ABSTRACT

Cloud security has become of paramount importance in the past few years with the ever increasing number of news headlines describing the latest cyber attacks, data breaches and even unauthorized governmental surveillance. SaaS systems tailored for 3D rendering such as 3drepo.io or webVis/instantHub3D store commercially sensitive 3D graphical information in their respective cloud-based architectures that are potentially vulnerable to such attacks. This paper, therefore, introduces general concepts as well as detailed considerations that should be taken into account when building a large-scale cloud-based Software as a Service (SaaS) data storage and visualization platform. Firstly, different levels of security threats that need to be addressed are identified. Next, five server configuration topologies with increasing level of complexity, scalability and data security are presented. These are contrasted on the basis of cost, utility, complexity and implicit security. Example implementations together with security improvements such as role-based access are developed in the Service Store and Intelligent Cloud Protection systems by British Telecom (BT). Finally, general guidelines and lessons learned are summarized and recommendation of the best practice is presented.

CCS Concepts
• Information systems → RESTful web services; • Computing methodologies → Graphics file formats; • Networks → Web protocol security;

Keywords
security, encryption, 3D Repo, REST

1. INTRODUCTION

Systems such as 3drepo.io [22] or webVis/instant3DHub [5], store detailed 3D models and metadata for various types of industrial applications. These are accessible in modern web browsers over the Internet so, obviously, there needs to be a level of protection from unauthorized access which previously has only been discussed in passing. The aim of this work is, therefore, to establish the best practice and implementation guidelines for cyber-security in such cloud-based asset management systems that is of high importance not only to the wider user base of industrial professionals who need to collaborate on Computer Aided Design (CAD) and Building Information Modelling (BIM). In the UK in 2011, the Cabinet Office mandated that all centrally procured construction projects use a Common Data Environment (CDE) by April 2016. This mandate was detailed in a series of five publicly available specifications (PAS) 1192 documents developed by the British Standards Institution as commissioned by the Centre for the Protection of National Infrastructure (CPNI). The latest in the series, PAS 1192-5 [27], is specifically devoted to cyber-security in information modelling and smart asset management. In this context, the assets represent 2D engineering drawings, 3D models, their associated metadata, bills of quantities, scope and schedules of work, maintenance manuals, etc. Such information is as relevant to the design, construction and tendering stages of a project, as it is to the ongoing maintenance and management of the built environment long past the delivery and handover stages. The same approach was also ratified by the European Public Procurement Directive [19] referencing the use of BIM in public construction. Since then, the
strategy has been adopted by countries in the wider European Economic Area (EEA) such as Netherlands, Denmark, Finland and Norway.

Understandably, 3D Repo aims to fulfil the governmental mandate by providing a centralised online repository 3drepo.io for storing and disseminating 3D models with semantic metadata and relevant documentation as required by the industry. With this in mind, providing a trusted cloud platform is of utmost importance to the commercial viability of any such a web-based platform. An intercepted transmission or breach of storage containing detailed engineering models for projects such as public train stations, highways\textsuperscript{1} or nuclear power plants would provide hostiles with potentially classified information. In addition, a single, centralized or multi-tenant repository would mean that different clients are storing their information in the same virtual system, yet expect complete competitor data segregation. Fig. 1 depicts a sample project which contains potentially sensitive information that needs to be stored and accessed in a secure manner. On this specific project, the design coordination across four different companies happened weekly. This alone, over the duration of a single month, generated 74.7 GB of data. Nonetheless, despite the requirement of non-disclosure agreements being entered, some consultants would exchange unencrypted information via emails including download links and passcodes. Providing them with a secure online access and data uploads would almost certainly help alleviate some but not all such vulnerabilities, see §6.

Contributions.
Our main contributions are as follows:
1. Identification of cyber-security threats faced by a SaaS host storing commercially sensitive data.
2. Definition of five multi-tier application architectures.
3. Evaluation of the proposed topologies in terms enhanced security but also commercial viability & maintainability.
4. Deployment of 3drepo.io with cloud security features such as BT Intelligent Protection addressing the above threats.
5. Security recommendations for general SaaS hosting.

Disclaimer.
Due to obvious security reasons, no specific port numbers, actual server layouts or settings are being described in this paper. Instead, provided is a high-level overview and a discussion of the relative merits of various systems without specific implementation details. For instance, MongoDB’s mongod (primary) and mongos (shard) daemons by default operate on a well-known 27,107 port number, while MongoDB’s status page runs always on port 1,000 greater than the daemon itself [16]. It is, therefore, advisable to change the default settings as those tend to be attempted by the attackers first regardless of what the default settings actually are. After all, they can also read the documentation same as anyone else. Similarly, the root access, various compilers and even package management tools such as yum or npm should be disabled altogether. These and other such suggestions presented throughout the paper should provide basic guidance when building a trusted cloud solution for commercial data hosting. Nevertheless, this text is not intended as an exhaustive security advice and other relevant publications and experts should always be consulted.

\textsuperscript{1}A556 Highway: https://youtu.be/D2uTyIzI9rQ

2. RELATED WORK

Following the release of various standards and publicly available specifications [24, 25, 26, 28] that set out requirements for collaborative information management in design, capital and operational stages of construction projects, it became necessary to also address the security aspects of such systems. The British Standards Institution together with key UK players such as CPNI, the Atomic Weapons Establishment, Metropolitan Police, Ministry of Justice and Construction Industry Council amongst others, highlighted main vulnerabilities, potential security threats and best practice for asset owners in physical as well as virtual built environment. PAS 1192-5 [27] states that good security requires a good risk assessment in order to identify likely threats and how best to mitigate them given a limited budget. There, the main security risks have been summarized as hostile reconnaissance, i.e. a third-party looking for behavioral patterns of an individual or a group, exploiting security vulnerabilities in the system or its configuration, and malicious acts, i.e. unauthorized deletion, usage or disclosure of intellectual property, commercially sensitive or personal data. Obviously, a well secured data hosting offering supporting collaborative aspects of CAD and BIM 3D modelling can be advantageous to commercial enterprises. Nevertheless, data leaked into the public domain cannot be removed easily so extra precautions preventing industrial espionage, cybercrime and terrorist acts have to be taken into account.

Web3D cyber-security.
Over the past years, there have been many significant technological advancements in Web3D community, attracting commercial interest in providing Software as a Service (SaaS) solutions reliant on network data storage and transmission. For instance, Mouton et al. [17] presented a scalable model-driven web service targeting 3D CAD in web browsers. Later, Behr et al. [5] introduced the webVist/instant3D Hub platform as an evolution of such a system, which combines a Visual Computing as a Service infrastructure along with a component-based framework for delivery of interactive 3D rendering over the Internet. This allows applications to compose and manipulate complex data setups with simple commands. Similarly, [3] demonstrated the bimserver.org project: an open source model server targeting BIM through the the open Industry Foundation Classes (IFC) [15] data format. However, neither of these have discussed security implications of their respective architectures. In contrast, [21] dedicated an entire book to security within virtual worlds and web-based 3D visualizations. Whilst the main focus there is on virtual communities and social behavior, the latter chapters address security issues when implementing a virtual world as a service.

3D Repo.
3D Repo is an open source Version Control System (VCS) for various types of 3D assets that was introduced by Dobos and Steed [11]. Since then, the system underwent several iterations and improvements and became a RESTful service, firstly as XML3DRepo [10] with integral support for XML3D rendering [23], and later as a fully-fledged commercial solution 3drepo.io [22] with support for X3DOM [4] rendering. In the back-end, the system is built on top of a NoSQL database MongoDB. The use of a polymorphic big
data store enabled the creation of a domain-specific VCS in order to persistently preserve 3D scenes consisting not only of standard assets such as meshes, materials, textures, etc. but also of additional metadata nodes, PDF drawings, original binary data formats, external resources and so on. All of these are then tracked over time regardless of their type or point of origin. In such a setup, each company is assigned their own database to manage a portfolio of projects. There, a project constitutes several collections—tables in relational databases terms—representing different facets of information regarding version controlled data stored within. Each node is a binary blob with unique identifiers to designate individual data objects and shared identifiers to track those objects over time. Such nodes are connected into a Labelled Directed Acyclic Graph (LDAG) representation which works equally well for scene graphs and revision histories. In [22], Scully et al. explained how 3D Repo can be set up with basic security considerations. This paper aims to extend that work with a main focus on overall topologies and network configurations by deploying 3D Repo onto BT Service Store [9] cloud-based system.

Cloud market places.

The idea of cloud market places is becoming more and more popular due to the increased demand for SaaS-based applications on the web. Utilizing a market place offers benefits in that the application developer is no longer the same as application provider, relieving them from operational overheads that might be challenging especially in the case of small and medium enterprises (SMEs). Cloud market places are generally hosted by a single provider, e.g. Amazon Web Services [1], or by a cloud broker who supports deployments to multiple providers simultaneously [31]. Juju [7], for example, enables the flexibility of deploying to multiple cloud providers offering easy management and horizontal scalability. Common asset management methods such as Docker [12], Chef [8] and Puppet [20] are supported for easy portability what fits various deployment and configuration needs. In such systems, the end user is also given the peace of mind that the application they are deploying is of a quality that is accepted by the market place. What is more, the service management is done directly by a party specialised in network services what should reduce operational and security risks. BT provides Infrastructure as a Service (IaaS) through the BT Cloud Compute product line. As with other cloud service providers (CSPs), the security of this infrastructure is critical in maintaining and growing their user base. Whilst CSPs focus on securing their infrastructure, the security of customer applications and data are typically the responsibility of the individual customers, see §4.

Cyber Security information sharing.

In the recent years, a lot of work has been focused on threat information sharing amongst communities. Recent works has been contributing into systems and platforms to share or distribute threat intelligence, c.f. [2, 6, 18]. For instance, Woods et al. [29] discussed how applying data mining and statistical learning methods can allow a better, more inclusive sharing effort. In contrast, Zhao et al. [30] focused on devising various models of information and resource sharing in a IaaS cloud. The description of multi-tenant, collaborating participants models holds many similarity to the model used within 3D Repo.

Figure 2: Project roles in 3D Repo GUI. Individual color-coded roles are assigned their respective high-level read/write privileges which are automatically translated into fine-grained operations such as find, insert, update, etc. at the collection level.

3. APPLICATION-LEVEL SECURITY

In 3D Repo, the use of a database per company provides a natural Chinese wall segregation preventing data from being accessed by unauthorized entities using the same SaaS platform. Access to the data is provided via a public facing REST API service. Users are authenticated with cookies being served after a successful challenge-response authentication at the database level. All information is exchanged via up to 2048-bit extended validation Secure Sockets Layer (SSL) transmission channel providing encryption security with a green bar validation in the web browser, see Fig. 1.

The user credentials themselves are stored in the DB-protected admin database, greatly simplifying the implementation by leveraging MongoDB’s built-in verification functionality. This supports x.509 certificate authentication as well as enterprise-level Kerberos and Lightweight Directory Access Protocol (LDAP). As the client visualisation service is built directly atop the REST API, security concerns are automatically handled through the same authentication processes. In [22], each of the user accounts stored a portfolio list of projects they had access to, which was checked at the REST API level before serving and receiving project data. Since a user “claiming” to have access to a project was obviously not sufficient from a security standpoint, privileges had to be checked against a record in the project settings. This effectively meant that the list of accessible projects stored by the user was conceptually no different to ordinary web-browser bookmarks. What is more, it introduced duplicity of records in DB which was not only wasteful, it also made the setting up of permissions rather cumbersome. Instead, we rely on custom roles stored independently of the users or the projects. As shown in Fig. 2, custom roles are created in a database belonging to a specific company, although, same as the users, the roles are actually stored in the DB-protected admin database within MongoDB. These color-coded roles are then assigned high-level read/write per-
BT Service Store

4.1 BT Service Store

The aforementioned challenges inspired the development of BT Service Store, which provides several capabilities to address them. Its main goals can be summarized as follows:

1. Provide control mechanisms for protecting hosts, applications and data in the cloud.
3. Allow tools to be selected and bound into complex architectures or application stacks.
4. Simplify and automate the protection and compliance processes.

Core utility.

Primary consumables for the Service Store are pre-packaged applications, virtual private clouds (VPC) and vanilla servers, i.e. virtual machines (VMs) with a chosen base operating system image. These are all deployable in combination as a workload to a chosen target cloud infrastructure. Predictably, every workload has three phases in its life cycle: i) its creation, ii) in-life and iii) decommissioning phase. Any new functionality has to adhere to and be synchronized with the life-cycle of workloads that it enhances. Hence, the common services architecture design has taken into account several requirements. Firstly, it must be available as a subscription service for use by Service Store tenants. Next, its operations need to be in sync with the life cycle of a workload in the Service Store. Finally, it must be manageable/configurable from within the workload life cycle, if required. Thus, BT cloud service architecture has been designed for integration such that the integrator of a common service needs to create an “extension services middleware” with predefined webhooks. These are then made available to be called by the Service Store to trigger a particular function or to retrieve a block of information. The webhooks carry out functions in the following categories:

- Extension service description including name, version and logo.
- Recipe declaration for agent installation or configuration scripts.
- Virtual machine and VPC state description to include VM information and VPC subnet level configuration requirements like ports that needs to be open in the firewall.
- Common services life cycle management such as user registration, configuration and maintenance action sequences.
- Common services REST API translation for registration and configuration of tenant specific information on corresponding common service management server.

Fig. 4 depicts the architecture of these interactions for two example common services, BT Intelligent Protection and BT Mail Protection. Intelligent Protection, see §4.2, is an application and host protection with a security control functioning at operating system (OS) kernel level, whereas Mail Protection is an email security service which acts as a network gateway to an email server. Both are forms of security services but with different life cycles with respect to a workload. In this regard, the middleware is further designed to translate calls made to the web hooks to the management server of the corresponding common service. As part of the deployment and management of the middleware, Docker containers are used for remote delivery and the middleware itself can be written in any programming language as long as it complies with being REST APIs.

4.2 BT Intelligent Protection

BT Intelligent Protection, shown in Fig. 5, has been designed and developed to address demand for protecting vir-
come selectable properties of any application stack that the user chooses to assemble regardless of cloud platform or infrastructure. Servers, physical or virtual, running remotely as well as locally can be connected to the protection service by installing an agent, i.e. a custom security daemon service, onto the machines. The usage of an agent-based model allows the administrators to monitor and manage their protection configurations remotely. Through this integration, the Intelligent Protection offers a new experience to seamlessly manage security as follows:

1. **Policy enforcement**: The mechanism to manage the overall protection of the system, based on pre-defined policies. It can either be an agent installed on a virtual machine, or a dedicated appliance, e.g. virtual or otherwise, with a plug-in to communicate with the internal cloud.

2. **Policy administration**: The mechanism used for defining, updating and enabling security policies based on a library of rules. This includes virtual patches for specified systems and applications; firewalls and protocol restrictions.

3. **Threat intelligence**: The mechanism used for enriching primitive rules for identifying attacks, virus signatures or vulnerabilities. This connects the system to a network of knowledge provided by a vast number of security and application vendors.

### 4.3 Data Protection as a Service (DPaaS)

Data Protection as a Service (DPaaS) is the result of ongoing contributions to the ESCUDO-CLOUD project [13] and is being realised as a technical use case by BT. DPaaS provides cloud computing users with not only the basic data encryption capability that is supported by the traditional cloud data security solutions like Amazon Encryption Service [1] and Google Cloud Storage [14], but also the capability to define fine-grained access control policies to protect their data. In this way, once the data is protected by an access control policy, it is automatically encrypted and only if the policy is satisfied, the data can be decrypted and accessed by either the data owner or anyone else specified in the policy. The basic motivation of the framework is to separate the management process from data encryption which is open-ended to work with any encryption method and any Key Management Server (KMS), as long as the server supports the Key Management Interoperability Protocol (KMIIP). For the management process, in addition to allow users to define access control policies for data, the framework provides automatic provisioning of the protection service at different levels of granularity: cloud tenant account level, virtual machine level, and the file/disk level inside the VMs. A combination of these levels provides a full cycle of data security, from tenant account creation, VM provision and data encryption to data decryption, VM de-provision and account termination. Therefore, the key chal-
Figure 7: Single server topology. Each of the three services resides within the same server, making it prone to intrusion as well as downtime. Nevertheless, it is a useful setup for development and testing as it can be easily executed locally on a single machine.

The challenge addressed by DPaaS is to offer the key management feature and policy-based access control as a complete service as shown in Fig. 6.

5. SERVER ARCHITECTURE

The 3D Repo system requires a bare minimum of 3 services to run:
1. MongoDB: a NoSQL database where all data consisting of client information, 3D models, access control settings, user information, etc. is stored.
2. REST API Service: a publicly available service that acts as an access point to the database. The API Service communicates with MongoDB to store/retrieve data and support other functionalities such as user authentication and modification of user/project settings. User requests always come to the DB via the REST API, or through the Web Service.
3. Web Service: a publicly accessible web server hosting templates of web pages populated dynamically by AngularJS. The Web Service offloads all requests onto the API and passes the responses onto users’ web browsers.

The following section describes those topologies that we have devised coupled with a brief explanation of what are the benefits and drawbacks of each design. Different topologies would allow for utilization of stricter security policies to secure client data and to enable higher scalability with the use of BT Research Platform auto-scale mechanism. The aim is also to benefit from the Intelligent Protection system and evaluate various policy settings to fit the exact needs of our purpose. It is important to examine different topologies that can be deployed onto the platform in order to conclude what would be a suitable deployment strategy for industrial use cases, fulfilling the need for security and scalability whilst fitting into often restrictive commercial budgets.

Single server topology.

Single server topology, shown in Fig. 7, is typically used only during development stages or as a basic test environment. Since every service resides in the same server, it is the most economical setup as the two main contributors to the running cost are the number of deployed servers, active or idle, and the bandwidth usage. Given massive 3D models, the bandwidth consumption can easily grow out of proportion even with a small number of users. This topology is also easy to deploy as it can be done with a single script on a single server in order to configure all three services.

For instance, there is no need to set up white listing of IP addresses between servers to ensure the services can communicate with each other. The latency between services will be extremely low, too, ensuring a fast turnaround for user requests. A VM hosting this solution would, however, require a high performance specification for anything larger than a basic test environment. It has to deal with concurrent requests, which may require heavy processing, whilst remaining responsive for low overhead requests such as handshakes, authentications, lookup queries, etc.

Separate database topology.

To allow for database redundancy and to improve overall protection of client data, it is advisable to keep the database instance out of the public facing domain, or even a proxy, by placing it into its own VM that would require a jump server, i.e. a machine allowed to manage boxes in separate security zones, to access it. In this setup, a white list security policy needs to be in place—the database server should only ever accept communication coming from a trusted REST API Service which has already been thoroughly tested for query injection and URL manipulation vulnerabilities. Communication channels between internal processes are SSL encrypted, and stringently restricted using a kernel-level firewall. Furthermore, all web service configuration files that contain credential information must be readable by the service account only, which is protected by encryption, keys to which are themselves stored encrypted on a enterprise-level data traveller vault. The separation of database also provides the flexibility to create DB replica configuration as well as sharding by adding a new server per each instance, also known as horizontal scaling. This makes the service more robust, reducing the chance of service disruption and loss of data. It also makes it more scalable as read only requests can be handled by the secondary databases, offloading the burden from the primary node. In the case of MongoDB, this feature is already available via its daemon mongod --replSet. A single database or a database cluster can serve multiple instances of Web/API Services, allowing better concurrency with added redundancies, minimizing the possibility of system failure.

Compute node topology.

Any requests that are deemed computationally or memory intensive need to be offloaded to a Compute Node, or a
Figure 9: Compute node topology. Computationally intensive processes are offloaded onto a compute node or a cluster of compute nodes, with a FIFO queue buffer to store awaiting requests.

Web Service
Internet
Database
REST API
Queue
Compute
Node

Web Service
Internet
Database
REST API
Router
Queue
Compute
Node

Figure 10: Router queue topology.

Web Service
Internet
Database
REST API
Router
Queue
Compute
Node

Figure 11: Maximum scalability topology.

Web Service
Internet
Database
Rest API
Queue /
Load Balancer
Compute
Node

Figure 12: Different 3D Repo topologies deployed onto BT Service Store. Same as with other open source solutions, 3D Repo can now be deployed on the BT managed infrastructure easily.

Cluster of nodes, residing in separate servers altogether. In 3D Repo, the Compute Node will be processing requests in a highly optimized and efficient C++ version of the 3D processing algorithm via a Bouncer library, allowing the process to execute as fast as possible. A queueing system also needs to be added in order to track the availability of the Compute Nodes and the list of awaiting requests. This is usually a First in, First out (FIFO) buffer, although certain amount of prioritization can be implemented depending on the specific needs of the application provider. For example, paying customers should be prioritized over free users. This setup also requires the introduction of a shared disk storage made accessible by both the queue and the nodes, as requests from the user may contain large amounts of data that is not adequate to be stored within the memory for prolonged period of time. Especially BIM files for 3D buildings tend to be rather large in size.

Router queue topology.

API Service passes on requests from authenticated users to the Router by means of a Binary JSON (BSON) protocol. The Router then validates the BSON to ensure the request is correct and to prune out possibilities of BSON injection attacks. Next, the Router determines whether the process is resource intensive and thus should be sent to the queue to be processed by a powerful Compute Node, or whether the Router can handle the request by itself and relay the results back to the API. The protocol between the API Service and the Router is, however, different to the protocol used to communicate with MongoDB. Consequently, it can restrict the functionality allowed from the API Service, removing support for any unwanted features such as dropping the whole database or removing users. For an intruder to gain a direct access to the database, they would now have to penetrate both the public facing server as well as a hidden internal server; thus increasing the difficulty of doing so and giving the operations team time to lock down the network when an intrusion is detected, for instance, with the help of BT’s intrusion detection service within the Cloud Services platform. The queue has also been moved into the Router server, meaning customer data ready for processing will no longer reside in a public facing domain.

Maximum scalability topology.

Nevertheless, to allow for maximum scalability and performance, the Web Service needs to be decoupled from the API depending on the traffic demands as shown in Fig. 11. It may be that there are many more requests coming from users directly utilizing the REST API than going through the Web Service. Thus, it may be potentially wasteful to deploy a new Web Service every time a new API VM is being initialized. Additionally, allowing an API Service to deploy on its own server allows it to fully utilize the available system resources.

6. EVALUATION AND DISCUSSION

To ensure consistent deployment across all platforms, 3D Repo defines two sets of deployment scripts. The first set, stored on GitHub define all necessary dependencies and configurations based on assumed environmental variables for IP addresses, user names and passwords that themselves are not stored anywhere. The second set of scripts, deployed to the BT Service Store, merely configure the server to be capable of downloading the appropriate scripts from GitHub, see Fig. 12. This way, the whole procedure is centrally controlled and assures that every deployment method, manually, or via an alternative deployment platform, is installed and configured in the same way. Any version updates with the application can be done easily without the need to update every script in every market place.
With various topologies designed and tested, see Fig. 13, we summarize the main findings and recommendations for a web-based 3D data hosting solution such as 3drepo.io. The Cloud Service Store provisioning common services is crucial to the framework as the fundamental requirement of a DPaaS is its ability to work with different cloud platforms. In other words, the framework is expected to provide DPaaS in a multi-cloud environment where users can have accounts from different cloud service providers. Given the popularity of cloud usage, it is not surprising if a user has different cloud accounts and depending on the data type, a specific account will be selected to manage the data. In the extreme cases, the data can be stored across multiple cloud platforms. Another advantage of using the Cloud Service Store is that the DPaaS framework supports different security solutions. A user may prefer to employ one security solution for a specific type of data, but may use another one for other type of data.

### 6.1 Server Topologies

With services tightly coupled in the single server topology, it would be difficult to scale, not to mention being prone to downtime. Given a large request requiring a lot of system resources, e.g. processing a massive 3D model import, the whole system would become unresponsive due to a single request hogging all of its available processing power. The recovery time for the system would also be long as every service within the system needs to be restarted. Furthermore, there is no benefit of setting up database in replica within the same server as it would share the same resources and provide no extra resiliency in case of downtime. From a security standpoint, such a setup is also prone to intrusion; Once an intruder gains elevated permissions on the server, they have gained access to every process within. In contrast, the separate database topology is the minimal and most economical setup considered suitable for a production-ready cloud hosting. In this design, however, the API Service is still holding the burden of dealing with processing all the requests coming into the system. As some of the big 3D models may require over 64 GB of memory to process, the service can easily become unresponsive due to a few requests once again hogging all of the available resources. What is more, the 3D Repo REST API Service is written in NodeJS. Whilst this in itself is ideal for processing concurrent web requests, it is neither as efficient nor as flexible as more conventional programming languages such as C/C++ when dealing with computationally and memory intensive 3D processing tasks including scene graph construction, model conversion or optimization for real-time rendering.

Despite the compute node topology offering the first truly scalable solution, there are still some notable security risks within such a design. For instance, large amount of sensitive client data will be stored in a shared space awaiting processing by the Compute Nodes. As this storage area is accessible by the queueing system, it would be ultimately available within the public facing server, easily accessible by anyone who compromises the API/Web server. In addition, the API Service in this design remains to hold a direct communication channel with the database. An intruder gaining control of the API Service can potentially retrieve or vandalize any data available within the database by executing nefarious queries. Fortunately, the router queue variation of this topology allows for better scalability and improved responsiveness. In the event of distributed attack or the services going down, a new instance of the API/Web Service can be instantiated and continue to serve the requests through the Router. Finally, the design of the maximum scalability topology removes the queue system from theRouter altogether. As the service grows in usage, it would be advisable to move it onto a more sophisticated cloud compute solution to improve the performance of processing algorithms, connecting to a heterogeneous cloud computing solution with a smart load balancing and job dispatching mechanism. This may be done by utilizing readily available commercial offerings such as Amazon Compute Cloud or constructing custom compute clusters.

### 6.2 Physical vs Virtual Security

Cloud compliance is a very important issue, too. In particular, the ability to trace and audit the use of infrastructure, applications and data in compliance assessment is critical. However, there is a significant challenge that relates to the difficulty of assessing risks and compliance due to the complexity of an application’s architecture and different control domains the application and associated infrastructure may span. The problem that comes in cloud computing scenario, especially in infrastructure as a service (IaaS) and platform as a service (PaaS), is that a customer choosing a cloud provider has to look at the service level agreements (SLAs) offered by cloud providers, and determine the exact level of assurance that they provide. The compliance of what is deployed on the cloud has to be assessed and assured on top of a sufficiently compliant infrastructure. For instance, data in relation to public construction in the UK must not leave the British soil.

Assuming the physical security and compliance of a data centre is handled by the service provider, there are still a few additional layers of physical security that need to be addressed. Firstly, distributing different services across multiple VMs for resiliency does not necessarily guarantee they will not reside on the same physical hardware. Hosting companies tend to cluster nodes together for performance reasons. Especially on shared systems such as Linode and Slice-host, for example, the same physical machine would always be split between multiple users each receiving a fair share of the available clock cycles. Assuming replica databases in different VMs end up sharing the same physical host, a hardware malfunction would bring down the entire system. What is more, the backups should not only reside in a different site of the same service provider in case of one
data centre becoming unavailable, they should be safely deposited with an entirely different hosting company in case of legal litigation, insolvency or other unforeseen circumstances preventing any kind of access to the main provider’s infrastructure. Backups and any persistent data should be stored encrypted wherever possible. Private keys should be stored remotely from the servers to ensure that the cloud provider has no way of accessing the information even with access to the physical hardware and data files themselves.

Jump servers.

Access for system administrators needs to be provided with minimal compromise to security. The idea of creating a jump server, i.e. a secure box that defines a single point of entry to the network, is good practice. Generally, it is necessary to disable root account, enable key-pair authentication, install automated login attempts lockout such as Fail2Ban and remove all unused services including blocking all but required application ports. Additional jump servers should be applied for subnets of the network, with different keys for each server. Unauthorised access and any suspicious activity needs to be clearly logged and administrative parties immediately notified. To this end, host-based intrusion detection systems for monitoring, logging and real-time reporting of malicious activity such as Open Source HIDS Security (OSSEC) are available.

API keys.

API keys are used widely as a general protocol for machine-to-machine communication between different web/API services. The advantage of using API keys over a username and password authentication includes:

- **Risk of interception**: User credentials are generally reused over a number of different sites, which increases the chance of it being intercepted and exposed.
- **Greater entropy**: A key will be much more random and longer in length; making it much harder to compromise.
- **Separation**: A key is only valid for a single user. If the API key is compromised, it can be simply deleted from the system and a new API key can be easily generated by the user or an administrator. User passwords will not be exposed.
- **Convenience**: Instead of prompting the user to authenticate themselves using a username and password or embedding their credentials in their code, the API key serves as a much more convenient method of communication from a machine to another machine. Whilst API keys can be a convenient alternative to user credentials, they can also effectively lower system’s security defences if not used with careful security considerations in mind. They should never replace a session key; an API key should be used in a similar manner as user credentials—to authorise a user of the system for a limited period of time. This is to avoid the key being exposed in network traffic. When an API key is compromised, the attacker gains free access to the system until notified. API keys should also have a limited life-time to limit the effectiveness of unused or compromised keys.

**Share a weblink.**

Sharing a weblink with unregistered users is a common practice adopted by many cloud storage provider including Google Drive, DropBox and even Business Collaborator (BC). This gives the ability to access data simply via possession of a URL. Whilst it is a very convenient way of sharing data, it induces a massive security risk. Application designers need to consider whether public access to data is really necessary; whether the user understands the underlying risks; and also how this publicly exposed data should be stored and accessed to ensure there are no ways of exposing private data elsewhere in the system. BC, for instance, lists the contact details of all web-link recipients with session-based protection to prevent unauthorized misuse such as email forwarding. Access to data should be revoked at the earliest possible instance if there is any suspicion on lost or compromised credentials. It is also important to ensure users do not have access to information for longer than necessary, be it credentials, API key or a weblink.

**6.3 Business Considerations**

A flexible, scalable and well secured system needs more servers to deploy with, thus coming at a higher cost. There is also the added complication of setting up each server to ensure firewall settings are correct and each service is able to communicate with each other. To decide on which topology is the most suitable for a situation, the client security requirements have to be considered. There are two typical deployment scenarios for 3D Repo: private and public.

**Private cloud.**

3D Repo solution for a specific company or a consortium would be deployed in a private cloud. This may be hosted in the company’s infrastructure or within a cloud provider. This would come with a strict white list policy, ensuring only users logging in from a recognized and authenticated IP address are allowed in. A dynamic white listing policy could be applied should the company wish for external access, i.e. working from home, accessing via 3G/4G network on construction sites, etc. with authenticated devices. As the usage is determined by the client company, it can be estimated up-front. Consequently, the chance of intrusion is lower with the lower user base and also the ability to deploy a white list policy. Hence, topologies in Fig. 8 and Fig. 9 may be sufficient.

**Public cloud.**

The second scenario is a multi-tenant cloud solution accessible by the general public, akin to GitHub. There users can sign up and use the service. Thus, it requires a capability to grow and maintain responsive as the traffic increases, so scalability without downtime is essential. As it is designed to serve anyone, white list policy is not applicable in this setup. However, as with any system, we can at least rely on black list policies for any detected malicious attempts such as three failed consecutive login attempts and similar. All customer data will have to share the same instance of DB, so a clean separation and protection of data is critical. Given the use case, only topology in Fig. 11 will be suitable.

**7. CONCLUSIONS**

As web technology advances, more 3D data than ever before is being hosted online. It is important to consider all vulnerabilities and implications of utilising such technology. There is no one solution fits all in constructing a web-based data hosting platform, as the needs and risks of each appli-
cation can be very different. Nevertheless, it is important to perform a thorough evaluation of what are the possible dangers and identify the severity of their consequences. Implementation of the system should take focus on preventing all high risk scenarios from happening, with consideration of the overall budget for the design.

This paper, therefore, presented an important use case for cyber-security within Saas applications in general and 3drepo.io in particular. Introduced were new role-based application level security measures that simplify overall management of projects. In addition, the latest research developments in cloud services in the form of BT Service Store and Intelligent Protection were outlined. Service Store has been designed to enable large-scale application deployment across various hosting providers in a unified way. On top of this, the Intelligent Protection added further security enhancements including malware and intrusion detection, packet sniffing, security patch management, logging and monitoring, etc. Using these systems, we designed five increasingly complex server layout topologies which were discussed with regards to their inherent security, scalability as well as commercial viability. Finally, a number of valuable recommendations were summarized.

In conclusion, advanced topologies with effective access control can distance critical data from front-line services, hence increasing the security. Nonetheless, it also complicates installations and configuration set-ups which can lead to undesirable vulnerabilities. A cloud market place with good administration tools can impose well defined security rules for all implementations, reducing the risk loop holes and malfunction. It also relieves the application developers from the burden of managing an enterprise level solution.

With this in mind, the human factor must also never be forgotten. Insider access is still one of the easiest and most prevalent ways of breaching an otherwise secure system. Audit trails and rigorous reviews of the systems in place are needed to minimise such risk and ensure the defences are effective.

Future work.

One crucial aspect which we did not address are the ways of mitigating various types of attacks. For business-to-business solutions, such attacks are preventable with white-listing of IP addresses, hence restricting access at the network level. However, for a public facing service, further precautions would have to be taken in order to guarantee uninterrupted access to the platform as those are currently outside the scope of this paper and are, therefore, left for future work.

Acknowledgements

This work was part-funded by the pan-European EIT Digital Trust High Impact Initiative (HII) and the Innovate UK Digitizing the Construction Sector grant No. 102,054 for which we are grateful. Commercial aspects of this work were evaluated by the Technical Research Centre of Finland (VTT). We would especially like to thank Maria Prokopi and Lee Tarrant of Digital Catapult, Markku Kuttoven of F-Secure and Dennis Moynihan of EIT Digital for their help, feedback and positive encouragements throughout the project.

8. REFERENCES

[1] Amazon. Amazon web services (aws) - cloud computing services, April 2016. [Online].


