Using Unikernels to Address the Cloud Forensic Problem and help Achieve EU GDPR Compliance

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

Andreas Happe
Dept. Digital Safety & Security
Austrian Inst. of Tech. GmbH
Vienna, Austria
Email: andreas.happe@ait.ac.at

Alfred Bratterud
Dept. of Computer Science
Oslo and Akershus University
Oslo, Norway
Email: alfred.bratterud@hioa.no

***Abstract*—IT security and privacy is a challenging problem to address, and when cloud is used, there is an exponential increase in the challenge. A particular challenge is the cloud forensic problem, which arises when an attacker succeeds in breaching a cloud system, because one of the first aims is to delete the forensic trail, and there is little to prevent this from happening in cloud. Quite apart from the obvious difficulties this will present to finding out who has breached the system and how they got in, there will now be a far more pressing problem to be dealt with, namely, the forthcoming European Union General Data Protection Regulation. Once a breach has been identified, it is also necessary for the company to report the impact of the breach, to include which records were accessed, modified, deleted, or ex-filtrated, on pain of punitive levels of fine. Where the forensic trail has been compromised, this might prove to be a huge challenge to comply with. We propose addressing this problem through the use of Unikernel based monitoring systems which can ensure both full forensic and audit trails can be maintained.**

Keywords–Cloud Forensic Problem; unikernels; EU GDPR, compliance.

I. INTRODUCTION

Every business is the subject of cyber attacks, no matter whether it is a public corporation, a private firm, a financial institution, a government agency, a non-government agency or a charity. No matter what type of organisation is involved, all will be subject to the rules of the forthcoming European Union (EU) General Data Protection Regulation (GDPR) [1], which comes into effect on 25th May 2018. It will apply to every single organisation that deals with any individual who is resident anywhere within the EU; and in a post-Brexit world, the UK Government has indicated that the GDPR will still apply in the UK. Indeed, the UK Government have indicated that the UK GDPR will be enforced with greater rigour, and will accord greater rights to private individuals.

The rules of the GDPR will mean considerable extra work, and expense, for all these organisations who fall under the scope of the GDPR, which will basically include any organisation, anywhere on earth, who process the personal data of any EU resident, anywhere within the EU. Each organisation will require to appoint a data controller, who either must have the requisite technical skills, or must be assisted by a person with such technical skills. This will no doubt be an unwelcome additional expense. They must also have a data processor and a data protection officer, meaning more costs. In addition, they will have to take all necessary technical steps to ensure the security and privacy of all Personally Identifiable Information (PII) belonging to data subjects of the organisation, at yet more expense.

A great many companies are totally unprepared for this. Many believe that because the reporting requirement has been changed from "within 72 hours of a breach occurring" to "within 72 hours of discovering a breach", they will have no problem being compliant [2]. They will be wrong! They must also be able to report which records were accessed, modified, deleted or ex-filtrated from the system.

Once an attacker breaches a system and becomes resident as an intruder, one of their first tasks is to delete the forensic trail of their incursion into the enterprise systems, so that their presence becomes more covert, thus allowing them to remain hidden inside the system. This allows them to harvest whatever information or secrets they desire for as long as they can stay hidden in the system.

Once the forensic trail has been deleted as part of the attacker seeking to retain an ever more capable foothold inside the system, there may be a reduced ability to actually comply with this particular GDPR requirement. In some cases, compliance may even be completely impossible. This will particularly be the case with cloud systems, since there is nothing to prevent such an intruder from deleting not only the forensic trail, but anything else they desire, including the very running cloud instance that they are hiding within.

Since failure to comply can result in fines which can rise to the greater of €20 million or 4% of global turnover, then this will certainly have a substantial impact, although there are many who still fail to grasp this important point.

We start by considering the cloud forensic problem in Section II, and discuss why this is such a challenge for GDPR compliance in cloud systems. We are concerned with achieving both good security and good privacy. While it is possible to have security without privacy, it is not possible to have privacy without security. Thus our approach will be to first ensure a good level of security can be achieved, and to that end, we start by listing the specific security issues we seek to address and discuss how we propose to tackle them in Section III. The remainder of the paper is organized as follows: in Section IV, we consider how we might go about finding a cloud based solution, in Section V, we discuss the outline technical details of our proposed approach; In Section VI, we consider possible attack vectors. In Section VII, we consider just how robust a unikernel approach might be. In Section VIII, we discuss our conclusions.

 ISBN: 978-1-61208-607-1

II. The Cloud Forensic Problem and the GDPR

Cloud computing has been around now for over 10 years, and a great deal of good quality research has been carried out, particularly regarding matters of security and privacy. Cloud systems have become highly popular with companies due to the flexibility of cloud offerings. The speed of starting a cloud instance, the removal of long lead times to access compute and storage resources, the ability to scale up, as well as down, to match demand presents a particularly good incentive to use cloud computing. The fact that companies can write costs off entirely against revenue provides a further attractive incentive for their use. Kratzke [3] has long warned of the dangers of thinking that conventional software is just the same as cloud-native software. Kratzke et al [4] do suggest the possibility of using existing Container technologies to improve cloud-native programming.

There have been many good papers produced on both security [5]–[9] and privacy [10]–[14], and we laud the efforts of countless researchers who have tried to provide this area with better security and privacy, which speaking generally, has been successfully achieved over the years. But there remains one fundamental weakness that has not been resolved, namely the "cloud forensic problem". All computer systems are the subject of attack, and cloud systems are no exception. Unfortunately, no system is immune to attack, and that is certainly true for cloud systems.

The primary goal of an attacker is to breach a system. This can involve quite a considerable amount of work on the part of a serious attacker. They will perform surveillance and compile many analyses of how the company systems are organised. Many will carry out huge amounts of work to understand the people of the organisation, since they are usually the weakest link. This intelligence gathering will be very extensive, usually covering every possible aspect of all the systems of the company in order to discover everything they can about the business architecture before they start their attacks.

Other attackers, are much less organised. They will simply try to hack in to company systems, without a thought of the overview of the company concerned. They will merely look for known vulnerabilities and try to attack them. Yet other attackers will attack the people of the company through social engineering, email attacks through malicious links and malware payloads, web based drive by attacks, phishing, vishing and many other approaches.

No matter which type of attacker they are, they all share one fundamental goal — to penetrate the system in order to become an intruder. The aim here is not just to get in, and out, as quickly as possible, but to develop a long term foothold inside company systems which will allow them to return, time and again, to help themselves to whatever they wish, as they escalate privileges more and more, the longer they remain inside the system.

It is rather unfortunate that they are often greatly aided in this quest by the companies themselves. Usually, this occurs through a degree of laziness whereby the companies are clearly failing to monitor server logs properly. Looking at previous cyber breach reports [15], at which time there was a global average of 6 months between breach and discovery, it is clear that had these companies been more rigorous about reading and analysing their server logs, they would have been in a better position to discover intruders rather more quickly. Even last year, where the time between breach and discovery has dropped to a number of weeks rather than months [16], this is still not good enough. Some companies contribute further by refusing or failing to update security patches to both operating systems and software systems, usually under the guise of "last time I did a security update, all the systems crashed".

This all leads towards the, as yet unresolved, cloud forensic problem — namely, that once an intruder is in the system, and has escalated sufficient privileges, there is absolutely nothing to prevent them from deleting the forensic trail, which allows them to hide all evidence of their successful penetration. Worse, by this stage they will also have control of all the system logs and audit trails, and there is nothing to prevent them from deleting every last trace of their intrusion and ongoing ex-filtration of private data.

Surely that has nothing to do with the GDPR you might ask? Sadly, that is not the case. In the event of a breach, you are required to report the breach within 72 hours of discovering the breach. You must be able to report how many relevant records have been compromised, whether by having been read, amended, deleted or ex-filtrated from the system. Given that many system logs are not even turned on by default, this means identifying which records have been compromised, whether by having been read, amended, deleted or ex-filtrated from the system, will present a serious enough challenge in the first place. However, given that the intruder will likely have done a complete job on all forensic trails in the system, the likelihood of being able to realise that a breach has occurred in the first place will be very slim, let alone having the ability to properly identify which records have been compromised.

From a holistic perspective, it would have been helpful if these matters might have been addressed by the Cloud implementation itself. However, no such attempt has taken place during the past decade, no doubt due to the massive challenge involved. Consequently, all organizations subject to the provisions of the GDPR are required to safeguard their own systems and therefore take such steps as are necessary to ensure adequate privacy is achieved.

This will mean non-compliance with the GDPR, which can then trigger fines which can rise to the greater of €20,000,000 or 4% of global turnover. This will certainly catch the attention of top management within organisations. Considering that these fines can be levied for every single breach, and that the upper limit is based on turnover rather than profit, that should be sufficiently concerning to get some serious attention. Of course, all sensible Cloud users should have been thinking about this long before now, and we are aware of many who on hearing that notification 'within 72 hours of discovery of a breach', rathr than 'within 72 hours of occurrence of a breach', heaved a collective sigh of relief and stopped worrying about implementing a solution. This is what motivates our work.

III. How do we tackle the problem?

At this time, no system is fully secure. Operating systems, transport protocols, software applications — all of this software has evolved during previous decades. Any relevant standards were defined decades ago. The primary goal at that time was functionality. Security and privacy were very much an afterthought, which has remained the case for decades.

 ISBN: 978-1-61208-607-1

Security and privacy has very much been a case of "Let us bolt something on to tackle that". Default settings are geared for ease of setting up, not for security and privacy. This means proper security and privacy presents a massive challenge, which increases exponentially for cloud, Internet of Things and Big Data.

We could opt to use Containers, such as Docker, LXD or Rocket. However, Bratterud et al [17] warn of some security issues with this approach, and Ktatzke [18] also warns of the unexpected, and unwelcome overhead these solutions can bring.

In previous work, [19], we considered how well unikernels might be used to improve on dealing with our target list of security goals, and found the potential for an improved approach. In [20], we developed a suitable framework, providing detailed definitions of how this might be tackled. In [21], we demonstrated how a unikernel based solution could reduce complexity, while improving security and privacy. We also considered in [22], whether unikernels could help address some of the key weaknesses introduced by use of the Internet of Things (IoT). In each case, we build on the work of the previous papers, in order to ensure we do not miss anything important as we develop the system.

Unikernels run natively on cloud, they have an exceptionally small footprint, they run without many of the conventional "toys" associated with normal web based cloud instances. This means a seriously minimal attack surface. They are lightweight, can be activated "on demand", and are extremely difficult to attack. Virtually every single conventional attack fails due to there being a heavily restricted means of accessing the running unikernels. A typical cloud instance will be over 150MB in size. Even Docker containers will be a minimum of 24MB in size, whereas a unikernel instance can be as little as 2MB in size.

Given the limitation we face in terms of most software being insecure, how can we approach developing a potential solution for this problem? In [23] [24] Duncan and Whittington proposed that all cloud based systems which would be subject to compliance under the GDPR, should have ALL audit trails and forensic logs captured and stored off-site in a highly secure immutable database running on a dedicated and highly secure server.

These proposals suggest the immutable database be set up off-site from the cloud instance. While we accept that advice might be highly appropriate given the pervasive extent of the cloud forensic problem, could there be any other way that we might be able to find a cloud based solution? As we shall see in the next section, there may be a way to achieve just that objective.

IV. Finding a Cloud Based Solution

We certainly do accept the sensible logic proposed by Duncan and Whittington [23] [24] to keep the immutable database separated from all running cloud instances. While that makes perfect sense, there is no reason, other than the cloud forensic problem, why the immutable database should not run on a cloud system. However, we do agree that it should not run on the same system as the company system it is trying to protect.

So the question we must first address is how we might go about solving this problem. This is where the unikernel based system might be able to help.

Let us first consider the advantages from a security point of view of unikernels:

- The larger a piece of software, the more vulnerabilities are usually present. As we already stated, a unikernel instance can be as little as 2MB;
- The smaller an instance is, the faster a new instance loads;
- The smaller instances are, the cheaper they are to run;
- There is no terminal to log into. The terminal presents one of the easiest attack routes into any system and is usually not well protected from attack. If the attacker cannot log in, achieving a successful attack will be rather difficult to perpetrate;
- The running instance of any unikernel runs with immutable code, meaning no user may inject code into the running unikernel instance.

And now, let us look at any potential disadvantages of unikernels:

- No terminal to log into — a disadvantage for most sys admins. We view this as a huge advantage. If the sys admin cannot login, the attacker has no chance of doing so;
- The running instance is immutable, so it cannot handle changes. We view this as a positive. We are particularly keen to be able to log all changes, system, forensic and audit trail data in a persistent and immutable storage medium off-site. If we cannot change anything, neither can the intruder.

In our view, every item in the above list of advantages and disadvantages are all positive attributes. From a performance, cost, reduced latency and minimised attack surface perspective, all the attributes are highly beneficial for our purposes. In the next section, we will look at how we might deploy these instances to help solve our security challenge.

V. Outline Technical Solution Proposed

We have seen that our unikernel instances can be extremely lightweight, are immutable in operation, have a very small attack surface, perform well, are cheap to run with reduced latency. Because of these advantages, we can use a number of these instances to build a much more robust system.

If we use the analogy of a bee hive, we can apply this approach as part of our solution. In a bee hive, there are a number of specialised bees — there is a single queen bee, hundreds of male drones (whose responsibility is to mate with a queen, after which they die), anything up to 80,000 female worker bees, who look after developing eggs, larvae and pupae, as well as the whole hive, gathering food from flowers outside the hive and defence duties, which they perform to the death, if needed. Each bee performs a specialised function depending on its age. And in the event a queen leaves, gets lost, or dies accidentally, the colony is capable of generating new queens, either full queens, or temporary queens. The ultimate in sustainability.

 ISBN: 978-1-61208-607-1

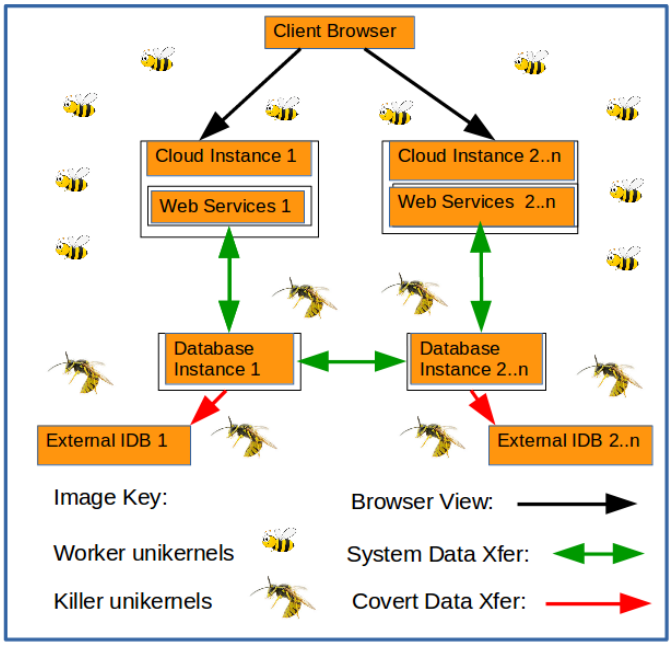

Figure 1. A Unikernel Based Solution to the Cloud Forensic Problem.

Our main company system will have a presence on a cloud platform, using one or more cloud instances as needed, which will be running on a conventional cloud setup. The cloud instance will have the capability to replicate at scale as demand increases and also to shut down instances when demand falls. The main cloud instance system will not be able to be shut down from within. We shall call this the front end Cloud Instance 1.

A conventional database management system will be included in all cloud instances in the normal way except they will instead be removed from within these instances and will run inside a single instance with every non-required function removed from that running instance in order to reduce the attack surface. Should database replication be later required, this can be accommodated through setting up similar database instances. We shall call this original Database Instance 1.

Thus Database Instance 1 will only accept input from the known running front end Cloud Instance 1. There will be no direct access allowed from outside the cloud environment. In the event that replication is required, Cloud Instance 1 will setup as many replicated instances as needed, including Database Instance 2..n, which will all be replicated, expanding to deliver the required resources.

Worker unikernels will be assigned to each Cloud Instance as they are spooled up, and shut down as no longer needed. They will have specific tasks to perform, such as policing, audit, or whatever. Killer unikernels will be assigned to the task of protecting database systems. Their primary goal will be to ensure the safety of both the forensic trail and the audit trail for all database components, which will be safely stored in the immutable database. These records cannot be deleted. If required, these killer unikernels can turn on attackers trying to breach the systems. All unikernel instances will be tracked, with forensic data collected also for them.

As we can see, each different type of instance is specialised, sticking to its own designated tasks. So what is special about this, apart from splitting up the functions? When a cloud instance runs with a variety of different types of software running on it, this can present a big challenge to configure the overall package in a secure way. By specialising each instance, it becomes much easier to configure securely, because every single unused port can be shut down. Security controls can focus on only what they have to, thus cutting down the potential attack surface.

Any new front end instance, if not registered with the control instance, will not be allowed access to any database instance. Likewise where any new database instance is not registered with the control instance, the front end instances will refuse to connect with it.

The secure immutable database for storing system logs, forensic and audit trail data. These should not be directly visible to the client browser. Each running instance will send a copy of all system logs, forensic and audit trail data to the immutable database instance as it is generated. The source and timing of all events will be logged by the immutable database.

With the unikernel instances, because they are so lightweight, we can deploy as many of them as we need to carry out very specific tasks. We can have some to police various events, some to carry out audit tasks, some to keep a track of what is live within the system. Each of the components of the main system can be looked after by a number of dedicated unikernel instances, whose sole function will be dedicated to looking after the one conventional cloud instance. Since these unikernels are self sufficient, there is unlikely to be any real adverse impact on the running main instances.

Figure 1 shows a cross-section of the proposed solution. The Client browser will see the front end which provides conventional running cloud instances, with controllers hidden behind the scenes. These controllers can be protected by ‘killer bee’ unikernels. The external Immutable Database instances will be hosted elsewhere, and can also be protected by ‘killer bee’ unikernels. The ‘worker bee’ unikernels clustering around the conventional cloud instances will carry out normal policing and other required tasks. Additional ‘bee workers’ of whatever kind needed can be spooled up as required. They are fast to provision, take little space and will carry out their programmed task.

As to the question of how many of each type of unikernel we should aim to use, we believe that it would be pointless to speculate at this stage until we can test what will be optimal after we carry out some live experimentation to establish what works well in various loading scenarios. With the use of proper control systems, we can ensure that each new instance is properly registered, constantly and properly monitored, with the control system having the capability to spool up new instances as needed qhickly and efficiently, as well as shutting down those which are no longer required. We expect that such flexibility will allow a highly scalable system to be developed, which can offer minimal running cost, in conjunction with a low latency approach to dealing with attacks. This testing will form part of our future work.

VI. Possible Attack Vectors to Consider

Since we are mostly working with web services, we will look at the Open Web Application Security Project (OWASP) 2017 Top 10 Web Vulnerabilities [25].

 ISBN: 978-1-61208-607-1

A1:2017-Injection Vulnerability: Injection flaws, such as Structured Query Language (SQL), Not Only SQL (NoSQL), Operating System (OS) injection and Lightweight Directory Access Protocol (LDAP) injection, 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. **Solution:** Use a strong Application Programming Interface (API), separate content from commands in the database, and sanitise **ALL** user input.

A2:2017-Broken Authentication Vulnerability: Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently. **Solution** Implement multi-factor authentication; no default passwords, especially from admins; reject all top 10,000 worst passwords.

A3:2017-Sensitive Data Exposure Vulnerability: Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser. **Solution:** Encrypt all PII.

A4:2017-XML External Entities (XXE) Vulnerability: Many older or poorly configured eXtensible Markup Language (XML) processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file Uniform Resource Identifier (URI) handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks. **Solution:** Whenever possible, use less complex data formats such as JavaScript Object Notation (JSON), and avoiding serialization of sensitive data.

A5:2017-Broken Access Control Vulnerability: Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc. **Solution:** With the exception of public resources, deny by default; no unrestricted access to users; log all failures.

A6:2017-Security Misconfiguration Vulnerability: Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured Hypertext Transfer Protocol (HTTP) headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion. **Solution:** Secure installation processes should be implemented. Keep it simple and log all errors.

A7:2017-Cross-Site Scripting (XSS) Vulnerability: XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create Hyper Text Markup Language (HTML) or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites. **Solution:** Preventing XSS requires separation of untrusted data from active browser content.

A8:2017-Insecure Deserialization Vulnerability: Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks. **Solution:** The only safe architectural pattern is not to accept serialized objects from untrusted sources or to use serialization mediums that only permit primitive data types.

A9:2017-Using Components with Known Vulnerabilities Vulnerability: Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts. **Solution:** There should be a patch management process in place to ensure known vulnerabilities are never used.

A10:2017-Insufficient Logging&Monitoring Vulnerability: Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring. **Solution:** This paper is all about solving this problem!

And for no 11 of 10, go check out your site and make sure your system is not vulnerable.

There are, of course, many more vulnerabilities you can check out, and you should. The more you eliminate, the stronger and more robust your system becomes. You can be sure the attacker already knows all the potential vulnerabilities, so you need to make sure you do too, and plug them.

VII. DISCUSSION

We strongly believe that a unikernel based system would have a positive and robust impact because of the extra muscle offered to check and log everything that is happening within the system. Given that unikernel instances have a very low attack surface, no conventional attacker 'toys', are immutable in operation, and highly compact, as well as everything being logged to the immutable database - we are cutting out a huge range of vulnerabilities from existing cloud systems. By ensuring the cloud instance running can also withstand the OWASP top ten web vulnerability test, we are in a very strong position to resist a great many attacks.

Some experimentation will be required to identify what the optimal setup of the 'unkernal hive' instances will be in order to obtain the most effective approach. We need to ensure the cotroller instances are efficiently organised to allow scalability of the overall cloud installation, while at the same time ensuring maximum security and privacy can be achieved. At this time, the Cloud Forensic Problem means that conventional cloud systems cannot guarantee GDPR compliance for cloud users. Container based solutions are likely to be sugject to the same issues as conventional cloud instaces. While they

 ISBN: 978-1-61208-607-1

may very well offer some improvement, it is likely that improvement will come at an overhead cost.

Using the unikernel approach, it is likely that it will certainly be possible to be compliant with the GDPR, that the overhead of running the unikernel instances will be minimal, and that the system can be highly responsive to the need for scalability. Not only that, but the ability to provide a means for compliance for cloud systems has to be big improvement on the status quo.

While we have carried out a number of minor tests on various aspects of our proposal, we have still to carry out any serious testing, which will form the main thrust of our next stage of the work. We are very keen to develop something that can provide a proper solution.

VIII. Conclusion and Future Work

As we have already stated, the Cloud Forensic Problem presents a very serious challenge for all cloud users, especially in light of the forthcoming GDPR. We have proposed a possible solution for this problem, which is a little different from conventional approaches. However, it offers a highly robust solution to a major challenge for all organisations who will be subject to compliance with the GDPR.

We believe this solution offers such merit that we plan to run a pilot test to establish just how well it will be able to cope with a system under serious attack. Initially, it will run on a private network, under attack from professional penetration testers. Once we are sure of how well the solution is likely to perform, we will set up a real live cloud instance to see just how well it might perform.

When the GDPR comes on stream, there will not be time for organisations to mess about. If they cannot comply with the regulation, and they are breached, resulting in a loss of PII, then they can expect huge fines, the like of which they have never seen before. It is time to wake up and smell the coffee.

References

[1] EU, "EU General Data Protection Regulation (GDPR)," 2017. [Online]. Available: http://www.eugdpr.org/ [Retrieved: December 2017]

[2] EU, "Reform of EU data protection rules," 2016. [Online]. Available: http://ec.europa.eu/justice/data-protection/reform/index_en.htm [Retrieved: December 2017]

[3] N. Kratzke and P.-C. Quint, "Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study," Journal of Systems and Software, vol. 126, 2017, pp. 1–16.

[4] N. Kratzke, P.-C. Quint, D. Palme, and D. Reimers, "Project cloud transit-or to simplify cloud-native application provisioning for smes by integrating already available container technologies," European Project Space on Smart Systems, Big Data, Future Internet-Towards Serving the Grand Societal Challenges. SCITEPRESS. In print, 2017.

[5] R. K. L. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, "TrustCloud: A framework for accountability and trust in cloud computing," Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011, 2011, pp. 584–588.

[6] R. K. L. Ko, B. S. Lee, and S. Pearson, "Towards achieving accountability, auditability and trust in cloud computing," Communications in Computer and Information Science, vol. 193 CCIS, no. PART 4, 2011, pp. 432–444.

[7] N. Papanikolaou, S. Pearson, and M. C. Mont, "Towards Natural-Language Understanding and Automated Enforcement of Privacy Rules and Regulations in the Cloud: Survey and Bibliography," Analysis, 2011, pp. 1–9.

[8] S. Pearson, "Taking account of privacy when designing cloud computing services," Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, CLOUD 2009, 2009, pp. 44–52.

[9] S. Pearson, "Towards Accountability in the Cloud," IEEE Internet Computing, vol. 15, no. 4, jul 2011, pp. 64–69.

[10] W. Jansen and T. Grance, "Guidelines on Security and Privacy in Public Cloud Computing," NIST, Tech. Rep. 7, 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf [Retrieved: December 2017]

[11] NIST, "Security and Privacy Controls for Federal Information Systems and Organizations Security and Privacy Controls for Federal Information Systems and Organizations," Natioinal Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. February, 2014. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf [Retrieved: December 2017]

[12] S. Pearson, "Privacy and Security for Cloud Computing," in Privacy and Security for Cloud Computing. e: Springer, 2013, pp. 3–42.

[13] S. S. Shapiro, "Privacy by Design," Communications of the ACM, vol. 53, no. 6, jun 2010, p. 27.

[14] J. Singh, T. F. J. M. Pasquier, and J. Bacon, "Securing tags to control information flows within the Internet of Things," 2015 International Conference on Recent Advances in Internet of Things, RIoT 2015, 2015.

[15] Verizon, "2012 Data Breach Investigations Report," Verizon, Tech. Rep., 2012.

[16] Verizon, "2016 Verizon Data Breach Report," Tech. Rep., 2016.

[17] A. Bratterud, A.-A. Walla, H. Haugerud, P. E. Engelstad, and K. Begnum, "IncludeOS: A Minimal, Resource Efficient Unikernel for Cloud Services," 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), 2015, pp. 250–257.

[18] N. Kratzke, "About microservices, containers and their underestimated impact on network performance," arXiv preprint arXiv:1710.04049, 2017.

[19] B. Duncan, A. Bratterud, and A. Happe, "Enhancing Cloud Security and Privacy: Time for a New Approach?" in Intech 2016, Dublin, 2016, pp. 1–6.

[20] A. Bratterud, A. Happe, and B. Duncan, "Enhancing Cloud Security and Privacy: The Unikernel Solution," in Cloud Computing 2017: The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization, 2017, pp. 1–8.

[21] A. Happe, B. Duncan, and Alfred Sewitsky Bratterud, "Unikernels for Cloud Architectures: How Single Responsibility can Reduce Complexity, Thus Improving Enterprise Cloud Security," in COMPLEXIS 2017 - Proceedings of the 2nd International Conference on Complexity, Future Information Systems and Risk, Porto, Portugal, 2017, pp. 1–12.

[22] B. Duncan, A. Happe, and A. Bratterud, "Enterprise IoT Security and Scalability: How Unikernels can Improve the Status Quo," in 9th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2016), Shanghai, China, 2016, pp. 1–6.

[23] B. Duncan and M. Whittington, "Creating an Immutable Database for Secure Cloud Audit Trail and System Logging," in Cloud Computing 2017: The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization. Athens, Greece: IARIA, ISBN: 978-1-61208-529-6, 2017, pp. 54–59.

[24] B. Duncan and M. Whittington, "Creating and Configuring an Immutable Database for Secure Cloud Audit Trail and System Logging," International Journal On Advances in Security, vol. 10, no. 3&4, 2017, pp. 155–166.

[25] OWASP, "OWASP home page," 2017. [Online]. Available: https://www.owasp.org/index.php/Main_Page [Retrieved: December 2017]

 ISBN: 978-1-61208-607-1