Chapter 2

Cloud Cyber Security: Finding an Effective Approach with Unikernels

Bob Duncan, Andreas Happe and Alfred Bratterud

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/67801

Abstract

Achieving cloud security is not a trivial problem to address. Developing and enforcing good cloud security controls are fundamental requirements if this is to succeed. The very nature of cloud computing can add additional problem layers for cloud security to an already complex problem area. We discuss why this is such an issue, consider what desirable characteristics should be aimed for and propose a novel means of effectively and efficiently achieving these goals through the use of well-designed unikernel-based systems. We have identified a range of issues, which need to be dealt with properly to ensure a robust level of security and privacy can be achieved. We have addressed these issues in both the context of conventional cloud-based systems, as well as in regard to addressing some of the many weaknesses inherent in the Internet of things. We discuss how our proposed approach may help better address these key security issues which we have identified.

Keywords: cloud security and privacy, unikernels, Internet of things

1. Introduction

There are a great many routes into an information system for the attacker, and while many of these routes are well recognized by users, many others do not see the problem, meaning limited work is being carried out on defense, resulting in a far weaker system. This becomes much more difficult to solve in the cloud, due to the multi-tenancy nature of cloud computing, where users are less aware of the multiplicity of companies and people who can access their systems and data. Cloud brings a far higher level of complexity than is the case with traditional distributed systems, in terms of both the additional complexity of managing

new relationships in cloud, and in the additional technical complexities involved in running systems within the cloud. It runs on other people's systems, and instances can be freely spooled up and down, as needed.

Add to this the conception, or rather the misconception, that users can take the software, which runs on their conventional distributed systems network and run it successfully on the cloud without modification, thus missing the point that their solid company firewall does not extend to the cloud, and that they thus lose control over who can access their systems and data. Often, users also miss the point that their system is running on someone's hardware, over which they have limited or no control. While cloud service providers may promise high levels of security, privacy and vetting of staff, the same rigorous standards often do not apply to their subcontractors.

There are many barriers that must be overcome before cloud security can be achieved [1]. A great deal of research has been conducted toward resolving this problem, mostly through technical means alone, but this presents a fundamental flaw. The business architecture of an enterprise comprises people, process and technology [2], and any solution, which will focus on a technological solution alone, will be doomed to failure. People present a serious weakness to enterprise security [3], and while process may be very well documented within an organization, often it is out of date due to the rapid pace of evolution of technology [4]. Technology can benefit enterprises due to the ever improving nature and sophistication of software, which is a good thing, but at the same time can present a greater level of complexity, making proper and secure implementation within enterprise systems much more difficult. Another major concern is that the threat environment is also developing at a considerable pace [5].

Cloud computing has been around for the best part of a decade, yet we still have to see an effective, comprehensive security standard in existence. Those that do exist tend to be focussed on a particular area, rather than the problem as a whole, and as stated above, they are often out of date [4]. Legislators and regulators are not much further advanced. The usual practice is to state what they are seeking to achieve with the legislation or regulatory rules. Usually, they are very light on the detail of how to achieve these goals. To some extent, this is deliberate—if they specify the principles to apply to achieve their desired objective rather than the exact details, they do not have to keep updating the legislation/regulations as circumstances change. Often, they have no clue as to how to achieve these goals anyway, leaving it up to the users to work it out. This is the approach favored by the UK authorities, and it can be argued that it generally works well. In the US, they favor the rules-based approach, which, of necessity, requires far more work on the part of the government and regulators to keep the rules up to date. It also spawns an active industry of specialists who constantly probe the boundaries to see how far they can be pushed. Global enterprises often have to deal with both types of approach. In addition, the methodology deployed to achieve compliance is often flawed [6]. To this complex environment, we must now, of necessity, add the impact of both Industry 4.0, which encompasses mostly high-value targets, e.g. factories, and the Internet of things (IoT), which is likely to see a massive global explosion, to the mix.

http://dx.doi.org/10.5772/67801

The IoT has been around now for a considerable time, but it did not get much traction until the arrival of cloud computing and big data. In 2007, Gantz et al. [7] suggested that global data collection would double every 18 months, a prediction that looks like being very light when compared to the reality of data creation coming from the expansion of the IoT. Cisco noted that the IoT had really come of age in 2008, when there were now more things connected to the Internet than people [8]. The massive impact arising from this enabling of the IoT by cloud computing brings some exciting new applications and future possibilities in the areas of defense, domestic and home automation, eHealth, industrial control, logistics, retail, security and emergencies, smart airports, smart agriculture, smart animal farming, smart cars, smart cities, smart environment, smart metering, smart parking, smart roads, smart trains, smart transport and smart water, but also brings some serious challenges surrounding issues of security and privacy. Due to the proliferation of emerging and cheaply made technology for use in the IoT, it is well known that the technology is particularly vulnerable to attack. When we combine the IoT and big data, we compound this problem further. This area is poorly regulated, with few proper standards yet in place, which would suggest it might be more vulnerable than existing cloud systems, which have been around for some time now.

We are concerned with achieving both good security and good privacy, and while it is possible to have security without privacy, it is not possible to have privacy without security. Thus, our approach is to first ensure a good level of security can be achieved, and in Section 2, we discuss from our perspective how we have set about developing and extending this idea. In Section 3, we identify the issues that need to be addressed. In Section 4, we discuss why these issues are important, and what the potential implications for security and privacy are likely to be. In Section 5, we consider some current solutions proposed to address some of these issues and consider why they do not really address all the issues. In Section 6, we outline our proposed approach to resolve these issues, and in Section 7, we discuss our conclusions.

2. Development of the idea

The authors have developed a novel approach to addressing these problems through the use of unikernel-based systems, which can offer a lightweight, green and secure approach to solving these challenging issues. Duncan et al. [9] started by outlining a number of issues faced and explained how a unikernel-based approach might be used to provide a better solution. Bratterud et al. [10] provide a foundation for the development of formal methods, and to provide some clarity on the identification and use of good clear definitions in this space.

A unikernel is by default a single threaded, single address space mechanism taking up minimal resources, and [11] look at how the concept of single responsibility might be deployed through the use of unikernels in order to reduce complexity, thus reducing the attack surface and allowing for a better level of security to be achieved. Given the worrying expansion of security exploits in IoT, as exemplified by recent DDoS attacks facilitated by the inherent security weaknesses present in IoT architecture, Duncan et al. [12] looked at how the

unikernel approach might be useful when used for IoT and big data applications. Duncan and Whittington [13] consider how to develop an immutable database system using existing database software, thus providing the basis for a possible solution for one of the major needs of the framework.

Unikernels use the concepts of both single address space and single execution flow. A monolithic application could be converted into a single large unikernel, but this would forfeit any real benefits to be gained from this architecture. To prevent this, we propose a framework that aids the deconstruction of business processes into multiple connected unikernels. This would allow us to develop complex systems, albeit in a much more simple, efficient, secure and private way. We must also develop a framework to handle the automated creation and shutting down of multiple unikernels, possibly carrying out a multiplicity of different functions at the same time. This concept is likely to be far more secure than conventional approaches. During runtime, the framework will be responsible for creation, monitoring and stopping of different unikernel services. While unikernels themselves do provide good functional service isolation, external monitoring is essential to prevent starvation attacks, such as where one unikernel effectively performs a denial-of-service attack by consuming all available host resources.

We have identified a number of other areas, which will need further work. We are currently working on developing a means to achieve a secure audit trail, a fundamental requirement to ensure we can retain as complete a forensic trail as possible, for which we require to understand how to properly configure an immutable database system, capable of withstanding penetration by an attacker. This work follows on from Ref. [13]. However, in order to run such a system, we will need to develop a control system to co-ordinate multiple unikernel instances operating in concert. We will also have to develop a proper access control system to ensure we can achieve confidentiality of the system and to maintain proper privacy. To help with the privacy aspects, we will also require to develop a strong, yet efficient approach to encryption.

In addition, the framework must provide means of input/output for managed unikernels, including facilities for communication and data storage.

Communication is both concerned with inter-unikernel communication as well as with providing interfaces for managed unikernels to the outside world. As we foresee a message-passing infrastructure, this should provide means for validating passed messages including deep packet inspection. This allows for per-unikernel network security policies and further compartmentalization, which should minimize the impact of potential security breaches.

In real-world use cases, we require the framework to be capable of handling mutable data, such as the ability to record temporary states, logging information or ensuring that persistent application and or user data can be maintained. Unikernels themselves by definition are immutable. In order to resolve this conflict, the framework must provide a means to persist and QUERY data in a race-free manner. It may be necessary to provide specialized data storage, depending on the use case. For example, system log and audit trail data require special treatment to prevent loss of a complete forensic record, thus requiring an append-only approach. Since persistent data storage is inherently contrary to our immutable unikernel

http://dx.doi.org/10.5772/67801

approach, we do not enforce data storage to be implemented within unikernels. Being pragmatic, we defer this functionality to the framework, i.e. a means of storage is provided by the framework, rather than by the unikernels themselves.

We also believe it may be possible to develop a unikernel-based system to work with the serverless paradigm. With those frameworks, source code is directly uploaded to the cloud service. Execution is triggered in response to events; resources are automatically scaled. Developers do not have any system access except through the programming language and provided libraries. We see unikernel and serverless frameworks as two solutions to a very similar problem, reducing the administrative overhead and allowing developers to focus their energy on application development. Serverless stacks signify the "corporate-cloud" aspect: developers upload their code to external services and thus invoke vendor lock-in in the long run. Unikernels also allow users to minimize the non-application code, but in contrast to serverless architectures, this approach maintains flexibility with regard to hosting. Users can provide on-site hosting or move toward third-party cloud offerings. We expect serverless architecture providers to utilize unikernels within their own offerings. They are well suited to encapsulate the user provided applications and further increase the security of the host's infrastructure.

We are also developing penetration testing approaches, using fuzzing techniques, adapting tools and sanitizers, hardening tools and whatever else we can do to strengthen the user environment to achieve our aims. The ultimate goal is to make life so difficult for the attacker that they will be forced to give up and move on to easier pickings elsewhere. We have also been applying all the usual attack methods to our test systems to assess whether our approach will work. This should allow us to be sure that each component will be fit for purpose before we move on to the next component. In this way, by developing each component of the system to automatically integrate with the rest, the system should ultimately become far more robust as a result.

We now have a good idea of how the concept needs to be developed, and what future plans are needed to progress the development toward a highly secure and efficient system for cloud users. In the next section, we consider what exactly the issues are that we need to address in more detail.

3. What are the issues?

The fundamental concepts of information security are confidentiality, integrity, and availability (CIA), which is also true for cloud security. The business environment is constantly changing [14], as are corporate governance rules and this would clearly imply changing security measures would be required to keep up to date. More emphasis is now being placed on responsibility and accountability [15], social conscience [16], sustainability [17, 18], resilience [19] and ethics [20]. 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. These, then, must be the main goals for information security.

We now consider a list of ten key management security issues identified in Ref. [1], which provide detailed explanations for each of these items on the list. These items represent management-related issues, which are often not properly thought through by enterprise management.

The 10 key management security issues identified are:

- the definition of security goals,
- compliance with standards
- audit issues,
- management approach,
- technical complexity of cloud,
- lack of responsibility and accountability,
- measurement and monitoring,
- management attitude to security,
- security culture in the company,
- the threat environment.

These are not the only issues to contend with. There are a host of technical issues to address, as well as other, less obvious issues, such as social engineering attacks, insider threats (especially dangerous when perpetrated in collaboration with outside parties), state-sponsored attacks, advanced persistent threats, hacktivists, professional criminals, and amateurs, some of whom can be very talented. There are many known technical weaknesses, particularly in web-based systems, but the use of other technology such as mobile access, "bring your own device" (BYOD) access, and IoT can all have an adverse impact on the security and privacy of enterprise data.

In spite of what is known about these issues, enterprises often fail to take the appropriate action to defend against them, or do not understand how to implement or configure this protection properly, leading to further weakness. Staff laziness can be an issue. Failure to adhere to company security and privacy policies can also be an issue. Use of passwords, which are too simple, is an issue. Simple things, such as the use of yellow stickies can be a dangerous weakness when stuck on computer screens, with the user password in full view for the world to see.

Lack of training for staff on how to properly follow security procedures can lead to weakness. Failure to patch systems can be a serious issue. Poor configuration of complex systems is often a major area of weakness. Poor staff understanding of the dangers in email systems presents a major weakness for enterprises. Failure to implement simple steps to protect against many known security issues presents another problem. Lack of proper monitoring of systems presents a serious weakness, with many security breaches being notified by third-party outsiders, usually long after the breach has occurred.

We will take a look at some of these technical vulnerabilities next, starting with one of the most obvious. Since cloud is enabled through the Internet, and web-based systems play a huge role in providing the fundamental building blocks for enterprise systems architecture, it makes sense to look at the vulnerabilities inherent in web-based systems.

3.1. Web vulnerabilities

Security breaches have a negative monetary and publicity impact on enterprises, thus are seldom publicly reported. This limits the availability of empirical study data on actively exploited vulnerabilities. However, web vulnerabilities are well understood, and we can source useful information on the risks faced through this medium by using data from the work of the Open Web Application Security Project (OWASP) [21], who publish a top 10 list of web security vulnerabilities every 3 years.

The OWASP Top 10 report [21] provides a periodic list of exploited web application vulnerabilities, ordered by their prevalence. OWASP focuses on deliberate attacks, each of which might be based upon an underlying programming error—for example, an injection vulnerability might be the symptom of an underlying buffer overflow programming error. OWASP also provides the most comprehensive list of the most dangerous vulnerabilities and a number of very good mitigation suggestions. The last three OWASP lists for 2007, 2010 and 2013 are provided in **Table 1**.

This list, based on the result of analysis of successful security breaches across the globe, seeks to highlight the worst areas of weakness in web-based systems. It is not meant to be

2013	2010	2007	Threat
A1	A1	A2	Injection attacks
A2	A3	A7	Broken authentication and session management
A3	A2	A1	Cross site scripting (XSS)
A4	A4	A4	Insecure direct object references
A5	A6	-	Security misconfiguration
A6	-	-	Sensitive data exposure
A7	-	-	Missing function level access control
A8	A5	A5	Cross site request forgery (CSRF)
A9	-	-	Using components with known vulnerabilities
A10	-	-	Unvalidated redirects and forwards

Table 1. OWASP top ten web vulnerabilities—2013 to 2007 [21].

exhaustive, but instead merely illustrates the worst 10 vulnerabilities in computing systems globally. It is clearly concerning that the same vulnerabilities continue to recur year after year, which clearly demonstrates the failure of enterprises to adequately protect their resources properly.

Thus in any cloud-based system, these vulnerabilities are likely to be present. However, there are likely to be additional potential vulnerabilities, which will also need to be considered. We group the different vulnerabilities into three classes based on their impact on software development. Low-level vulnerabilities can be solved by applying local defensive measures, such as using a library at a vulnerable spot. High-level vulnerabilities cannot be solved by local changes, but instead need systematic architectural treatment. The last class of vulnerability is application workflow-specific and cannot be solved automatically but instead depends on thoughtful developer intervention.

Two of the top three vulnerabilities, A1 and A3, are directly related to either missing input validation or output sanitation. Those issues can be mitigated by consistently utilizing defensive security libraries. Another class of attack that can similarly be solved through a "low-level" library approach is A8. By contrast, "high-level" vulnerabilities should be solved at an architectural level. Examples of these are A2, A5 and A7. The software architecture should provide generic means for user authentication and authorization, and should enforce these validations for all operations. Some vulnerability classes, i.e. A4, A6 and A10, directly depend on the implemented application logic and are hard to protect against in a generic manner. Some other vulnerabilities can be prevented by consistently using security libraries, while other vulnerabilities can be reduced by enforcing architectural decisions during software development.

New software projects are often based upon existing frameworks. Those frameworks bundle both default configuration settings as well as a preselection of libraries providing either features or defensive mechanisms. Software security is mostly regarded as a non-functional requirement and thus can be hard to get funding for. Those opinionated frameworks allowed software developers to focus on functional requirements while the frameworks took care of some security vulnerabilities.

Over the years, those very security frameworks have grown in size and functionality, and as they themselves are software products, they can introduce additional security problems into otherwise secure application code. For example, while the Ruby on Rails framework, properly used, prevents many occurrences of XSS-, SQLi- and CSRF-attacks, recent problems with network object serialization introduced remotely exploitable injection attacks [22]. The affected serialization capability was not commonly used but was included in every affected Ruby on Rails installation. Similar problems have plagued Python and its Django framework [23]. All of these are further aggravated as, by design, software frameworks are generic—they introduce additional software dependencies, which might not be used by the application code at all. Their configuration often focuses on developer usability, including an easy debug infrastructure. Unfortunately, from a security perspective, everything that aids debugging also aids penetration.

http://dx.doi.org/10.5772/67801

OWASP acknowledged this problem in its 2013 report by introducing A9. The reason for adding a new attack vector class was given as: "the growth and depth of component based development has significantly increased the risk of using known vulnerable components" [21].

Of course, when it comes to the use of IoT with cloud, we need to look beyond basic web vulnerabilities. The IoT can also use mobile technology to facilitate data communication, as well as a host of inherently insecure hardware, and we look at this in more detail in the next section.

3.2. Some additional IoT vulnerabilities

OWASP now produces a list of the worst 10 vulnerabilities in the use of mobile technology, which we show in the list of **Table 2**.

Of course, it is not quite as simple as that the IoT mechanics extend beyond traditional web technology and mobile technology. In 2014, OWASP also developed a provisional top 10 list of IoT vulnerabilities, which we outline in **Table 3**.

An important point to bear in mind is that the above list represents just the OWASP top 10 vulnerability list. OWASP is currently working on a full list of 130 possible IoT vulnerabilities, which should be taken into account. OWASP also provides some very good suggestions on how to mitigate these issues.

While the above just covers security issues, we also have to consider the challenges presented by privacy issues. With the increase in punitive legislation and regulation surrounding issues of privacy, we must necessarily concern ourselves with providing the means to ensure the goal of privacy can be achieved. The good news is that if we can achieve a high level of security, then it will be much easier to achieve a good level of privacy [9]. Good privacy is heavily dependent on having a high level of security. We can have security without privacy, but we cannot have privacy without security.

2013 code	Threat
M1	Insecure data storage
M2	Weak server side controls
M3	Insufficient transport layer protection
M4	Client side injection
M5	Poor authorization and authentication
M6	Improper session handling
M7	Security decisions via untrusted inputs
M8	Side channel data leakage
M9	Broken cryptography
M10	Sensitive information disclosure

Table 2. OWASP top ten mobile vulnerabilities—2013 [21].

2014 Code	Threat
I1	Insecure web interface
I2	Insufficient authentication/authorization
I3	Insecure network services
I4	Lack of transport encryption
I5	Privacy concerns
I6	Insecure cloud interface
I7	Insecure mobile interface
I8	Insufficient security configure-ability
I9	Insecure software/firmware
I10	Poor physical security

Table 3. OWASP top ten IoT vulnerabilities—2014 [24].

While the IoT has progressed significantly in recent years, both in terms of market uptake and in increased technical capability, it has very much done so at the expense of security and privacy. For example, accessing utility companies, including nuclear in the US [25], damage caused to German steel mill by hackers [26], drug dispensing machines hacked in US [27], plane taken over by security expert mid-air [28], and a hack that switched off smart fridges if it detected ice cream [29]. While enterprises often might not care too much about these issues, they should. If nothing else, legislators and regulators are unlikely to forget, and will be keen to pursue enterprises for security and privacy breaches. In previous years, it was often the case that legislators and regulators had little teeth, but consider how punitive fines have become in recent years following the banking crisis in 2008. In the UK in 2014, the Financial Conduct Authority (FCA) fined a total of £1, 427, 943, 800 [30], during the year, a more than 40 fold increase on 5 years previously.

As we already stated in Section 1, there are no standards when it comes to components for the IoT. This means there is a huge range of different architectures vying for a place in this potentially massive market space. Obviously, from a technical standpoint, greater flexibility and power can be obtained through good use of virtualization. Virtualization is not new and has been around since 1973 [31]. Bearing in mind that dumb sensors do not have enough resources or lack hardware support for virtualization (or at least Linux-based virtualization), we will have a quick look at some of the most popular hardware in use in this space.

ARM [32] presented the ARM capabilities in 2009. ARM is one of the most used platforms in the IoT and has virtualization extensions. Columbia University has developed KVM/ARM, an open-source ARM virtualization system [33]. Dall and Nieh [34] have written an article on this work for LWN.net and for a conference [35]. Paravirtualization support in ARM Coretex A8 has been around since 2006, and ARM Coretex A9 since 2008, with full virtualization since approximately 2009. Virtualization is also in Linux Kernel 3.8. There are also MMU-less ARMs, although it is unlikely that these could be used, unless we were to forfeit the unikernel's protection.

Many modern smart devices can handle virtualization—devices such as play stations, smart automotive systems, smart phones, smart TVs and video boxes. This may not necessarily be the case for small embedded components, such as wear-ables, sensors and other IoT components. MIPS also supports virtualization [36, 37]. Some Intel Atom processors support virtualization (the atom range is huge). However, the low-power Intel Quark has absolutely no support for virtualization. The new open-source RISC-V architecture [38] does support virtualization.

Many current IoT systems in use do have the capability to handle virtualization. For example, most high-powered NAS systems now have virtualization and application support. Thus, we could potentially utilize NAS or other low-powered devices, many of which are ARM, MIPS or x86, to aggregate data on-site and then transport the reduced volume of data to the cloud.

Right now, we must carefully consider the current state of security and privacy in a massively connected world. It is clear that "big brother" is very capable of watching. Not just through the use of massive CCTV networks, but also through IoT-enabled devices, which will become embedded in every smart city. It is estimated that in smart cities of the future, there will be at least 5000 sensors watching as you move through the city at all times. How much personal information could leak as you walk? How much of your money could NFC technology in the wrong hands steal from you, without you being aware of it happening? Do you trust the current technology? We can read about more of these issues in Ref. [39].

3.3. Some basic enterprise vulnerabilities

Of course, there are some additional enterprise vulnerabilities that we also need to take into account. These are frequently exploited by the threat environment, and thankfully, we have access to some statistics collected by various security breach reports issued by many security companies [40–42], which will clearly demonstrate the security and privacy problems still faced today, including the fact that the same attacks continue to be successful year on year, as demonstrated by the six-year summary of the Verizon reports shown in **Table 4**. There is no figure provided by Verizon for 2015, as they changed the layout for that year.

We have been looking at an extensive range of management and technical issues above. Yet, there are some fundamental issues which impact directly on the people in the enterprise, as exemplified by the image below in **Figure 1**.

These attacks have been successfully used for decades, in particular the first three, which also happen to be the most devastating. It is no joke to state that in any organization "People are the weakest link", because the first three rely entirely on the inattentiveness and stupidity of users to succeed.

Thus, we can see that there are a considerable number of issues, which all enterprises will face when considering how to run a system, which can offer both a good level of security and privacy. It is necessary to raise awareness of these issues and reasons as to why they are important, and so, we take a look at this in the next section.

Threat	2010	2011	2012	2013	2014	2016
Hacking	2	1	1	1	1	1
Malware	3	2	2	2	2	2
Misuse by company employees	1	4	5	5	5	4
Physical theft or unauthorized access	5	3	4	3	4	6
Social engineering	4	5	3	4	3	3

Table 4. Verizon top 5 security breaches—2010 to 2014, 2016 (1 = highest) [40, 43–47].

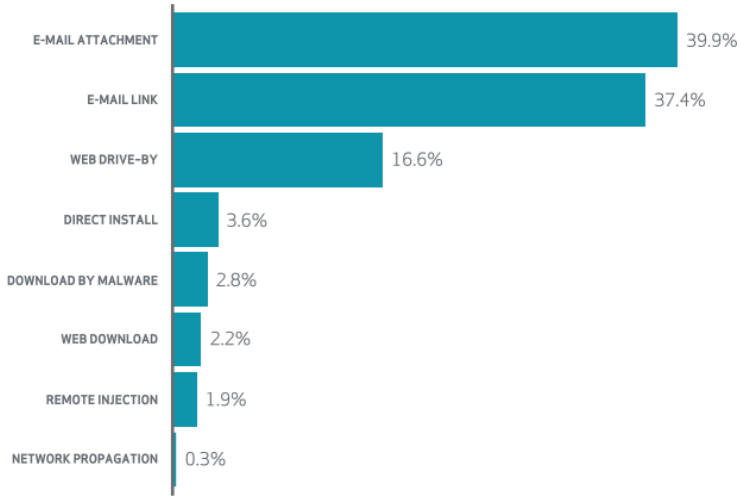

Figure 1. The most successful people attacks ©2015 Verizon.

4. Why this is important

This question is fairly obvious and easy to answer. Security breaches have a negative monetary and publicity impact on enterprises. In light of the increasing fine levels being applied by regulators, particularly in the light of the forthcoming EU General Data Protection Regulation (GDPR), which introduces the potential for a massive increase in fine levels, enterprises are starting to understand just how much of an impact this will have for them. The impact on an enterprise of a breach can be considerable, particularly, as often happens, where the breach is identified by third parties. This generally ensures that the impact on reputation will be much higher.

Staff often cannot believe that a breach has taken place, and by the time they realize what has happened, the smart attackers have eradicated all traces of their incursion into the system.

http://dx.doi.org/10.5772/67801

There will be virtually zero forensic evidence, so even bringing in expensive security consultants with highly competent forensic teams will be of little value. Quite apart from the possible financial loss to the company from any theft of money or other liquid assets, there will be the possibility of huge reputational damage, which can have a serious negative impact on their share price. Such enterprises are unlikely to have a decent business continuity plan either, which makes recovery even more difficult.

If an enterprise cannot tell how the attackers got in, what data they stole, or compromised, or how they got their hands on whatever was removed, it will be very difficult, time consuming and expensive to recover from such an event. Where an enterprise has a robust defensive strategy implemented, at least in the event of a breach, they will have a better chance of containing the impact of the breach, and a better chance of being able to learn from the evidence left behind.

Another good reason for considering this as an important issue is that it may well be a mandatory legislative or regulatory requirement for the enterprise to maintain such a defensive system. It is clear that legislators and regulators have started to take cyber security far more seriously, and therefore, it is likely that the level of fines levied for breaches will continue to rise.

We provide here a few examples to demonstrate how the authorities are starting to get tougher:

- In 2016, ASUS was sued by the Federal Trade Commission (FTC) [48], because they were not providing updates for their insecure routers.
- In January, the FTC started investigations into D-Link [49, 50], "Defendants have failed to take reasonable steps to protect their routers and IP cameras from widely known and reasonably foreseeable risks of unauthorized access, including by failing to protect against flaws which the Open Web Application Security Project has ranked among the most critical and widespread web application vulnerabilities since at least 2007".
- Schneier [51] recently spoke in front of the US House in favor of legal frameworks for IoT Security.
- In the EU, the General Data Protection Regulation (GDPR) will come into force in May 2018. It will also increase this (monetary) problem for companies with a maximum monetary penalty for a single breach of the higher of €10m or 2% of global turnover, and for a more serious breach involving multiple failings, the higher of €20m or 4% of global turnover.

Despite the fact that cyber security is often seen as falling into the "Cinderella" category of IT expenditure, there is abundant evidence that there are clear benefits to be derived from taking this matter seriously. For those who are not entirely convinced by these arguments, there are many security breach companies who compile annual reports showing the most successful breaches, providing some insight into the weaknesses that expose these vulnerabilities, and discussing the potential mitigation strategies that might be deployed in defense of these attacks. In the next section, we take a look at how current solutions approach these issues.

5. Current solutions

Over recent years, a great many solutions to these problems have been proposed, developed and implemented. The early works were generally directed toward conventional corporate distributed IT systems. By comparison to the cloud, these are generally well understood and relatively easy to resolve, and many companies have enjoyed good levels of protection as a result of these efforts. However, cloud changes the rules of the game considerably; because there is often a poor understanding of the technical complexities of cloud, and often a complete lack of understanding that the cloud runs on someone else's hardware and often software too, resulting in a huge issue from lack of proper control. With cloud, the lack of proper and complete security standards [6] also presents a major issue, as does the method of implementing compliance with these standards [52].

The other major issue, particularly when cloud is involved, is that implementing partial solutions, while very effective for the specific task in hand, will not offer a complete solution to the security problem. There are a great many potential weaknesses involved in cloud systems, and without taking a much more comprehensive defensive approach, it will likely prove impossible to effectively secure systems against all the threats faced by enterprises.

Any solution, which only addresses partially the overall security of the enterprise, will be doomed to failure. The only way to be sure a solution will work, will be to properly identify the risks faced by the organization and provide a complete solution to attempt to mitigate those risks, or to accept the particular risks with which the enterprise is comfortable. A fundamental part of this process, which is all too frequently forgotten, is the need to monitor what is happening in the enterprise systems after the software solution is installed, not whenever compliance time comes around, but day to day on an ongoing basis.

There are two major trends developing in the quest to tackle these issues:

1. preventing security breaches,
2. accepting that security breaches will happen and trying to contain breaches.

Traditionally, perimeter security such as firewalls is utilized to prevent attackers from entering the corporate network. The problem with this approach is that this model was well suited to network architectures comprising a few main servers with limited connections between clients and servers. However, as soon as the number of connections explodes, this approach can no longer cope with the demand. This first occurred with the influx of mobile devices and will again become problematic as new Industry 4.0 and IoT devices continue to be implemented and expanded.

We take a look at some of the current solutions available today and will discuss potential weaknesses which pertain to them. The following list shows the most common approaches to tackle cloud cyber security issues:

- Perimeter security (e.g. firewalls) problems with IoT, etc.

  Perimeter Security prevents malicious actors from entering the internal—private—network. Traditionally based on network packet inspection at important network connection points, e.g., routers, by now it includes endpoints such as client computers or mobile devices;

http://dx.doi.org/10.5772/67801

- Sandboxes, e.g. for analysis

  Sandboxes: traditionally anti-virus software utilized signature-based blacklists. Due to newer evasion techniques, e.g., polymorphic payloads, those engines provide insufficient detection rates. Newer approaches utilize sandboxes, e.g., emulate a real system within which the potentially malicious payload is executed and traced. If the payload's behavior seems suspicious, it is quarantined;

- Virtualization

  Virtualization solutions allow program execution within virtual machines. As multiple virtual machines can execute on the same physical host, this allows separation of applications into separate virtual machines. As those cannot interact, this creates resilience in the face of attacks—a successful attack only compromises a single application instead of the whole host. Virtualization is also used to implement sandboxes, easy software deployment, etc.;

- Containers

  Containers: originally used to simplify software deployments. Newer releases of container technologies allow for better isolation between containers. This solves similar problems as virtual machines while improving memory efficiency;

- Software development: secure lifecycles (testing, etc.)

  Software Development: recently security engineering has become a standard part of software development approaches, e.g., Microsoft Secure Development Life-cycle. Mostly, this focuses on early security analysis of potential flaws and attacks and subsequent verification of implemented mitigations;

- Software development: new "safe" languages

  New Programming Languages: as programs are implemented within a given programming language, their security features are very important for the security of the final software product. A good example are common memory errors in programs written in C-based programming languages. Recently, a new generation of programming languages aim to provide both performance and safety—examples of those are Rust, Go or Swift. Rust has seen uptake by the Mozilla community and is being used as part of the next-generation JavaScript engine. Go is used for systems programming and has been adopted by many virtualization management and container management solutions;

- Software development: hardening (i.e., fuzzing)

  Fuzzing: new tooling allows for easy fuzzing of application or operation system code. Automated appliance of this attack technique has yielded multiple high-level memory corruption errors recently;

- Software development: hardening (i.e., libraries)

  Hardened Libraries and Compilers: recent processors provide hardware support for memory protection techniques. Newer compilers allow for automatic and transparent usage of those hardware features—on older architectures, similar features can be implemented in software yielding slightly lower security and performance;

- Architecture: (micro)services

  Microservice Architectures encapsulate a single functionality within a microservice that can be accessed through standardized network protocols. If implemented correctly, each service contains minimal state and little deep interaction with external services. This implicitly implements the Separation of Concern principle and thus helps security by containing potential malicious actors.

The serverless paradigm is also gaining some traction and examples for this software stack can be seen in **Table 5**.

Another recent security battleground is the IoT. Recent examples are the 1Tbps attack against "Krebs on Security" or the 1.2Tbps DDoS attack taking out Dyn in October 2016 [53]. While the generic security landscape is already complex, IoT adds additional problems such as hard device capability restrictions, manifold communication paths and very limited means of updating already deployed systems. In addition, software is often an after-thought; for example, for usage within power-grids, all hardware must be certified with regard to their primary focus (i.e. power distribution). All later alterations, such as security-critical software updates, would void the certification and thus produce substantial costs for the device manufacturer. This leads to a situation where security updates are scarce in the best scenario. Unikernels can improve upon the current situation [12] on both the device as well as at the cloud-level. It should be understood that device level means on the local sensors/actors (problems with security and updates) and cloud level is for scale-out of backend processing of data generated by IoT devices.

Product	Supported languages
Amazon Lambda	Java, Node.js, Python
Google Cloud Functions	Node.js
IBM BlueMix	Languages supported by CloudFountry, e.g. Java, Node.js, Go, C#, PHP, Python, Ruby
Microsoft Azure Functions	C#, F#, Node.js, Python, PHP

Table 5. Example of serverless offerings ©2016 Happe, Duncan and Bratterud.

6. Proposed solutions

The most interesting aspect of unikernels from a security point of view is their inherent minimalism. For the purpose of this chapter, we define unikernels as:

- a minimal execution environment for a service,
- providing resource isolation between those services,
- offering no data manipulation on persistent state within the unikernel, i.e. the unikernel image is immutable,

http://dx.doi.org/10.5772/67801

- being the synthesis of an operating system and the user application,
- only offering a single execution flow within the unikernels, i.e. no multitasking is performed.

The unikernel approach yields an architecture that implicitly follows best software architecture practices, e.g. through the combination of minimalism and single execution flow, the separation of concern principle is automatically applied. This allows for better isolation between concurrent unikernels.

It is absolutely necessary to recognize the magnitude of the dangers posed by the threat environment. Enterprises are bound by legislation, sometimes regulation, the need to comply with standards, industry best practice, and are accountable for their actions. Criminals have no such constraints. They are completely free to bend every rule in the book, do whatever they want, manipulate, cajole, hack or whatever it takes to get to the money. They are constantly probing for the slightest weakness, which they are more than happy to exploit without mercy. It is clear that the threat environment is developing just as quickly as the technological changes faced by the industry [6, 54, 55]. We need to be aware of this threat and minimize the possible impact on our framework. While we have absolutely no control over attackers, we can help reduce the impact by removing as many of the "classic attack vectors" as possible, thus making their life far more difficult. The more difficult it becomes for them to get into the system, the more likely they will be to go and attack someone else's system.

In the interests of usability, many more ports are open by default than are needed to run a system. An open port, especially one which is not needed (and therefore not monitored) is another route in for the attacker. We also take the view that the probability of vulnerabilities being present in a system increases proportionally to the amount of executable code it contains. Having less executable code inside a given system will reduce the chances of a breach and also reduce the number of tools available for an attacker once inside. As Meireles [56] said in 2007 "… while you can sometimes attack what you can't see, you can't attack what is not there!". We address these issues by making the insides of the virtual machine simpler. We also propose to tackle the audit issue by making configuration happen at build time [57, 58], and then making services be "immutable" after deployment, making "configuration lapses" (i.e. through conflicts caused by unnecessary updates to background services etc.) unlikely.

Bearing in mind the success with which the threat environment continually attacks business globally, it is clear that many enterprises are falling down on many of the key issues we have highlighted in Section 3. It is also clear that a sophisticated and complex solution is unlikely to work. Thus, we must approach the problem from a more simple perspective.

6.1. Unikernel impact on efficiency

Cloud achieves maximum utilization and high energy efficiency through consolidating processing power within data centers. Multiple applications run on the same computing node, but often control on node placement or on concurrently running applications are not possible. This means for security, isolation between different applications, users, or services, is critical. A popular but inefficient solution is to place each application or service within a virtual machine [59]. This is very similar to the initial usage of virtualization within host-based

systems; Madnick gives a good overview of the impact of virtualization in Ref. [60]. Containers could present a more efficient approach [61], but as they were originally developed to improve deployment, their security benefits are still open to debate [62].

A useful benefit of the applied minimalism is a reduced memory footprint [63, 64] plus a quick start-up time for unikernel-based systems. Madhavapeddy et al. utilize this for on-demand spin-up of new virtual images [65], allowing for higher resource utilization, leading to improved energy efficiency.

6.2. Unikernel impact on security

The most intriguing aspect of unikernels from a security perspective is their capability as a minimal execution environment. We can now define the attack surface of a system as:

Definition 6.1 attack surface. The amount of bytes within a virtual machine [10].

When it comes to microcode, firmware and otherwise mutable hardware such as field-programmable gate arrays (FPGAs), physical protection can be seen as a gray area. This definition is intentionally kept general in order to allow further specifications to refine the meaning of "physically available" for a given context. The following example can serve to illustrate how the definition can be used for one of many purposes.

Building a classic VM using Linux implies simply installing Linux and then installing the software on top. Any reduction in attack surface must be done by removing unneeded software and kernel modules (e.g. drivers). Take TinyCore Linux as an example of a minimal Linux distribution and assume that it can produce a machine image of 24MB in size.

During the build of a unikernel, minimization is performed, meaning the resulting system image only includes the minimum required software dependencies. This implies that no binaries, shell or unused libraries are included within the unikernel image. Even unused parts of libraries should never be included within the image. This radically reduces included functionality and thus the attack surface. In addition, this can loosen the need for updates after vulnerabilities have been discovered in included third-party components—if the vulnerable function was not included within the set of used functions, an update can be moot.

The situation after a security breach with unikernels is vastly different to traditional systems. Assuming that the attacker is able to exploit a vulnerability, e.g. buffer overflow, he gains access to the unikernel system's memory. Due to having no binaries and only reduced libraries, writing shell code [66], a machine code that is used as a payload during vulnerability execution will not work. Common payloads spawn command shells or abuse existing libraries to give attackers access through unintended possibilities, which is complicated. Pivot attacks depending on shell-access are thwarted. But, all direct attacks against the application, e.g. data extraction due to insecure application logic, are still possible. A good example of this is the recent OpenSSL Heartbleed vulnerability [67]. A unikernel utilizing OpenSSL would also be vulnerable to this, thus allowing an attacker to access its system memory, including the private SSL key. We argue that functionality should be split between multiple unikernels to further compartmentalize breaches.

http://dx.doi.org/10.5772/67801

Next-generation hardware-supported memory protection techniques can benefit from minimalism. For example, the Intel Secure Guard Extensions [68, 69] allow for protected memory enclaves. Accessing these directly is prohibited and protected through specialized CPU instructions. The protection is enforced by hardware, so even the hypervisor can be prevented from accessing protected memory. Rutkowska has shown [70] that deploying this protection scheme for applications has severe implications. Just protecting the application executable is insufficient, as attacks can inject or extract code within linked libraries. This leads us to conclude that the whole application including its dependencies must be part of the secure-memory enclave. Simplicity leads to a "one virtual machine per application" model, which unikernels inherently support. We propose that unikernels are a perfect fit for usage with those advanced memory protection techniques.

Returning to the theme of "*software development frameworks providing sensible defaults but getting bloated, and thus vulnerable over time*", unikernels provide an elegant solution; while the framework should include generic defensive measures, the resulting unikernel will, by definition, only include utilized parts, thus reducing the attack surface.

6.3. Service isolation

A fundamental premise for cloud computing is the ability to share hardware. In private cloud systems, hardware resources are shared across a potentially large organization, while on public clouds, hardware is shared globally across multiple tenants. In both cases, isolating one service from the other is an absolute requirement.

The simplest mechanism to provide service isolation is *process isolation* in classic kernels, relying on hardware supported virtual memory, e.g. provided by the now pervasive x86 protected mode. This has been used successfully in mainframe setups for decades, but access to terminals with limited user privileges has also been the context for classic attack vectors such as stack smashing, root-kits, etc., the main problem being that a single kernel is being shared between several processes and that gaining root access from one terminal would give access to everything inside the system. Consequently, much work was done in the 1960s and 1970s to find ways to completely isolate a service without sharing a kernel. This work culminated in the seminal 1974 paper by Popek and Goldberg [71], where they present a formal model describing the requirements for complete instruction level virtualization, i.e. *hardware virtualization*.

Hardware virtualization was in wide use on e.g. IBM mainframes since that time, but it was not until 2005 that the leading commodity CPU manufacturers, Intel and AMD introduced these facilities into their chips. Meantime, paravirtualization had been reintroduced as a workaround to get virtual machines to run on these architectures, notably in Ref. [72]. While widely deployed and depended upon, the Xen project has recently evolved its paravirtualization interface toward using hardware virtualization in, e.g., PVH [73], stating that *"PVH means less code and fewer Interfaces in Linux/FreeBSD: consequently it has a smaller Trusted Computing Base (TCB) and attack surface, and thus fewer possible exploits"* [74].

Yet another mechanism used for isolation is operating system-level virtualization with containers, e.g. Linux Containers (LXC) popularized in recent years by Docker, where each

container represents a userspace operating environment for services that all share a kernel. The isolation mechanism for this is classic process isolation, augmented with software controls such as cgroups and Linux namespaces. Containers do offer less overhead than classic virtual machines. An example where containers makes a lot of sense would be trusted in-house clouds, e.g. Google is using containers internally for most purposes [75]. We take the position that hardware virtualization is the simplest and most complete mechanism for service isolation with the best understood foundations, as formally described by Popek and Goldberg, and that this should be the preferred isolation mechanism for secure cloud computing.

6.4. Microservices architecture and immutable infrastructure.

Microservices is a relatively new term founded on the idea of separating a system into several individual and fully disjoint services, rather than continuously adding features and capabilities to an ever growing monolithic program. Being single threaded by default, unikernels naturally imply this kind of architecture; any need for scaling up beyond the capabilities of a single CPU should be done by spawning new instances. While classic VMs require a lot of resources and impose a lot of overhead, minimal virtual machines are very lightweight. As demonstrated in Ref. [76], more than 100,000 instances could be booted on a single physical server and Ref. [58] showed that each virtual machine including the surrounding process requires much less memory than a single "Hello World" Java program running directly on the host.

An important feature of unikernels in the context of microservices is that each unikernel VM is fully self contained. This also makes them immune to breaches in other parts of the service composition, increasing the resilience of the system as a whole. Adding to this, the idea of *optimal mutability* (defined below) and each unikernel-based can in turn be as immutable as is physically possible on a given platform. In the next paper in this series, we will expand upon these ideas and take the position that composing a service out of several microservices, each as immutable as possible, enables overall system architects and decision makers to focus on a high-level view of service composition, not having to worry too much about the security of their constituent parts. We take the view that this kind of separation of concerns is necessary in order to achieve scalable yet secure cloud services.

6.5. No shell by default and the impact on debugging and forensics

One feature of unikernels that immediately makes it seem very different from classic operating systems is the lack of a command line interface. This is, however, a direct consequence of the fact that classic POSIX-like CLIs are run as a separate process (e.g. bash) with the main purpose of starting other processes. Critics might argue that this makes unikernels harder to manage and "debug", as one cannot "log in and see what's happened" after an incident, as is the norm for system administrators. We take the position that this line of argument is vacuous; running a unikernel rather corresponds to running a single process with better isolation, and in principle, there is no more need to log in to a unikernel than there is to log in to, e.g., a web server process running in a classic operating system.

It is worth noting that while unikernels by definition are a single-address-space virtual machine, with no concept of classic processes, a read-eval-print loop (REPL) interface can

http://dx.doi.org/10.5772/67801

easily be provided (e.g. IncludeOS does provide an example)—the commands just would not start processes, but rather call functions inside the program. From a security perspective, we take the view that this kind of ad-hoc access to program objects should be avoided. While symbols are very useful for providing a stack trace after a crash or for performance profiling, stripping out symbols pointing to program objects inside a unikernel would make it much harder for an attacker to find and execute functions for malicious and unintended purposes. Our recommendation is that this should be the default mode for unikernels in production mode.

We take the view that logging is of critical importance for all systems, in order to provide a proper audit trail. Unikernels, however, simply need to provide the logs through other means, such as over a virtual serial port or ideally over a secure networking connection to a trusted audit trail store.

Many UNIX period system administrators will require some mental readjustment due to the lack of shell access. On the other hand, the growing DevOps movement [77] abolishes the traditional separation into software development and system administration but places high importance on the communication between and integration of those two areas. Unikernels offer an elegant deployment alternative. The minimized operating system implicitly moves system debugging to application developers. Instead of analyzing errors through shell commands, developers can utilize debuggers to analyze the whole system, which might be beneficial for full-stack engineering.

Lastly, it is worth mentioning that unikernels in principle have full control over a contiguous range of memory. Combined with the fact that a crashed VM by default will "stay alive" as a process from the VMM perspective and not be terminated, this means that in principle the memory contents of a unikernel could be accessed and inspected from the VMM after the fact, if desired. Placing the audit trail logs in a contiguous range of memory could then make it possible to extract those logs also after a failure in the network connection or other I/O device normally used for transmitting the data. Note that this kind of inspection requires complete trust between the owner of the VM and the VMM (e.g. the cloud tenant and cloud provider). Our recommendation would be not to rely on this kind of functionality in public clouds, unless all sensitive data inside the VM is encrypted and can be extracted and sent to the tenant without decrypting it.

6.6. Why use unikernels for the IoT?

Why use unikernels for the IoT [78]? Unikernels are uniquely suited to benefit all areas (sensor, middleman and servers) within the IoT chain. They allow for unified development utilizing the same software infrastructure for all layers. This may sound petty, but who would have thought JavaScript could be used on servers (think node.js) a couple of years ago?

Using hardware virtualization as the preferred isolation mechanism, we take the view that there are three basic approaches we can use to deliver our requirements, namely the monolithic system/kernel approach, the microkernel approach and the unikernel approach. IaaS cloud providers will typically offer virtual machine images running Microsoft Windows or one or more flavors of Linux, possibly optimized for cloud by, e.g., removing device drivers that are not needed. While specialized Linux distributions can greatly reduce the memory footprint

and attack surface of a virtual machine, these are general purpose multi-process operating systems and will by design contain a large amount of functionality that is simply not needed by one single service. We take the position that virtual machines should be specialized to a high degree, each forming a single purpose microservice, to facilitate a resilient and fault tolerant system architecture which is also highly scalable.

In Ref. [11], we discuss six security observations about various unikernel operating systems: choice of service isolation mechanism; use of a single address space, shared between service and kernel; no shell by default and the impact on debugging and forensics; the concept of reduced attack surface; and microservices architecture and immutable infrastructure. We argue that the unikernel approach offers the potential to meet all our needs, while delivering a much reduced attack surface, yet providing exactly the performance we require. An added bonus will be the reduced operating footprint, meaning a more green approach is delivered at the same time.

6.7. For IoT on the client

Unikernels are a kind of virtualization and offer all of its benefits. They provide application developers with a unified interface to diverse hardware platforms, allowing them to focus on application development. They provide the ability to mask changes of the underlying hardware platform behind the hypervisor, allowing for application code to be reused between different hardware revisions. Also, disparate groups within an enterprise often perform system and application development. Use of a unikernel decouples both groups, allowing development in parallel. Application developers can use a virtualized testing environment on their workstations during development, which will mirror the same environment within the production environment.

Unikernels can certainly produce leaner virtual machines compared to traditional virtualization solutions. This results in a much reduced attack surface, which creates applications that are more secure. Use of a resource efficient unikernel, such as IncludeOS, minimizes the computational and memory overhead that otherwise would prevent virtualization from being used. The small memory and processing overhead enables the use of virtualization on low-powered IoT devices and also aids higher capacity devices. Lower resource utilization allows for either better utilization (i.e. running more services on the same hardware) or higher usage of low power modes, reducing energy consumption, both of which increase the sustainability of IoT deployments.

A feature in high demand by embedded systems is atomic updates. A system supporting atomic updates either installs a system update or reverts back to a known working system state. For example, Google's Chrome OS [79] achieves this by using two system partitions. A new system upgrade is installed on the currently unused partition. On next boot, the newly installed system is now used, but the old system is preselected as a backup boot option if this boot does not work. If the new system boots, the new system becomes the new default operating system and the (now) old partition will be used for the next system upgrade.

This delivers high resilience in the face of potentially disrupting Chrome OS updates. A similar scheme is set to be introduced for the forthcoming Android Version 7. This scheme would be greatly aided by unikernels, as they already provide a clear separation of data and control

logic. A system upgrade would therefore start a new unikernel and forward new requests to it. If the underlying hypervisor has to be upgraded, likely a very rare event, the whole system might incorporate the dual boot-partition approach.

6.8. For IoT on the server

Taking account of the large estimated number of IoT devices to be deployed in the near future, computational demand on the cloud can be immense. While IoT amplifies the amount of incoming traffic, it has some characteristics that should favor unikernel-like architectures.

Our envisioned unikernels utilize a non-mutable state and are event-based. This allows simplified scale-out, i.e. it allows for dynamically starting more unikernels if incoming requests demand it. We believe that many processing steps during an IoT dataflow's lifetime will be parallelizable, e.g. data collected from one household will not interact with data gathered by a different household from another continent during the initial processing steps, or possibly never at all. Since they do not interact, there is no chance of side effects, thus the incoming data can instantly be processed by a newly spawned unikernel.

Two recent trends in cloud computing are cloudlets and fog computing. The first describes a small-scale data center located near the Internet's edge, i.e. co-located near many sensors and acting as the upstream collection point for the incoming IoT sensors, while the second describes the overall technique of placing storage or computational capabilities near the network edges. A unified execution environment is needed to allow easy use of this paradigm. When the same environment is employed, application code can easily be moved from the cloud toward the networks' edge, i.e. into the cloudlets. Unikernels offer closure over the application's code, so the same unikernel can be re-deployed at a cloudlet or within a central data center.

The unikernel itself might place requirements on external facilities such as storage, which would need to be provided by the current execution environment. A consumer-grade version of this trend can already be seen in many high-powered NAS devices, which allow for local deployment of virtual machines or containers. This moves functionality from the cloud to a smallest-scale local processing environment. A good use case for this would be Smart Homes; here, a local NAS can perform most of the computations and then forward the compressed data toward a central data center. Also, this local preprocessing can apply various cryptographic processes to improve the uploaded data's integrity or confidentiality.

7. Conclusions

We have taken a good hard look at cyber security in the cloud, and in particular, we have considered the security implications of the exciting new paradigm of the IoT. While the possibilities are indeed exciting, the consequences of getting it wrong are likely to be catastrophic. We cannot afford to carry blindly on. Instead, we must recognize that if the issues we have outlined on security and privacy are not tackled properly, and soon, we will all be sleepwalking into a disaster. However, if we realize that we need to take some appropriate actions

now, then we will be much better placed to feel comfortable in living in an IoT world. There are considerable potential benefits for everyone to be offered from using our unikernel-based approach. While we see security and confidentiality of data as paramount—and given the forthcoming EU's GDPA, we believe the EU agrees. Security and privacy do not directly translate into a direct monetary benefit for companies and thus are seldom given enough incentive for change to allow serious improvement to gain traction. To better convince enterprises, we offer the added benefit of increased developer efficiency. Experienced and talented developer resources are scarce at hand, so making the most of them is in an enterprise's best interest. The broad application of a virtualization solution allows them to better reuse existing knowledge and tools, as developers gain a virtual long-term environment that they can work in.

Virtualization in combination with the special state-less nature of many unikernels provides a solution for short-term processing spikes. Processing can be scaled-out to in-company or public clouds by deploying unikernels as they do not require external dependencies and as they do not contain state, deployments are simplified. After their usage, they can be discarded (no state also means that no compromising information is stored at the cloud provider). In the case of sensitive information, special means, e.g. homomorphic encryption or verifiable computing technologies need to be employed to protect data integrity or confidentiality.

Unikernels offer a high energy efficiency. This allows companies to claim higher sustainability for their solutions while reducing their energy costs. We view our proposed solution as taking a smart approach to solving smart technology issues. It does not have to be exorbitantly expensive to do what we need, but by taking a simple approach, sensibly applied, we can all have much better faith in the consequences of using this technology (as well as having the comfort of being able to walk through a smart city without having our bank account emptied).

Author details

Bob Duncan[1]*, Andreas Happe[2] and Alfred Bratterud[3]

*Address all correspondence to: bobduncan@abdn.ac.uk

1 Computing Science, University of Aberdeen, Aberdeen, UK

2 Department of Digital Safety & Security, Austrian Institute of Technology GmbH, Vienna, Austria

3 Department of Computer Science, Oslo and Akershus University, Oslo, Norway

References

[1] Duncan, Bob, and Mark Whittington. "Enhancing Cloud Security and Privacy: The Power and the Weakness of the Audit Trail."Cloud Comput (2016): 125-130. Publisher: IARIA, ISBN: 978-1-61208-460-2

[2] PWC, "UK Information Security Breaches Survey - Technical Report 2012," PWC2012, Tech. Rep., April, 2012.

[3] R. E. Crossler, A. C. Johnston, P. B. Lowry, Q. Hu, M. Warkentin, and R. Baskerville, "Future directions for behavioral information security research," Comput. Secur., 2013, vol. 32, pp. 90-101.

[4] G. T. Willingmyre, "Standards at the crossroads," StandardView, vol. 5, no. 4, pp. 190-194, 1997.

[5] Cisco, "2013 Cisco annual security report," Cisco, Tech. Rep., 2013. [Online]. Available: http://grs.cisco.com/grsx/cust/grsCustomerSurvey.html?SurveyCode=4153 ad_id=US-BN-SEC-M-CISCOASECURITYRPT-ENT KeyCode=000112137 Last Accessed: 5 Jan 2017.

[6] 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.

[7] J. F. Gantz, D. Reinsel, C. Chute, W. Schlichting, J. McArthur, S. Minton, I. Xheneti, A. Toncheva, and A. Manfrediz, "The expanding digital universe: a forecast of worldwide information growth through 2010," in Extern. Publ. IDC Analyse Futur. Inf. Data. IDC, 2007, pp. 1-21. [Online]. Available: https://www.tobb.org.tr/BilgiHizmetleri/Documents/Raporlar/Expanding_Digital_Universe_IDC_WhitePaper_022507.pdf Last Accessed: 5 December 2016

[8] Evans, Dave. "The internet of things: How the next evolution of the internet is changing everything." CISCO white paper 1.2011 (2011): 1-11.

[9] 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.

[10] A. Bratterud, A. Happe, and B. Duncan, "Enhancing cloud security and privacy: the Unikernel solution," in Accept. CloudComput. 2017, 2017, pp. 1-8.

[11] A. Happe, B. Duncan, and A. Bratterud, "Unikernels for cloud architectures: how single responsibility can reduce complexity, thus improving enterprise cloud security," in Submitt. to Complexis 2017, 2016, pp. 1-8.

[12] B. Duncan, A. Happe, and A. Bratterud, "Enterprise IoT security and scalability: How Unikernels can improve the status Quo," in 9th IEEE/ACM Int. Conf. Util. Cloud Comput. (UCC 2016), Shanghai, China, 2016, pp. 1-6.

[13] Duncan, Bob, and Mark Whittington. "Creating an Immutable Database for Secure Cloud Audit Trail and System Logging." CLOUD COMPUTING 2017(2017): pp54-59. Publisher: IARIA, ISBN: 978-1-61208-529-6

[14] B. Duncan and M. Whittington, "Enhancing cloud security and privacy: the cloud audit problem," in Cloud Comput. 2016: 7th Int. Conf. Cloud Comput., GRIDs, Virtualization, Rome, 2016, pp. 1-6.

[15] M. Huse, "Accountability and creating accountability: a framework for exploring behavioural perspectives of corporate governance," Br. J. Manag., March 2005, vol. 16, no. S1, pp. S65–S79.

[16] A. Gill, "Corporate governance as social responsibility: a research agenda," Berkeley J. Int'l L., 2008, vol. 26, no. 2, pp. 452-478.

[17] 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.

[18] A. Kolk, "Sustainability, accountability and corporate governance: exploring multinationals' reporting practices." Bus. Strateg. Environ., 2008, vol. 17, no. 1, pp. 1-15.

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

[20] S. Arjoon, "Corporate governance: an ethical perspective," J. Bus. Ethics, November 2012, vol. 61, no. 4, pp. 343-352.

[21] OWASP, "OWASP Top Ten Vulnerabilities 2013," 2013. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project Last Accessed: 5 Jan 2017.

[22] C. Climate, "Rails' Remote Code Execution Vulnerability Explained," 2013. [Online]. Available: http://blog.codeclimate.com/blog/2013/01/10/rails-remote-code-execution-vulnerability-explained/ Last Accessed: 5 Jan 2017.

[23] A. Blankstein and M. J. Freedman, "Automating isolation and least privilege in web services," in Secur. Priv. (SP), 2014 IEEE Symp. IEEE, 2014, pp. 133-148.

[24] OWASP, "OWASP Top 10 IoT Vulnerabilities (2014)," 2014. [Online]. Available: https://www.owasp.org/index.php/Top_10_IoT_Vulnerabilities_(2014) Last Accessed: 5 Jan 2017.

[25] USA Today, "Hackers Breach US Dept of Energy Copmputers 150 Times in 4 Years, Including 19 Nuclear Breaches," 2015. [Online]. Available: http://www.usatoday.com/story/news/2015/09/09/cyber-attacks-doe-energy/71929786/Last Accessed: 5 Jan 2017.

[26] Wired, "German Steel Mill HackedCausing Massive Damage," 2015. [Online]. Available: https://www.wired.com/2015/01/german-steel-mill-hack-destruction/Last Accessed: 5 Jan 2017.

[27] SecurityWeek, "FDA Issues Alert Over Vulnerable Hospira Drug Pumps," 2015. [Online]. Available: FDA Issues Alert Over Vulnerable Hospira Drug Pumps Last Accessed: 5 Jan 2017.

[28] DailyMail, "Security Expert Who 'Hacked a Commercial Flight and made it Fly Sideways' Bragged that he also Hacked the International Space Station," 2015. [Online]. Available: http://www.dailymail.co.uk/news/article-3090288/Security-expert-admitted-FBI-took-control-commercial-flight-bragged-hacker-convention-2012-playing-International-Space-Station-getting-yelled-NASA.html Last Accessed: 5 Jan 2017.

[29] CBR, "IoT Security Breach Forces Kitchen Devices to Reject Junk Food," 2015. [Online]. Available: http://www.cbronline.com/news/internet-of-things/consumer/iot-security-breach-forces-kitchen-devices- to-reject-junk-food-4544884 Last Accessed: 5 Jan 2017.

[30] FCA, "Fines Table - 2014," 2014. [Online]. Available: http://www.fca.org.uk/firms/being-regulated/enforcement/fines Last Accessed: 5 Jan 2017.

[31] G. J. Popek and R. P. Goldberg, "Formal Requirements for Virtualizable Third Generation Architectures," ACM SIGOPS Oper. Syst. Rev., 1973, vol. 7, no. 4, p. 112.

[32] J. Goodacre, "No Title," in Virtualization Euro Work. 2009, 2009 [Online]. Available: ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/computing/virtualization-euro-workshop-29-9-09-john-goodacre-arm_en.tif Last Accessed: 5 Jan 2017.

[33] Dall, Christoffer, and Jason Nieh. "KVM/ARM: the design and implementation of the linux ARM hypervisor." Acm Sigplan Notices. Vol. 49. No. 4. ACM, 2014.

[34] C. Dall and J. Nieh, "Supporting KVM on the ARM Architecture," 2013. [Online]. Available: https://lwn.net/Articles/557132/Last Accessed: 5 Jan 2017.

[35] C. Dall and J. Nieh, "KVM/ARM: the design and implementation of the linux ARM hypervisor," in ACM SIGPLAN Not., 2014, vol. 49, no. 4. ACM, pp. 333-348.

[36] Imgtech, "MIPS Virtualization," 2016. [Online]. Available: https://imgtec.com/mips/architectures/virtualization/Last Accessed: 5 Jan 2017.

[37] I. Technologies, "The MIPS Architecture and Virtualization," 2016. [Online]. Available: https://imagination-technologies-cloudfront-assets.s3.amazonaws.com/mips-downloads/m51xx/The-MIPS-architecture-and-virtualization-for-web-download.tif Last Accessed: 5 Jan 2017.

[38] Riskv.org, "Open-Source RISK V Architecture," 2016. [Online]. Available: https://riscv.org/Last Accessed: 5 Jan 2017.

[39] S. Sharma, V. Chang, U. S. Tim, J. Wong, and S. Gadia, "Cloud-based Emerging Services Systems," Int. J. Inf. Manage., 2016, pp. 1-19.

[40] Verizon, "2014 Data Breach Investigations Report," Tech. Rep. 1, 2014. [Online]. Available: http://www.verizonenterprise.com/resources/reports/rp_Verizon-DBIR-2014_en_xg.tif Last Accessed: 5 Jan 2017.

[41] PWC, "2014 Information Security Breaches Survey," Tech. Rep., 2014. [Online]. Available: https://www.pwc.co.uk/assets/pdf/cyber-security-2014-technical-report.pdf Last Accessed: 5 December 2016

[42] Trustwave, "2013 Global Security Report," Trustwave, Tech. Rep., 2013. [Online]. Available: https://www.trustwave.com/Resources/Library/Documents/2013-Trustwave-Global-Security-Report/ Last Accessed: 5 December 2016

[43] Verizon, "2010 Data Breach Investigation Report: A study conducted by the Verizon RISK Team in cooperation with the United States Secret Service," Verizon/USSS, Tech.

Rep., 2010. [Online]. Available: http://www.verizonenterprise.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf Last Accessed: 5 December 2016

[44] Verizon, "2011 Data Breach Investigation Repeort: A study conducted by the Verizon RISK Team in cooperation with the United States Secret Service and Others," Verizon/USSS, Tech. Rep., 2011. [Online]. Available: http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2011_en_xg.pdf Last Accessed: 5 December 2016

[45] 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.tif Last Accessed: 5 Jan 2017.

[46] Verizon, "2013 Data Breach Investigation Report: A study conducted by the Verizon RISK Team in cooperation with the United States Secret Service and Others," Verizon, Tech. Rep., 2013.

[47] Verizon, "2016 Data Breach Investigations Report," Tech. Rep. 1, 2016. [Online]. Available: http://www.verizonenterprise.com/resources/reports/rp_DBIR_2016_Report_en_xg.tif Last Accessed: 5 Jan 2017.

[48] FTC, "ASUS Settles FTC Charges That Insecure Home Routers and "Cloud" Services Put Consumers' Privacy At Risk," 2016. [Online]. Available: https://www.ftc.gov/news-events/press-releases/2016/02/asus-settles-ftc-charges-insecure-home-routers-cloud-services-put Last Accessed: 5 Jan 2017.

[49] C. Brook, "FTC: D-Link Failed to Secure Routers, IP Cameras," 2017. [Online]. Available: https://threatpost.com/ftc-d-link-failed-to-secure-routers-ip-cameras/122895/Last Accessed: 5 Jan 2017.

[50] D. Kravets, "Unsecure routers, webcams prompt feds to sue D-Link," 2017. [Online]. Available: http://arstechnica.com/tech-policy/2017/01/unsecure-routers-webcams-prompt-feds-to-sue-d-link/Last Accessed: 5 Jan 2017.

[51] B. Schneier, "Regulation of the Internet of Things," 2016. [Online]. Available: https://www.schneier.com/blog/archives/2016/11/regulation_of_t.html Last Accessed: 5 Jan 2017.

[52] B. Duncan and M. Whittington, "Reflecting on whether checklists can tick the box for cloud security," in Cloud Comput. Technol. Sci. (CloudCom), 2014 IEEE 6th Int. Conf. Singapore: IEEE, 2014, pp. 805-810.

[53] Guardian, "Can We Secure the Internet of Things in Time to Prevent Another Cyber-Attack?," 2016. [Online]. Available: https://www.theguardian.com/technology/2016/oct/25/ddos-cyber-attack-dyn-internet-of-things Last Accessed: 5 Jan 2017.

[54] 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.

[55] B. Duncan and M. Whittington, "Information security in the cloud: should we be using a different approach?" in 2015 IEEE 7th Int. Conf. Cloud Comput. Technol. Sci., Vancouver, 2015, pp. 1-6.

[56] P. Meireles, "Narkive Mailinglist Archive," 2007. [Online]. Available: http://m0n0wall.m0n0.narkive.com/OI4NbHQq/m0n0wall-virtualization Last Accessed: 5 Jan 2017.

[57] A. Madhavapeddy, R. Mortier, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, and J. Crowcroft, "Unikernels: library operating systems for the cloud," in ASPLOS '13 Proc. 18th Int. Conf. Archit. Support Program. Lang. Oper. Syst. vol. 48, 2013, pp. 461-472.

[58] A. Bratterud, A.-A. Walla, H. Haugerud, P. E. Engelstad, and K. Begnum, "IncludeOS: a minimal, resource efficient Unikernel for cloud services," in 2015 IEEE 7th Int. Conf. Cloud Comput. Technol. Sci., pp. 250-257, 2015.

[59] R. Jithin and P. Chandran, "Virtual machine isolation," in Int. Conf. Secur. Comput. Networks Distrib. Syst. Springer, 2014, pp. 91-102.

[60] S. E. Madnick and J. J. Donovan, "Application and analysis of the virtual machine approach to information system security and isolation," in Proc. Work. virtual Comput. Syst. ACM, 1973, pp. 210-224.

[61] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors," in ACM SIGOPS Oper. Syst. Rev., vol. 41, no. 3. ACM, New York, 2007, pp. 275-287.

[62] T. Bui, "Analysis of docker security," arXiv Prepr. arXiv1501.02967, 2015.

[63] A. Bratterud and H. Haugerud, "Maximizing hypervisor scalability using minimal virtual machines," in Cloud Comput. Technol. Sci. (CloudCom), 2013 IEEE 5th Int. Conf., vol. 1. IEEE, New York, 2013, pp. 218-223.

[64] A. Bratterud, A.-A. Walla, P. E. Engelstad, K. Begnum, and Others, "IncludeOS: a minimal, resource efficient unikernel for cloud services," in 2015 IEEE 7th Int. Conf. Cloud Comput. Technol. Sci., IEEE, 2015, pp. 250-257.

[65] A. Madhavapeddy, T. Leonard, M. Skjegstad, T. Gazagnaire, D. Sheets, D. Scott, R. Mortier, A. Chaudhry, B. Singh, J. Ludlam, and Others, "Jitsu: just-in-time summoning of unikernels," in 12th USENIX Symp. Networked Syst. Des. Implement. (NSDI 15), 2015, pp. 559-573.

[66] I. Arce, "The shellcode generation," IEEE Secur. Priv., 2004, vol. 2, no. 5, pp. 72-76.

[67] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, and Others, "The matter of heartbleed," in Proc. 2014 Conf. Internet Meas. Conf., ACM, New York, 2014, pp. 475-488.

[68] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU based attestation and sealing," in Proc. 2nd Int. Work. Hardw. Archit. Support Secur. Priv., vol. 13, 2013.

[69] V. Costan and S. Devadas, "Intel sgx explained," Cryptology ePrint Archive, Report 2016/086, 2016. https://eprint. iacr. org/2016/086, Tech. Rep. Last Accessed: 5 Jan 2017.

[70] J. Rutkowska, "Thoughts on Intel's upcoming Software Guard Extensions (Part 1)," 2013. http://theinvisiblethings.blogspot.co.at/2013/08/thoughts-on-intels-upcoming-software. html. Last Accessed: 5 Jan 2017.

[71] G. J. Popek and R. P. Goldberg, "Formal requirements for virtualizable third generation architectures," Commun. ACM, 1974, vol. 17, no. 7, pp. 412-421.

[72] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," ACM SIGOPS Oper. Syst. Rev., 2003, vol. 37, p. 164.

[73] D. Chisnall, "Xen PVH: Bringing Hardware to Paravirtualization." Inf. IT, 2014. [Online]. Available: http://www.informit.com/articles/article.aspx?p=2233978 Last Accessed: 5 December 2016

[74] X. Project, "Xen Project Software Overview," 2015.

[75] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in Proc. 10th Eur. Conf. Comput. Syst. - EuroSys '15, 2015, pp. 1-17.

[76] A. Bratterud and H. Haugerud, "Maximizing hypervisor scalability using minimal virtual machines," in 2013 IEEE 5th Int. Conf. Cloud Comput. Technol. Sci., 2013, pp. 218-223.

[77] L. Bass, I. Weber, and L. Zhu, DevOps: A Software Architect's Perspective. Addison-Wesley Professional, Boston, 2015.

[78] R. Pavlicek, "Unikernel-based Microservices will Transform the Cloud for the IoT ge," 2016. [Online]. Available: http://techbeacon.com/unikernel-based-microservices-will-transform-cloud-iot-age Last Accessed: 5 Jan 2017.

[79] Google, "Google Chrome OS," 2015. [Online]. Available: https://www.chromium.org/chromium-os Last Accessed: 5 Jan 2017.