Creating and Configuring 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 and one of the main components, the database, is frequently configured incorrectly, frequently using default settings, thus leaving the system wide open to attack. Once a system has been attacked, valuable audit trail and system log data is usually deleted by the intruder to cover their tracks. Considering the average industry time between breach and discovery, there is often little or no forensic trail left to follow. While this presents a significant challenge to these conventional systems, when such a system uses cloud computing, the challenge increases considerably. In a conventional setting, the enterprise can use a robust firewall to afford some protection to enterprise users, however in a cloud setting, the enterprise firewall will not extend to external services, and a lot more people than are often considered can have access to cloud resources. Of equal importance is that in cloud settings, where new instances may be automatically spooled up and shut down to follow the demand curve, any data stored on the running instance before shut down will be irretrievably 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 usage of modern databases, and while there are a number of mitigating strategies that can be deployed, they seem not to be sufficient, as evidenced by their continual recurrence on annual security breach report lists. This failure to take even simple, inexpensive measures to try to mitigate the problem is tantamount to aiding and abetting the intruders. Invariably, once embedded in a system, the first goal of the intruder is usually to delete the forensic trail to eliminate all sign of their intrusion. Duncan and Whittington [1] proposed an interim solution to try to address this problem, and this paper extends that previous work.

Duncan and Whittington [2] have written about the difficulties surrounding proper audit of cloud based systems. They talked about the need for enterprises to maintain a proper audit trail in their systems, and about the weaknesses arising from poor configuration of databases, particularly in the context of cloud systems [3]. 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 [4]. They used this approach to develop a proposal for using an immutable database system to log audit trail data and forensic system data [1]. The main idea here was to start with a simple, easy to configure system utilising existing technology to assist in resolving this challenging security requirement, with the intention of adapting the use of this idea to address the as yet unresolved issue of retaining cloud audit trail and forensic data.

As today's corporate enterprises evolve, there is an ever changing move to develop more and more complex software. This presents a considerable challenge in the development of ever more complex software systems, especially for configuration, because security has traditionally taken a back seat to the functionality of the software programmes. The vast majority of software and software development tools have their origin in a time when software security was not a major concern. Many of these pieces of software were developed when little thought was given to security. Before the internet took off, security was just a case of keeping the computer under lock and key. However, the internet changed all that in a big way. Anyone with access to the internet could then access any enterprise system connected to the internet, and with sufficient ingenuity, could gain access to poorly secured enterprise systems.

This is when poorly written software would come back to haunt enterprises. Few people thought about potential vulnerabilities in operating systems, or software systems — excepting potential attackers. A huge industry sprung up around finding vulnerabilities in operating systems and software systems that could be exploited. Many standards set for operating systems, software systems, systems such as email systems go back decades, back to the pre-internet days. By adding more and more complex software onto already vulnerable operating systems and software systems, the security problems are compounded. To this, we can add the lack of robustness of approach to analysing audit trail and server logs.

Some five years ago in 2012, Trustwave [5] 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 reduced, but nevertheless remains a concern, particularly in the light of forthcoming legislation, such as the EU General Data Protection Regulation (GDPR) [6]. Looking at the latest security breach reports, the average time between breach and discovery is still in the range of several weeks to months, meaning that 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.

Thus, we can see that many enterprises are adding ever more complex software on top of already weak and vulnerable systems, are often failing to analyse server logs properly, and are failing to effectively configure ever more complex systems securely, leading to an inability to understand when they have been breached.

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 explain the detailed mechanics of how to create and configure a secure immutable database server on which to host our proposed system. In Section VII, we discuss weaknesses, how to mitigate them, and how to move forward to provide further improved levels of security in order to minimise the possibility for attackers to succeed in any attack on this valuable resource. In Section VIII, we discuss our conclusions and future work.

II. Background and Motivation

In the early years of enterprise computing, a mainframe computer was used to process all the enterprise's information needs. Of course, this option was only open to the largest enterprises. As computer systems evolved, following the prediction of Moore's Law [7], this computing model also evolved, opening up more opportunities for ever smaller organisations to take advantage of the benefits offered by computerising their information and process systems. Once the internet arrived, opportunities increased significantly, but this brought with it additional exposure to the risks of poor security, traditionally an area given little thought.

Security practices started to evolve to try to keep up with this changing business environment, including the development of sophisticated enterprise firewalls. The development of new paradigms such as mobile computing, Bring Your Own Device (BYOD) and cloud computing, started to offer massive new opportunities, yet the increased risks associated with these practices were slow to be addressed. Assumptions such as that enterprise firewalls would protect all enterprise data, including on mobile computing, BYOD and cloud systems were erroneous. When cloud computing enabled the Internet of Things (IoT) and Big Data to gain huge traction, these erroneous assumptions continued, without considering the further increase in risks brought by the many inherent weaknesses introduced by this new technology.

As the business environment is constantly changing, so are corporate governance rules and this would clearly imply changing security measures are needed to keep up to date. Many managers are unable, unwilling or unsure of how to define proper security goals [8] [9] [10]. With more emphasis being placed on responsibility and accountability [10]–[14], social conscience [15]–[17], sustainability [18]–[22], resilience [23]–[29] ethics [17], [30]–[32] and Corporate Social Responsibility (CSR) [33]–[40], there is a need to consider more than the traditional security requirements of Confidentiality, Integrity and Availability (CIA).

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 (SRE) [41]. Thus expanding security requirements can not only help address some of the shortcomings of agency theory, but can also provide a perfect fit to stewardship theory. Stewardship carries a broader acceptance of responsibility than the self-interest embedded in agency. This breadth extends to encourage stewards to act in the interests of enterprise owners as well as society and the environment as a whole [42]. Broadening the definition of security goals provides a more effective means of achieving a successful cloud audit, although the additional complexity cloud brings will potentially complicate the audit trail.

A fundamental issue with anything cloud related is that while the software being used works well on their in-house systems, it will not necessarily be as secure when running on cloud, since enterprise firewalls will no longer provide the protection that enterprises traditionally relied on. Often, enterprises fail to realise just how many people may have access to their data in cloud based systems. While Cloud Service Providers (CSP)s often vet their staff to exacting standards, often their temporary staff providers do not. A favourite trick of attackers is to have one of their team be employed in a CSP's datacenter in order to have better access to many potential targets. Where a risk is identified, it can be quantified and properly addressed or mitigated. Whereas, an unrecognised risk can pose a very serious threat to an enterprise.

Often enterprises simply load their secure enterprise software onto cloud systems and assume they will still be secure. While the software may very well run in a functional way, the enterprise can not be assured that these systems will run securely. A major cloud issue which has yet to be resolved [43]–[45] is that once an attacker breaches a cloud system, there is nothing to stop them adjusting or deleting both the

audit trail and the forensic trail of such systems. A less obvious weakness arises when systems are automatically scaled up, and down, to meet demand. Often, these systems assiduously collect server log data, including audit and forensic trail data, but fail to record this data securely elsewhere, meaning that as each instance is shut down to match falls in demand, these records are lost for ever [46].

In this paper, we use the MySQL relational database management system (RDBMS) to illustrate what is currently possible. While not all databases are identical, many exhibit similar weaknesses, often arising through improper configuration. In the next stage of our research, we will compare and test a number of SQL, NoSQL and NewSQL systems to gain a better understanding of how well each might perform for our purposes. MySQL, a RDBMS, has long been the most popular database globally, powering large scale websites such as Google, Facebook and Twitter, no doubt helped by its open source nature. The community is very well defined.

NoSQL, on the other hand, does not use SQL and can be considered a non-relational database, meaning it is table-less, the thought being it will be easier to manage. It also offers higher flexibility, newer data models, is mostly open source and low cost, offering scalability through support for Map Reduce, with no need for detailed database models. On the other hand, the community is not well defined, it is lacking in user tools, both for analysis and performance testing, and lacks standardization as well as not complying with Atomicity, Consistency, Isolation, and Durability (ACID), but instead relying on complying with **B**asically **A**vailable, **S**oft state, **E**ventual consistency (BASE). This is likely to be a major barrier to overcome when considering the importance of ACID compliance for both the audit and forensic trails.

NewSQL, on the other hand, tries to bridge the gap between SQL systems and NoSQL systems, offering to combine the ACID guarantees of SQL with the scalability and high performance of NoSQL. Again, being a relatively young technology, is suffers from many of the drawbacks of NoSQL, but does at least offer ACID compliance.

Clearly, there will be benefits and drawbacks in the case of each different database offering, and it will be necessary to clearly identify the specific details of which will offer the best utility for our purposes.

Often, 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 [47].

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 firms 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 [1].

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) [48] for example. However, the same phrase may not carry the same meaning in both settings. Quoting from the Oxford English Dictionary (OED) [49]: "(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 [48] 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 [50] sees the trail including: events, logs, and the analysis of these, whilst Chaula [51] gives a longer, more detailed list: raw data, analysis notes, preliminary development and analysis information, processes notes, and so on. Indeed, Pearson and Benameuer [52] accept that the attaining of consistent, meaningful audit trails in the cloud is a goal rather than reality. More worryingly Ko et al. [46] 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 [53] then details the requirements for accountability. Indeed, the EU Article 29 Working Party [54] 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 to enable the reconstruction of a system were it to have been completely deleted, not just trace an errant single 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. [46] 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 frequently allows 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 [55] 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 the forthcoming EU GDPR, this requirement must be addressed.

Why should this be of concern? The EU GDPR has some serious teeth. Failure to report a breach within 72 hours will be a contravention, as will failure to take proper steps to protect data assets. There are serious penalties that can be enforced. A single data breach can result in a fine of up to the greater of €10 million or 2% of Global Turnover based on the previous year's accounts. Multiple breach elements can result in the fine increasing to the greater of €20 million or 4% of Global Turnover based on the previous year's accounts.

That is sure to catch the attention of enterprises, particularly in line with the current industry standard time between breach and discovery. Given that the enforcement date of the EU GDPR is 25th May 2018, and that enterprises have yet to get the time between breach and discovery down to hours, let alone days, this has to be concerning. Those enterprises who are UK based, will also have no respite, as the UK have agreed to implement the EU GDPR and continue with it after Brexit. Indeed, they propose additional changes to give users greater rights.

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. Users who do not, are effectively aiding and abetting attackers. 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 [56] 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.

It is also appropriate to consider the work done by the Open Web Application Security Project (OWASP), who carry out a survey around every 3 years in which they collate the number of vulnerabilities which have the greatest impact on enterprises globally. In TABLE I, we can see the top ten lists from 2017, 2013, 2010 and 2007:

Sitting at the top of the table for 2017, 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 management systems properly. Injection attacks rely on mis-configured 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.

TABLE I. OWASP TOP TEN WEB VULNERABILITIES — 2017 - 2007 [57]

2017	2013	2010	2007	Threat
A1	A1	A1	A2	Injection Attacks
A2	A2	A3	A7	Broken Authentication and Session Management
A3	A3	A2	A1	Cross Site Scripting (XSS)
A4	A7	-	-	Missing Function Level Broken Access Control
A5	A5	A6	-	Security Misconfiguration
A6	A6	-	-	Sensitive Data Exposure
A7	-	-	-	Insufficient Attack Protection
A8	A8	A5	A5	Cross Site Request Forgery (CSRF)
A9	A9	-	-	Using Components with Known Vulnerabilities
A10	A10	-	-	Unprotected APIs

But injection attacks are not the only attacks which involve databases, numbers A3 and A8 in the 2017 column also are directly related to either missing input validation or output sanitation. Equally, databases might also be used in most of the other top ten vulnerabilities, which means database mis-configuration, failure to carry out proper input validation 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. Indeed the top three attacks have been around for over a decade. 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.

There are many ways that we might approach developing an immutable database beyond the MySQL route. We could use new database technology such as NoSQL [58], [59] and NewSQL [60], or we could take the blockchain/bitcoin approach [61]–[63]. These approaches do show some promise, but are out of scope for this current paper, which concentrates on a pragmatic and simple approach. We do, however, include them for consideration in our future work as outlined in Section VIII.

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 [3]. 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 [3], 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 Section 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 an enterprise 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 [55].

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 [64], 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.

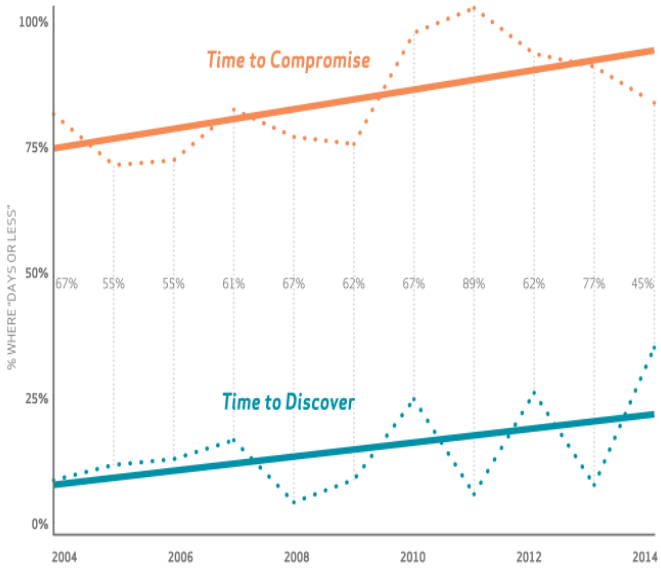

Fig. 1. The Lag Between Breach and Discovery © 2015 Verizon

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 [65], 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 enterprises 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 [66], 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 [67] 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.

These preventative measures suggested by OWASP are not particularly difficult to implement, yet judging by the recurring success of these simple attacks year after year after year, enterprises are clearly failing to take even the simplest of actions to protect themselves against them.

In addition to making the simple suggestions we propose above, cloud users should also make sure they actually review the audit trail logs. IF you do not review the logs, how will you know whether you have been breached? 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 we recognise 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 enterprises 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 enterprise 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, enterprises 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 an enterprise. 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 enterprise 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 enterprise 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 an enterprise 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, enterprises 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. Creating and Configuring Securely an Immutable Database System

From Section IV, we can see how to create an immutable database. We do not want to install this in the same cloud system we are trying to protect, as this would leave the immutable database open to direct attack by the successful intruder. Rather, we would wish to place this into a dedicated server, preferably installed in a secure system under the control of the enterprise. However, in some circumstances, it may be necessary to run the immutable database in a cloud system, and in this case, we strenuously recommend that a different CSP is chosen. Our preference is, of course, for an in-house dedicated secure server, so we shall start by outlining the requirements for that system first. Later in this section, we will consider what special measures might need to be taken for setting up this system in a cloud environment, and we finish off with a comparison between the two options.

A. The In-House Secure Server

This server should be placed behind the enterprise firewall and an Intrusion Detection System (IDS). There should be no direct external web access to this system. There should be no external login to a shell allowed to this system. It is necessary to remove as many toys as possible from the attacker to limit the scope for attack. Clearly, once the attacker discovers the presence of this system, it is likely to become a prime target for attack. Thus we must remove as many routes in as we possibly can to this system. Direct web access is an attacker's dream. Removing this option makes life far more difficult for the attacker, and that is precisely what we want to achieve.

This server should have no wireless components attached, especially for connection to the network, as wireless can be readily subject to attack. For a paranoid approach, the server can be placed inside a locked room, with keyboard, video screen and mouse removed from the server and stored in a locked cabinet installed for the purpose, meaning it will then be physically impossible to interact with the server. The key should not then be available to the system administrator for this server. Only collected data from the cloud source will be allowed in through the hard wired internet connection. The bandwidth and speed of this connection will have to be more than adequate to service the projected needs of the required data flow. Also, the server will require to have sufficient performance and permanent storage for the collected data that will require to be stored over time.

When installing the operating system for this server, the operating system software must be analysed and ALL unneeded software should be removed. Similarly, only the immutable database software should be installed, with no other software installed on this system. All open ports must be closed, both on the server and on the network configuration. The immutable database server systems administrator should not be granted any privileges on the immutable database. The administrator for the immutable database should not be granted root access for the immutable database either. Once the immutable database has been set up by a user who is granted root privilege through a dongle to be inserted solely for that purpose, the dongle and the access credentials should also be securely locked away in the secure cabinet. Access to the cabinet should be through two members of senior management, with their keys securely stored elsewhere. In Unix based systems, Cron is a time-based job scheduler in an operating system, designed to carry out specific tasks at specific times. The Cron can handle a multiplicity of commands (or shell scripts) over time, thus ensuring the right tasks are carried out at the right time. Thus maintenance routines can be set as Cron jobs to run tasks which can be performed by the server itself at fixed times, dates, or intervals.

Server software updates can either be set to operate automatically, or can be done under controlled conditions by the system administrator. Similarly, database updates can either be set to operate automatically, or can be done under controlled conditions by the database administrator.

For a super paranoid approach, this system can be replicated elsewhere, and the data mirrored as it is streamed, whereby it is operated under the same conditions with no physical access for anyone involved in the system that is being protected.

For extreme levels of paranoia, Write Once Read Many (WORM) times hard drives might be used. This is already well established technology for CD disks, DVD disks and RDX disks. These are usually considered too slow for enterprise use. While conventional hard disks are available to use in a WORM high security Network Attached Storage (NAS) configuration, they are still expensive and not super fast yet.

We earlier mentioned that this immutable database server could be configured to operate in the cloud, and in the next sub-section we note how this can be achieved, bearing in mind that so doing will introduce additional vulnerabilities.

B. The Cloud Based Immutable Database Server

The first point to stress with a cloud based immutable database server, is that it will be considerably less secure than the in-house version. This is due to the inherently less secure nature of cloud technology, which is why we are attempting to resolve this problem in the first place. On the plus side, it will be considerably more pragmatic in use, since it will be

impossible to lock up the server and remove keyboard, video screen and mouse.

Taking all of that into account, on the plus side, the instance will be capable of scaling easily to meet demand. On the negative side, it will be much easier to attack. However, since it will not have direct web access, this will present a much greater challenge to the attacker. Also, it will not be set up with the same CSP as the main system, which means unless the intruder gets far enough into the main system, it will take some figuring out. But it would not be impossible, and therefore the immutable database will become a very promising target.

Everything that can be done in Sub-Section VI-A, with the exception of the physical tasks can be carried out in this case. Careful setup of who is allowed to access this system can help control who can gain access to the system, and with no direct external web access to this system and only tightly restricted login to a shell available, this will make the attacker's job much more difficult. But, you must always remember it is running on a cloud system, and is therefore subject to the same pitfalls as the main system you are trying to protect.

This means that in addition to IDS systems, you will also require to have a seriously good monitoring system in place. It will be vital to understand who is in your system and what they are trying to do. In fact, let us re-phrase that. Other than receiving the data you expect — anyone inside your system is an intruder, so you need to have instant warning, bells ringing, lights flashing, klaxons blaring — whatever it takes, in order that the intruder can instantly be dealt with.

C. A Comparison Between the Two Options

Before attempting to make a decision between the two options, we must first consider the pros and cons of each system.

For the in-house immutable database system option:

Pros:

- In-house will be more secure than cloud-based;
- In-house offers the advantage of extra physical security;
- In-house will be fully under the control of the enterprise;
- In-house will gain protection from enterprise firewall and IDS;
- In-house system will benefit from not requiring external web access;
- In-house system will benefit from no wireless access.

Cons:

- Lead time for implementation and expansion increases can be a factor;
- Insufficient internet bandwidth could adversely impact on performance;
- In-house costs may be greater than for cloud-based systems;
- In-house system will become a highly attractive target.

For the cloud-based option:

Pros:

- Cloud-based systems are simple to implement and can be rapidly deployed;
- Cloud-based systems respond well to changes in demand;
- Cloud-based systems cope well with increased volumes of data storage.

Cons:

- Cloud-based systems will be less secure than in-house;
- Cloud-based systems will become a highly attractive target;
- Cloud-based systems will be easier to attack than in-house systems;
- Cloud-based systems will need to ensure that all data from closed down instances are permanently stored.

Thus it is clear that compromises will have to be made depending on which route is chosen to store the data collected into the immutable database. However, being able to understand the pros and cons of each option provides a good basis on which to evaluate the impact of either on the enterprise, thus leading to the right decision for the enterprise.

Regardless of which system is chosen, either will require the installation of a good monitoring system. In the next subsection, we consider the requirements for a suitable monitoring system.

D. Monitoring the Immutable Database Server

The data contained in the main system is very valuable to an enterprise. The data collected and contained in the immutable database is also extremely valuable to both the enterprise and to law enforcement. Under conventional attack scenarios, the audit trail and forensic trail are usually modified or deleted by the intruder, in order to cover their trail. Without proper audit trail or forensic data, it becomes very difficult to understand what records have been accessed, modified or deleted. When this concerns data covered by the EU GDPR, this brings a serious problem to bear on the enterprise — the potential impact of fines. Thus the audit trail and forensic trail data captured in this remote server becomes an especially useful resource for the enterprise.

With conventional successful cloud systems attacks , the forensic trail is usually obliterated, or partially destroyed by the intruder, and this will be the approach for the successful attack on the main system. Very few intruders will be skilled enough to understand that there is a secret cache of forensic data. However, if an intruder is skilled enough to realise that this is the case, then they most certainly will come after the audit and forensic data, and will attempt to discover where it is and attack that system. The setup of this system means this will present a much greater challenge, which will defeat all but the most skilled intruder. This is why it is vital to have a successful monitoring system in place.

It would be sensible to use a software agent to monitor, and log, all system calls made inside the immutable database server system. Any system call made other than the writing of data to the immutable database is likely to arise from the unexpected actions of an intruder. So by monitoring and looking for system calls that do not match the expected pattern, then this provides evidence of a possible intruder in the system.

Naturally, it also makes sound sense to monitor the incoming data from the main system being protected by the

immutable database. It would make sense to create another software agent, or agents, do handle these tasks also. The agents could be used to scan the incoming data to search for know patterns suggestive of the presence of an intruder. This can provide a second line of defence in the event that such agents in the main system had been knocked out by the intruders.

Monitoring of these systems is vital in order to be able to realise the moment an intruder breaches the system, particularly in the light of the stringent reporting requirements of the forthcoming EU GDPR.

VII. Discussion on Security Issues

In our quest to secure cloud based systems in the light of the forthcoming EU GDPR, we need to face facts. Achieving any kind of security in IT systems at this time is akin to trying to perform all one's daily tasks with one's hands tied behind one's back. The combination of the requirement for legacy compatibility, poor inherent security of software due to bug riddled software and insufficient testing, both the operating systems and for the ever more complex software running on these systems, coupled with insufficient understanding of how to configure all these products securely, means that there is little prospect of a successful outcome.

Also, many standards for various software implementations were developed decades ago, long before the internet opened up every user to exploitation due to non-existent or limited security. Decades of limited software testing are opening up ever more vulnerabilities for attackers to exploit. The insistence on backward compatibility of software products is a case in point. Adding a more complex system on top of an already vulnerable system is simply a recipe for disaster.

What is needed is a recognition that we are collectively going about this the wrong way. In software development, the reuse of software is a laudable software engineering goal. But the reuse of inherently insecure software systems simply perpetuates the problem. This is why we have weaknesses in operating systems, database systems, web systems, network systems, email systems, indeed pretty much all our current software systems.

A new approach is required, whereby all software systems are re-written from the ground up — to be secure. A good start would be to enforce the writing of proper secure software systems, APIs, DLLs and drivers for all new hardware being produced. A revision of email and network protocols would provide a useful improvement to reduce delivery of attack vectors for attackers. Operating systems and all other software in general should be re-written in a much more secure way. Default configuration should be "super secure", so that every software installation will be guaranteed secure. Detailed security configuration instructions should be provided with all software, to minimise the effect of mis-configuration opening up unexpected vulnerabilities.

It is comforting to note that many operating system developers have started initiatives to develop secure operating systems. Over recent years, it is clear that a lot of work has gone into this effort, but it is equally clear that it may be some time before we see a fully secure operating system available for use. Equally, many software development businesses have also started similar initiatives, which is also very welcome. Again, it may be some time before we see the full fruits of these initiatives. Solving the major cloud issue of how easily cloud audit and forensic data can be deleted remains a serious concern.

Initiatives, such as the Bright Internet [68], are also very welcome as a means of providing greater accountability by all internet service companies and users. The status quo can not continue. Last year, the global cost of cybercrime is estimated to have exceeded global income from illicit drugs for the first time. As long as the status quo remains, the impact of cybercrime will continue to climb. Add to that the cost of potential fines arising from penalties arising from cyber breaches around the globe, and it is clear that something positive needs to happen.

VIII. Conclusion and Future Work

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.

It is common for experts to recommend simple housekeeping solutions when security vendors want to sell new technology. This proposed solution is more of the former, though development of audit trail interrogation tools would help meet the tight deadlines for discovery and rectification in the GDPR. A solution based on an immutable database of an audit trail may seem a very boring and low-tech solution, but since high level technological solutions have yet to be able to resolve this very important weakness, it represents a pragmatic short term approach to addressing a serious problem with cloud. As if that were not enough to get some attention, 4% of turnover fines ought to focus minds, even if protecting customer and employee data doesn't.

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. Until such time as this major cloud weakness can be properly resolved, this proposal offers an interim mitigating solution.

Equally, our proposal also offers the benefit of being able to discover precisely which records have been viewed, compromised, or deleted. This presents a means of ensuring compliance with the GDPR, which is likely to offer a significant mitigation in the event that any regulator proposes a significant fine, since the enterprise 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 database, 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.

However, as a follow through to the limited work contained in this paper, we will extend our view to include NoSQL, NewSQL and blockchain technology. We will also consider several methodologies for security the immutable database so that it might be run securely on cloud, including the use of a Unikernel solution based on UnikernelOS software. This might provide an interesting synergy for security due to both the ultra small profile that UnikernelOS offers together with the immutability of running instances.

REFERENCES

[1] B. Duncan and M. Whittington, "Creating an Immutable Database for Secure Cloud Audit Trail and System Logging," in *Cloud Comput. 2017 Eighth Int. Conf. Cloud Comput. GRIDs, Virtualization*. Athens: IARIA, ISBN: 978-1-61208-529-6, 2017, pp. 54–59.

[2] 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*, April 2016, pp. 119-124.

[3] 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*, April 2016, pp. 125-130.

[4] B. Duncan and M. Whittington, "Cloud cyber-security: Empowering the audit trail," *Int. J. Adv. Secur.*, vol. 9, no. 3 & 4, pp. 169–183, 2016.

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

[6] EU, "EU General Data Protection Regulation (GDPR)," 2017. [Online]. Available: http://www.eugdpr.org/ Last accessed: 28 August 2017.

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

[8] N. Papanikolaou, S. Pearson, M. C. Mont, and R. K. L. Ko, "Towards Greater Accountability in Cloud Computing through Natural-Language Analysis and Automated Policy Enforcement," *Engineering*, pp. 1–4, 2011.

[9] A. Baldwin, D. Pym, and S. Shiu, "Enterprise Information Risk Management: Dealing with Cloud Computing," *Abdn.Ac.Uk*, pp. 257–291, 2013. [Online]. Available: http://link.springer.com/10.1007/978-1-4471-4189-1{_}8 Last accessed: 30 November 2017.

[10] B. Duncan and M. Whittington, "Enhancing Cloud Security and Privacy: Broadening the Service Level Agreement," in *14th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. (IEEE Trust.*, Helsinki, Finland, 2015, pp. 1088–1093.

[11] 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.

[12] S. Pearson, "Toward accountability in the cloud," *IEEE Internet Comput.*, vol. 15, no. 4, pp. 64–69, jul 2011.

[13] D. Catteddu, M. Felici, G. Hogben, A. Holcroft, E. Kosta, R. Leened, C. Millard, M. Niezen, D. Nunez, N. Papanikolaou, S. Pearson, D. Pradelles, C. Reed, C. Rong, J.-C. Royer, D. Stefanatou, and T. W. Wlodarczyk, "Towards a Model of Accountability for Cloud Computing Services," in *Int. Work. Trust. Account. Forensics Cloud*, 2013, pp. 21–30.

[14] W. Benghabrit, H. Grall, J.-C. Royer, M. Sellami, M. Azraoui, K. Elkhiyaoui, M. Onen, A. S. D. Olivera, and K. Bernsmed, "A Cloud Accountability Policy Representation Framework," in *CLOSER-4th Int. Conf. Cloud Comput. Serv. Sci.*, 2014, pp. 489–498.

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

[16] M. Low, H. Davey, and K. Hooper, "Accounting scandals, ethical dilemmas and educational challenges," *Crit. Perspect. Account.*, vol. 19, no. 2, pp. 222–254, Feb 2008.

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

[18] 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.

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

[20] K. Gilman and J. Schulschenk, "Sustainability Accounting Standards Board," pp. 14–17, 2012. [Online]. Available: www.sasb.org Last accessed: 30 November 2017.

[21] R. G. Eccles, K. Perkins, and G. Serafeim, "How to become a sustainable company," *MIT Sloan Manag. Rev.*, vol. 53, pp. 43–50, 2012.

[22] R. G. Eccles, I. Ioannou, and G. Serafeim, "The impact of corporate sustainability on organizational processes and performance," *Manage. Sci.*, vol. 60, no. 11, pp. 2835–2857, 2014.

[23] F. S. Chapin, G. P. Kofinas, and C. Folke, *Principles of ecosystem stewardship: Resilience-based natural resource management in a changing world*. New York: Springer, 2009.

[24] G. Hamel and L. Välikangas, "The quest for resilience," *Harv. Bus. Rev.*, vol. 81, no. 9, pp. 52–63, 131, sep 2003.

[25] G. Sundström and E. Hollnagel, "Learning How to Create Resilience in Business Systems," *Resil. Eng. Concepts Precepts.*, pp. 1–20, 2006.

[26] J. Birchall and L. H. Ketilson, *Resilience of the Cooperative Business Model in Times of Crisis Sustainable Enterprise Programme*. International Labour Organization, 2009.

[27] G. C. Avery and H. Bergsteiner, "Sustainable leadership practices for enhancing business resilience and performance," *Strateg. Leadersh.*, vol. 39, no. 3, pp. 5–15, 2011.

[28] T. Prior and J. Hagmann, "Measuring resilience: methodological and political challenges of a trend security concept," *J. Risk Res.*, no. January 2015, pp. 37–41, 2013.

[29] 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.

[30] L. G. Price, "The Concept of Fiduciary Duty As a Basis for Corporate Ethics," *J. Business, Soc. Gov.*, vol. 3, pp. 21–30, 2011.

[31] B. Withers and M. Ebrahimpour, "The Effects of Codes of Ethics on the Supply Chain: A Comparison of LEs and SMEs," *J. Bus. Econ. Stud.*, vol. 19, no. 1, pp. 24–40,118–119, 2013.

[32] G. R. Weaver, "Encouraging Ethics in Organizations: A Review of Some Key Research Findings," *Am. Crim. Law Rev.*, vol. 51, no. 107, pp. 293–316, 2014.

[33] T. Hahn, F. Figge, J. Pinkse, and L. Preuss, "Editorial Trade-Offs in Corporate Sustainability: You Can't Have Your Cake and Eat It," *Bus. Strateg. Environ.*, vol. 19, pp. 217–229, 2010.

[34] A. Lindgreen and V. Swaen, "Corporate social responsibility," *Int. J. Manag. Rev.*, vol. 12, pp. 1–7, 2010.

[35] D. J. Wood, "Measuring corporate social performance: A review," *Int. J. Manag. Rev.*, vol. 12, pp. 50–84, 2010.

[36] T. Green and J. Peloza, "How does corporate social responsibility create value for consumers?" *J. Consum. Mark.*, vol. 28, pp. 48–56, 2011.

[37] A. Christofi, P. Christofi, and S. Sisaye, "Corporate sustainability: historical development and reporting practices," *Manag. Res. Rev.*, vol. 35, no. 2, pp. 157–172, 2012.

[38] N. Rahman and C. Post, "Measurement Issues in Environmental Corporate Social Responsibility (ECSR): Toward a Transparent, Reliable, and Construct Valid Instrument," *J. Bus. Ethics*, vol. 105, pp. 307–319, 2012.

[39] M. A. Delmas, D. Etzion, and N. Nairn-Birch, "Triangulating Environmental Performance: What Do Corporate Social Responsibility Ratings Really Capture?" *Acad. Manag. Perspect.*, vol. 27, no. 3, pp. 255–267, 2013.

[40] I. Montiel and J. Delgado-Ceballos, "Defining and Measuring Corporate Sustainability: Are We There Yet?" *Organ. Environ.*, pp. 1–27, 2014.

[41] 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.

[42] B. Duncan and M. Whittington, "Company Management Approaches Stewardship or Agency: Which Promotes Better Security in Cloud Ecosystems?" in *Cloud Comput. 2015*. Nice: IEEE, 2015, pp. 154–159.

[43] T. Keyun, R., Carthy, J., & Kechadi, "Cloud Forensics An Overview," in *7th IFIP Conf. Digit. Forensics*, no. January, 2011, pp. 35–46.

[44] NIST, "NIST Cloud Computing Forensic Science Challenges," p. 51, 2014.

[45] S. Almulla, Y. Iraqi, and A. Jones, "A State-Of-The-Art Review Of Cloud," *J. Digit. Forensics, Secur. Law*, vol. V9N4, pp. 7–28, 2014.

[46] R. K. L. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, B. S. Lee, and Q. Liang, "TrustCloud: A Framework for Accountability and Trust in Cloud Computing," *Perspective*, pp. 1–9, 2011.

[47] 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.

[48] 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: 30 November 2017.

[49] OED, "Oxford English Dictionary," 2016. [Online]. Available: www.oed.com Last accessed: 30 November 2017.

[50] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the Intercloud Protocols and Formats for Cloud Computing Interoperability," in *Internet Web Appl. Serv. 2009. ICIW'09. Fourth Int. Conf.*, 2009, pp. 328—-336.

[51] 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: 30 November 2017.

[52] S. Pearson and A. Benameur, "Privacy, Security and Trust Issues Arising from Cloud Computing," *2010 IEEE Second Int. Conf. Cloud Comput. Technol. Sci.*, pp. 693–702, nov 2010.

[53] R. K. L. Ko, "Data Accountability in Cloud Systems," in *Secur. Priv. Trust Cloud Syst.* Springer, 2014, pp. 211–238.

[54] 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: 30 November 2017.

[55] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, C. A. Long, Ed. Wiley, 2008, vol. 50, no. 5.

[56] Verizon, "2012 Data Breach Investigation Report: A study conducted by the Verizon RISK Team in cooperation with the United States Secret Service and Others," Tech. Rep., 2012. [Online]. Available: http://www.verizonenterprise.com/resources/reports/rp{_}data-breach-investigations-report-2012{_}en{_}xg.pdf Last accessed: 30 November 2017.

[57] OWASP, "OWASP home page," 2017. [Online]. Available: https://www.owasp.org/index.php/Main{_}Page Last accessed: 30 November 2017.

[58] C. J. M. Tauro, S. Aravindh, and A. B. Shreeharsha, "Comparative study of the new generation, agile, scalable, high performance NOSQL databases," Int. J. Comput. Appl., vol. 48, no. 20, pp. 14, 2012.

[59] D. G. Chandra, "BASE analysis of NoSQL database," Futur. Gener. Comput. Syst., vol. 52, pp. 1321, 2015.

[60] Y. N. Silva, I. Almeida, and M. Queiroz, "SQL: From traditional databases to big data," in Proceedings of the 47th ACM Technical Symposium on Computing Science Education, 2016, pp. 413418.

[61] D. Wilson and G. Ateniese, "From pretty good to great: Enhancing pgp using bitcoin and the blockchain," in International Conference on Network and System Security, 2015, pp. 368375.

[62] V. L. Lemieux, "Trusting records: is Blockchain technology the answer?," Rec. Manag. J., vol. 26, no. 2, 2016.

[63] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Blockchain-based database to ensure data integrity in cloud computing environments," in CEUR Workshop Proceedings, 2017, vol. 1816.

[64] Verizon, "Verizon 2015 Data Breach Investigation Report," Tech. Rep., 2015.

[65] OWASP, "OWASP IoT Security Guidance," 2016. [Online]. Available: https://www.owasp.org/index.php/IoT{_}Security{_}Guidance Last accessed: 30 November 2017.

[66] OWASP, "OWASP Injection Prevention Cheat Sheet," 2016. [Online]. Available: https://www.owasp.org/index.php/Injection{_}Prevention{_}Cheat{_}Sheet Last accessed: 30 November 2017.

[67] OWASP, "OWASP LDAP Injection Prevention Cheat Sheet," 2016. [Online]. Available: https://www.owasp.org/index.php/LDAP{_}Injection{_}Prevention{_}Cheat{_}Sheet Last accessed: 30 November 2017.

[68] AIS, "Bright Internet Initiative," 2017. [Online]. Available: http://aisnet.org/?page=BrightICT Last accessed: 30 November 2017.