in the next Section.

V. Typical Database Attack Methodologies

SQL injection attacks are relatively straightforward to defend against. OWASP provide an SQL injection prevention cheat sheet [18], in which they suggest a number of defences:

- Use of Prepared Statements (Parameterized Queries);
- Use of Stored Procedures;
- Escaping all User Supplied Input;

They also suggest that companies should enforce least privilege and perform white list input validation as useful additional precautions to take.

For operating system injection flaws, they also have a cheat sheet [19], which suggests that LDAP injection attacks are common due to two factors, namely the lack of safer, parameterized LDAP query interfaces, and the widespread use of LDAP to authenticate users to systems. Their recommendations for suitable defences are:

- Rule 1 Perform proper input validation;
- Rule 2 Use a safe API;
- Rule 3 Contextually escape user data.

And for LDAP system injection flaws, their cheat sheet [20], recommends the following injection prevention rules:

- Defence Option 1: Escape all variables using the right LDAP encoding function;
- Defence Option 2: Use Frameworks that Automatically Protect from LDAP Injection.

None of these preventative measures suggested by OWASP are particularly difficult to implement, yet judging by the recurring success of these simple attacks, companies are clearly failing to take even simple actions to protect against them.

Thus, in addition to making the simple suggestions we propose above, cloud users should also make sure they actually review the audit trail logs. It is vital to be able to understand when a security breach has occurred, and exactly which records have been accessed, compromised or stolen. While recognising that this is not a foolproof method of achieving cloud security, it is likely to present a far higher level of affordable, achievable security than many companies currently achieve.

Implementing these suggestions will not guarantee security, but will make life so much more difficult for the attacker that they are more likely to move on to easier 'low hanging fruit' elsewhere. There is currently an abundance of other options for them to choose from.

However, the company must remain vigilant at all times. It would be prudent to subscribe to security feeds, and follow leaders in the field to ensure they remain aware of all the latest security vulnerabilities and exploits. Of course, companies must realise that the threat environment is not restricted to outside parties alone. A greater concern is the threat posed by malicious internal actors, which can be even more serious where they act in concert with outside parties. This presents one of the most serious weaknesses to the security of a company. Equally, laziness on the part of staff or lack of

knowledge, particularly where they have not been regularly trained to provide them with full awareness of all the latest threats, including social engineering attacks, and the consequence of falling victim to them, can also pose an extremely serious risk to company security.

In the event of a security breach, not if, but rather when it happens, it may be necessary to conduct a forensic examination to establish how the company defences were breached. With traditional distributed systems, there is usually something for the forensic computer scientists to find, somewhere in the system. They are completely accustomed to dealing with being able to find only partial traces of events, from which they can build a forensic picture of the breach. This becomes more problematic the longer the time between breach and discovery.

However, once a company adopts cloud use, this becomes far more problematic. While forensic computer scientists can work wonders with a range of partial discoveries, deleted or otherwise, once a cloud instance is shut down, there is virtually zero chance of regaining access to the shut down system. The disk space used by that system could be re-used, literally within seconds, and where the time interval between breach and discovery is considerably longer, as is generally the norm, then this opportunity becomes a physical impossibility. Thus, for forensic purposes, companies need to pay far more attention to what is actually going on in the cloud.

The suggestions we make can go a long way to providing a greater level of security, and perhaps more importantly, can ensure there is actually a forensic trail to follow in the event of a breach.

VI. Conclusion

We have considered a wide range of security issues in cloud based systems, with a view to highlighting that the attack surface of any cloud based system extends well beyond technical issues. We have identified that databases present a considerable weakness in cloud based systems, in addition to the unintended potential loss of forensic data caused by the manner in which scalability is handled in large cloud systems.

We have suggested a simple approach that could be easily implemented, with minimal technical knowledge, which would offer a considerable improvement on cloud security, with the additional benefit of maintaining a vastly improved forensic trail to explore in the event of a breach. Equally, our proposal also offers the benefit of being able to discover precisely which records have been viewed, compromised, or deleted. This presents a significant mitigation in the event that any regulator proposes a significant fine, since the company will be in a position to comply fully with the reporting requirements.

We plan to test this proposal to identify any loss in performance resulting from not being able to use key searching in the immutable databases, and to identify how it will stand up to attack. In the longer term, it would be useful to develop a software solution that might add the key search capability to the immutable database.

References

[1] B. Duncan and M. Whittington, "Enhancing cloud security and privacy: The cloud audit problem," in *Cloud Comput. 2016 Seventh Int. Conf. Cloud Comput. GRIDs, Virtualization*. Rome: IEEE, 2016, pp. 119–124.

[2] B. Duncan and M. Whittington, "Enhancing cloud security and privacy: The Power and the weakness of the audit trail," in *Cloud Comput. 2016 Seventh Int. Conf. Cloud Comput. GRIDs, Virtualization*. Rome: IEEE, 2016, pp. 125–130.

[3] B. Duncan and M. Whittington, "Cloud cyber-security: Empowering the audit trail," *Forthcom. Int. J. Adv. Secur.*, vol. v9, no. 3&4, p. 15, 2016.

[4] Trustwave, "2012 Global Security Report," Tech. Rep., 2012.

[5] 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.

[6] B. Guttman and E. A. Roback, "NIST special publication 800-12. An introduction to computer security: The NIST Handbook," NIST, Tech. Rep. 800, 2011. [Online]. Available: csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf Last Accessed: Jan 2017

[7] OED, "Oxford English Dictionary," 1989. [Online]. Available: www.oed.com

[8] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the intercloud - Protocols and formats for cloud computing interoperability," in *Proc. 2009 4th Int. Conf. Internet Web Appl. Serv. ICIW 2009*, 2009, pp. 328–336.

[9] J. A. Chaula, "A socio-technical analysis of information systems security assurance: A case study for effective assurance," Ph.D. dissertation, 2006. [Online]. Available: http://scholar.google.com/scholar?hl=en{\&}btnG=Search{\&}q=intitle:A+Socio-Technical+Analysis+of+Information+Systems+Security+Assurance+A+Case+Study+for+Effective+Assurance{\#}1 Last Accessed: Jan 2017

[10] S. Pearson and A. Benameur, "Privacy, security and trust issues arising from cloud computing," in *2010 IEEE Second Int. Conf. Cloud Comput. Technol. Sci.*, no. December. Ieee, nov 2010, pp. 693–702.

[11] R. K. L. Ko et al., "TrustCloud: A framework for accountability and trust in cloud computing," *Proc. - 2011 IEEE World Congr. Serv. Serv. 2011*, pp. 584–588, 2011.

[12] L. F. B. Soares, D. a. B. Fernandes, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, "Security, privacy and trust in cloud systems," in *Secur. Priv. Trust Cloud Syst.* Springer, 2014, ch. Data Accou, pp. 3–44.

[13] EU, "Unleashing the potential of cloud computing in europe," 2012. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=SWD:2012:0271:FIN:EN:PDF Last Accessed: Jan 2017

[14] R. J. Anderson, *Security engineering: A guide to building dependable distributed systems*, C. A. Long, Ed. Wiley, 2008, vol. 50, no. 5.

[15] Verizon, N. High, T. Crime, I. Reporting, and I. S. Service, "2012 data breach investigations report," Verizon, Tech. Rep., 2012.

[16] OWASP, "OWASP top ten vulnerabilities 2013," 2013. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project Last Accessed: Jan 2017

[17] Verizon, "Verizon 2015 data breach investigation report," Tech. Rep., 2015.

[18] OWASP, "OWASP SQL injection cheat sheet," 2016. [Online]. Available: https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet Last Accessed: Jan 2017

[19] OWASP, "OWASP injection prevention cheat sheet," 2016. [Online]. Available: https://www.owasp.org/index.php/Injection_Prevention_Cheat_Sheet Last Accessed: Jan 2017

[20] OWASP, "OWASP LDAP injection prevention cheat sheet," 2016. [Online]. Available: https://www.owasp.org/index.php/LDAP_Injection_Prevention_Cheat_Sheet Last Accessed: Jan 2017