Extrusion Detection of Illegal Files in Cloud-Based Systems

Rob Hegarty and John Haggerty*

School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University,
Chester Street, Manchester, M1 5GD
R.Hegarty@mmu.ac.uk

*School of Science and Technology, Nottingham Trent University, Clifton Campus, Nottingham, NG11 8NS
John.Haggerty@ntu.ac.uk

Abstract: Cloud-based architectures have become the predominant paradigm for organisational infrastructure development due to the flexibility and scalability that these systems provide. However, issues around privacy and trust in such environments remain as has been demonstrated in recent attacks. There are two security challenges for cloud providers to resolve. First, they must ensure that only authorised downloads of potentially sensitive data can be made and they should have a means by which to detect any malicious activities. Second, any files that are uploaded to cloud providers must adhere to geographical legalities. Current security mechanisms employed in the cloud, such as firewalls and Intrusion Detection Systems, find these issues problematic. This paper therefore presents a novel approach, *XDet*, for the extrusion detection of illegal files being maliciously uploaded to or downloaded from the cloud, which can be used in conjunction with other security countermeasures to ensure robust and secure cloud systems. This is achieved through the creation and detection of signatures from files of interest within the cloud network environment. The feasibility and performance study in this paper, whereby *XDet* has been applied to network traffic to detect files of interest, demonstrates the applicability of this approach.

Keywords: Cloud security, privacy and trust, extrusion detection

1 Introduction

Cloud-based architectures have become the predominant computing paradigm in recent years for organisational IT infrastructure development. These systems provide a number of advantages over traditional environments, such as flexibility in meeting customer demands, automated control over heterogeneous systems and networks, economies of scale, and streamlined processes. One key advantage to users is accessibility and scale of data storage; users can access their data from anywhere rather than requiring access to local networked resources. This has ensured that it provides an attractive model to home and business users alike.

Despite the popularity of this model, with support from organisations such as Google, Amazon and Fujitsu, issues around privacy and trust in cloud computing remain. For example, users of cloud-based systems are required to hand over their data, including proprietary or personal information, to external organisations to manage. This leads to two security challenges. First, the cloud providers must ensure the privacy of the information that they manage on their user's behalf. Second, due to the potential value of that data, cloud computing providers are targets of attack. Global access means that attacks can happen from anywhere, with access control mechanisms often relying on usernames and passwords. Cloud computing users must be able to trust their provider to provide the levels of security that they require by both protecting their data as well as detecting privacy violations as they occur. As Huang and Yang (2010) suggest, one advantage of cloud computing for securing data and systems is that providers are able to sample and share security information dynamically.

Security is a major concern irrespective of the underlying cloud technology with providers required to detect and respond to attacks on their resources. Recent work has focused on providing countermeasures to a wide range of attacks such as Denial of Service, attacks against the hypervisor or virtual machines, and privilege escalation. Many of these approaches apply perimeter-based defences, such as firewalls and Intrusion Detection Systems (IDS), to cloud architectures. However, these countermeasures are limited in their ability to protect cloud users from privacy violations such as Intellectual Property theft or cloud providers being held accountable for illegal file storage as they focus on packet information rather than message content. This paper therefore proposes a novel approach, *XDet*, for the detection of such security incidents.

Extrusion detection is a field of intrusion detection that analyses outbound connections and data leaving an organisational boundary to identify potential breaches of security. In particular, it examines data leaving the organisation to identify malicious users within the organisation's perimeter, malicious software residing on internal systems, or potential threats or attacks to neighbouring networks. However, the focus of this paper is to extend this approach to detect files of interest being maliciously uploaded or downloaded from cloud-based systems. Illegal files of concern include indecent images of children shared between paedophile rings, proprietary documents relating to intellectual property being accessed by unauthorised users, software piracy, and data shared by terrorist cells. The novel approach posited in this paper can be deployed as a stand-alone countermeasure for extrusion detection, or can be incorporated into other cloud security approaches such as that proposed by Yu *et al* (2013).

This paper is organised as follows. Section 2 discusses related work in cloud security and file analysis. Section 3 posits the *XDet* approach and in particular, discusses the underlying architecture of the system. Section 4 presents a feasibility and performance study to demonstrate the applicability of the approach for extrusion detection in cloud-based systems. Section 5 provides a discussion of the *XDet* approach. Finally, we make our conclusions and discuss future work.

2 Related work

Firewalls and IDS are commonly deployed in traditional, perimeter-based network environments to protect systems from attack. Used in conjunction with one another, firewalls enforce the network security policies of an organisation whilst IDS detect any malicious use by network insiders. Both countermeasures protect these systems from unauthorised access to private data by blocking traffic based on packet signatures or by identifying anomalous user behaviour. Whilst these systems are suited to perimeter-based networks, recent work has attempted to employ these security techniques in cloud-based architectures due to their ability to share information dynamically and in real-time across large-scale systems (Huang and Yang, 2010).

Firewalls protect networks or systems from unauthorised access by allowing or blocking traffic to and from a private network. This is problematic in cloud-based computing as the management of the security policies enforced by the firewall and security of data may be controlled by either the cloud provider or the organisation. As Kurdi et al (2013) note, traditional firewall approaches may be used in cloud environments of up to 150 nodes, they are not suitable for larger architectures. The issue that either party may have conflicting interests remains. For example, the cloud provider may not want their users to have access to their firewall security policies and audit information, and vice versa. Therefore, Khakpour and Liu (2012) propose a firewall outsourcing approach where businesses outsource their firewall services to ISPs without revealing their firewall policies to them in order to protect the integrity of their security policies. This overcomes the issue noted in (Liyanage and Fernando, 2013) where the client can configure inbound or outbound traffic based on protocols and ports and the cloud providers also have permission to change these settings. To overcome this issue, they propose an approach using encrypted channels between users and cloud providers to prevent unauthorised policy updates and eavesdropping in firewall systems. Alternatively, Perera et al (2012) propose an approach where computations are moved to the public domains while keeping the data within the private network. Yu *et al* (2013) propose a modular cloud-based firewall where each module is responsible for detecting a particular attack.

Intrusion detection aims to identify and respond to malicious activities within a network or system. Intrusion detection in cloud-based networks remains problematic due to the architecture and scale of such networks leading to issues such as false alarm reduction (Meng et al, 2013) and placement of IDS components. However, a number of approaches have recently been proposed due to the importance of this activity in protecting cloud users. Oktay et al (2013) posit an approach based on IDS in traditional networks. Their proxy Network IDS architecture is a gateway-based approach whereby the detection and response to attacks is conducted by an entity outside the cloud environment. Alqhatani et al (2014) posit a framework for the Service Intrusion Detection System in Cloud Computing (SIDSCC), and whilst in its early stages of development, recognises the requirements of IDS in cloud-based systems over traditional networks. Ficco et al (2013) posit an approach that adapts to cloud-based architectures. Their system utilises probes at different cloud architectural levels; hypervision, infrastructure, platform, and application, to detect attacks and use security agents to pass data to security management engines. Bharadwaja et al (2011) posit Collabra, which identifies misuse of the hyper-call interface to compromise guest operating systems. Collabra detects attacks by scanning each call and incorporating integrity checking and collaborative detection. Nikolai and Wang (2014) propose an approach that utilises hypervisor performance metrics for the purpose of detecting attacks by identifying abnormal activity.

The approaches above have a number of disadvantages in the provision of privacy and trust in cloud architectures. Cloud-based firewalls and IDS are designed to identify attacks against the system rather than for identification of file or user data as it enters or leaves the cloud. These schemes use network traffic characteristics located at the Transport Layer and below of the TCP/IP stack rather than Application Layer content to detect attacks, and therefore, are limited to detection of network attacks such as Denial of Service (DoS), port scanning, unauthorised access, viruses, and network worms rather than the legitimacy of file transfers. Moreover, the management and placement of security policies enforced by such systems is also challenging. For example, if the countermeasures are managed by the cloud providers, users will have a limited input into those policies enforced; if the users manage policy enforcement, cloud providers may override those policies. Therefore, the effective and efficient placement of these countermeasures is challenging.

In order to identify files being maliciously uploaded or downloaded to the cloud, content analysis will have to be conducted. A number of approaches have been proposed for data identification from file features where file system information may not be present, as is the case with the cloud environment. For example, Crossley et al (2013) discuss the potential of metadata tags as a potential approach to tracking original file information in the cloud. Chawathe (2012) proposes the use of block-wise rather than file hashes for file fingerprinting whilst Sportiello and Zanero (2011) posit a scheme to perform file block classification by using Support Vector Machines for the identification of the relative file blocks. da Cruz Nassif and Hruschka (2013) propose clustering as an approach to identify similarities between files and retrieve files of interest. Pierris and Vidali (2012) propose an approach based on block hash values combined with Hierarchical Self-Organizing Map algorithms in order to classify broken chains of previously unknown files. Schaefer et al (2012) and Edmundson and Schaefer (2013) propose an approach for the retrieval of multimedia files from Huffman tables located in the header of JPEG files. Shabtai et al (2011) present eDare and Liu and Chen (2012) posit MalPEFinder to detect malware by searching for executable files. These schemes have in common three issues that restrict their use in the identification of files in cloud-based systems. First, many of the schemes require complete file or block data to be present and identifiable, which may not be the case in network traffic or individual packets. Second, if a single bit is changed in the original file, the detection scheme may be circumvented. A single bit may be altered by a malicious user or due to the network environment. Third, encrypted channels, such as Secure Sockets Layer (SSL) connections, may be used between the user and the server which may defeat packet analysis.

This paper therefore proposes *XDet*, a novel scheme for the identification of illegal file transfers to and from cloud platforms to overcome these issues. Whilst the approach has its roots in firewalls and IDS approaches to cloud security, it focuses on file signatures, i.e. Application layer data, rather than packet signatures, which are used for the detection of network attacks. As will be seen in the next section, this approach does not assume entire block or file information to be present in a networked environment, so it posits a novel scheme as the basis for detection. In addition, as unique file signatures are generated on cloud content creation, knowledge of previous attacks on files need not be known as is the case with traditional intrusion detection signature schemes. Moreover, the placement of *XDet* between storage and network functions ensures that data may be inspected pre and post network encryption operations.

3 *XDet* approach

Recently, the ability of cloud computing architectures to provide the privacy and trust that users require for data storage has been called into question. Cloud storage may allow malicious users to store illegal files, leading to issues for cloud providers such as loss of reputation, legal responsibility for storing illegal files, or to not be able to prevent data theft. For example, Microsoft recently reported one of their users who had stored an illegal image of a young girl on their OneDrive cloud storage to the authorities (Kelion, 2014). Alternatively, Apple is to review its iCloud storage facility following the theft of intimate pictures of celebrities and their subsequent publication online (BBC, 2014). This section posits the *XDet* approach that may be used as a countermeasure to such privacy and trust issues.

The design goals provide the system requirements for the *XDet* approach. The principal goals are as follows:

- *High-speed, large volume analysis.* A large amount of data must be analysed within tight temporal constraints as data is uploaded or downloaded between the cloud provider and the user. This analysis will impact on network and memory resources within the analysis application, and the effect must be kept to a minimum.
- *Analysis irrespective of the underlying technology.* The cloud environment allows connections from many different devices on a multitude of platforms, from those commonly used such as Microsoft Windows, to the more parochial such as mobile phones. Moreover, connections between users and their cloud providers may or may not be over encrypted channels.
- *Keeping false positives to a minimum.* A problem with any type of signature analysis is that of false positives, i.e. falsely reporting the presence of data of interest when it is not present. Given that cloud providers wish to maintain their users' data privacy or the results of signature matching may identify illegal files that may lead to judicial action, the results produced by this approach must be robust. Therefore, the system aims to reduce false positives by providing robust file signatures.

These requirements form the focus of the approach posited in this paper and are addressed in the remainder of this section.

Figure 1 provides an overview of the *XDet* approach.The dashed line represents the separation of the signature search application from its external components. Files are selected by the user or the cloud provider that are deemed private and it is recognised that not all files meet that criteria. In the case of the user, these files may include sensitive photographs, proprietary information or design blueprints. In the case of the cloud provider, these files may be indecent images of children or stolen sensitive files that should not be uploaded to their storage servers. Guidance on what constitutes files to be protected by the system should be provided in corporate security policies or by the private user. Therefore, as described below, unlike traditional IDS and firewall approaches that use Transport layer and below packet characteristics as the basis for attack detection, a signature is created from the file, i.e. Application layer data, and stored for searching message content. As the file is sent over the network, packet reassembly is carried out and the file is searched for the signature. If a match is found, a report is generated or the upload/download is blocked until remedial action is taken.

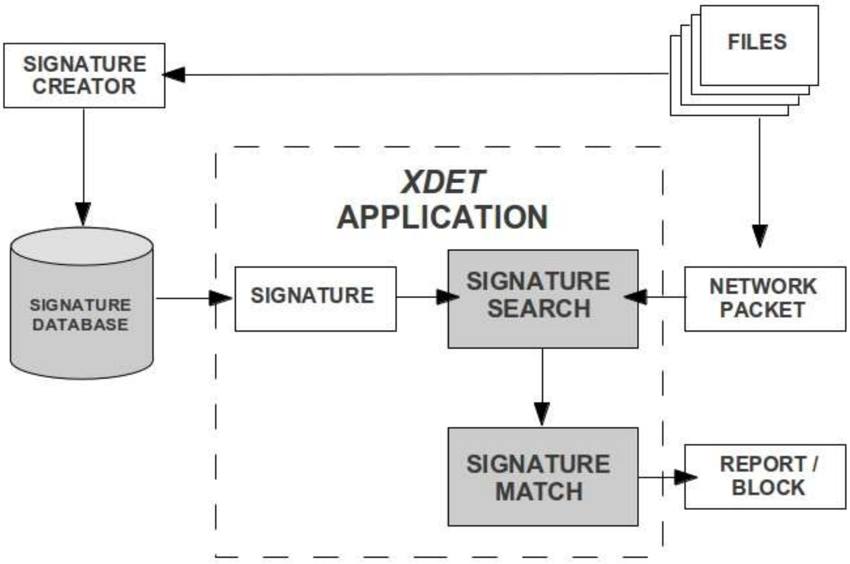

Figure 1. Overview of the *XDet* approach.

As described above, if a match is found, the system reports the download/upload attempt and depending on user preferences, blocks further transfer of the file to the user's device or the cloud system. In report only mode, the user's identity, such as their user name, and the IP address of the access is recorded. This may then form the basis for audit or digital investigations. As discussed above, the issue of user names and passwords as the main access control mechanism within many cloud systems is problematic due to the global access from a multitude of devices and this has been demonstrated in the Apple iCloud attack. However, the reports may be used to track the perpetrators of such attacks, users uploading indecent images of children, or if a large number of account accesses come from a single or limited range of IP addresses, it may be used to indicate that such an attack is taking place. This is especially the case if the cloud provider makes use of the scale and collaborative working that such systems enable. In report and block mode, the system reports the file download attempt as above but blocks further transfer of the file until the validity of the user can be ascertained. A number of authentication and authorisation schemes to achieve this exist. For example, a cookie may set when a file is uploaded.

However, with the multitude of devices available, the legitimate download of a file may be from any number of computers where the cookie resides. Alternatively, passwords can be set to authorise the download of a blocked file providing two factor authentication. Finally, hashes of the original file may form the basis of the authentication scheme to be compared with hashes owned by the user.

Figure 2 shows the placement of the *XDet* device within a cloud architecture. As illustrated, *XDet* is placed between a cloud server and data storage rather than on the perimeter, as is the case for many cloud firewall approaches. However, it is envisaged that it may be placed in modular firewalls, such as that proposed by Yu *et al* (2013), in smaller cloud architectures such as those used by home users for data storage. Data is passed from a user's Virtual Machine (VM) via a hypervisor to the cloud server to which they are connected. This data is then passed through *XDet* to distributed storage. It should be noted that whilst figure 2 represents *XDet* as a separate device, it can be an application residing on the cloud server itself. In larger cloud architectures where there are many cloud servers passing data to distributed storage, multiple *XDet* devices may be required to meet the need for scalability. In architectures where home users store data to their cloud provider, such as that used by iCloud, SkyDrive, or Google, VMs and hypervisors are replaced by software that runs on the user's device. However, storage is typically distributed.

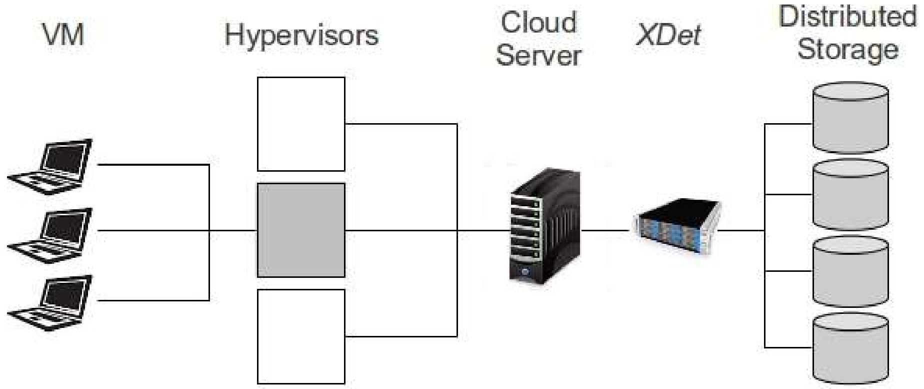

Figure 2. Cloud architecture illustrating the placement of *XDet*.

The advantages of this architecture are threefold. First, *XDet* is located within the cloud provider's perimeter and can therefore be protected by security countermeasures already in place, such as firewalls and IDS that detect attacks against the system based on packet characteristics. Second, *XDet* is able to take advantage of the scale and collaborative working that cloud-based systems provide which may add to the security effort. For example, signatures may be shared between *XDet* devices to protect a larger number of users, particularly if they use a number of cloud servers. This is particularly the case for the protection of service providers by preventing the upload of indecent images of children to distributed storage where police authorities can make available signatures formed from their signature databases. Third, users are able to make use of encrypted connections, such as those utilising SSL, when transferring data to and from the cloud. Data is decrypted by the server prior to writing to storage as users upload their files, and is encrypted by the server when they download them. Hence *XDet* is placed between the server and distributed storage before or after the data encryption/decryption process to enable file data to be read for signature matching.

XDet analyses HTTP/TCP data streams to detect the transmission of known files across a computer network using our block based analysis technique. The average size of images downloaded from the top 1000 websites is approximately 1MB (HTTP Archive, 2014); a 1MB file represents 2046 blocks for use in the signature detection process, as the first and last blocks are not suitable for reasons explained later in this paper. In addition to this JPEG & GIF files make up 74% of the image requests made across the top 1000 websites as of August 2014 (HTTP Archive, 2014). *XDet* targets these image formats to ensure the majority of images transferred across the Internet can be analysed.

XDet analyses data traversing the network by monitoring for HTTP traffic, reassembling image files from the Application layer data in HTTP/TCP streams, and carrying out block level analysis of the reconstructed files. Driftnet (Driftnet, 2014) is used to monitor the network traffic and carry out reassembly of image files. By monitoring traffic in promiscuous/monitor mode, it is possible to analyse data traversing the network regardless of whether it is addressed to the monitoring device or not. This feature may be useful when monitoring traffic in an environment where data addressed to multiple virtual machines is present.

The reassembly of files is required to enable the application of our block level analysis technique to data that has become segmented through transmission over HTTP/TCP. Files are constructed of fixed size (typically 512 byte) blocks referenced by the file system. This provides a standard interface for operating systems and applications to interact and to create modify, and delete files. There is not a direct mapping of the blocks that represent files to the packets used to transfer them. Packet sizes vary, and the addition of header information from the TCP/IP layer and HTTP protocols mean the payload of packets that contain data from the blocks representing a file are not aligned to the same (512 byte) boundaries found in file systems. To further complicate matters, HTTP compression techniques such as *gzip* alter the structure and representation of data in transit. These challenges mean that raw packet analysis would require complex and computationally expensive processing. We avoid this through reassembly of files and the application of our block-based analysis technique.

Our block level signature detection technique performs a comparison between bytes selected from a selection of off-sets from within a block to determine if a match can be found with a signature set containing the signatures of a private file. This signature scheme has been known in file system forensics for some time as a means by which efficient and effective file identification in storage media may be achieved (see for example, Haggerty and Taylor, 2007; Mohamad and Deris, 2009). Of the blocks that make up a file, all but the first and final blocks are suitable for signature detection. The first block is unsuitable as the metadata and Huffman tables stored in the first block are low entropy and/or likely to result in signatures that are prone to collision. The final block in a file is unsuitable as it contains slack space. Slack space occurs when the data from within a file is not exactly divisible by the block size of the file system. The final block occupied by data from the file can also contain data beyond the EOF (end of file). This data may be the remnants of files previously stored in the file system. The use of signatures generated from this block may result in false negatives as files transmitted via HTTP do not contain slack space. False positives may also occur if a signature is generated from data in the final block that belongs to a file that previously resided in the file system.

In order to carry out analysis two processes must be carried out. First a signature set must be generated, and second, each of the files undergoing analysis must be compared with the signature set. In order to generate a set of signatures from a collection of files identified as being private, a signature from a specified block within the file is selected. We call this offset the block offset; the offset must be between 2 and L_{-1} (where L is the length of the file in blocks) for reasons previously stated. The signature is generated by selecting a predefined number of bytes from specified offsets within the block, which we call *byte offsets*. The same block and byte offsets are used to generate signatures for each private file signature added to the set. These offsets are noted in a configuration file, so that they can be used in the signature detection process when file signatures are created for comparison with the signature set.

Signature detection is carried out to determine the presence of files deemed to be private by either the cloud user or provider. Each file undergoing analysis has a signature generated from the block and byte offsets specified in the configuration file, associated with the signature set being used in the analysis process. Once a signature has been generated from the file it is compared with the signature set to determine if a match can be found. As illustrated in a simplified representation of file signature comparison in figure 3, File A and File B provide a match, whereas File C does not.

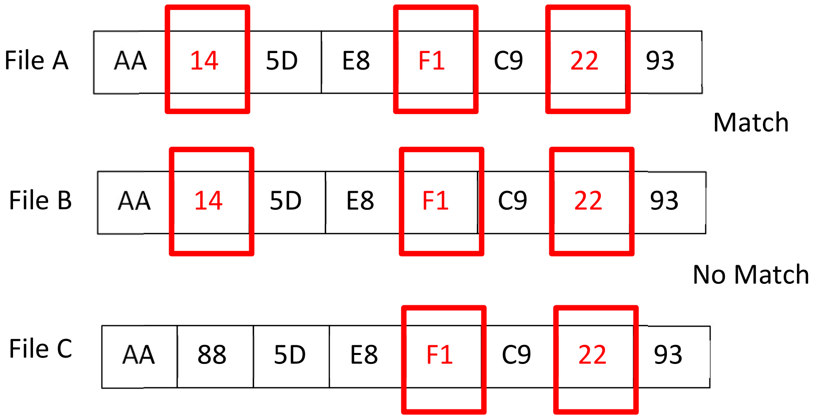

Figure 3. Simplified representation of file signature comparison.

The comparison illustrated in figure 3 is simplified as only 3 bytes have been selected to form the signature. The actual number of points can be chosen depending on the desired balance between the potential for false positives and false negatives; as the number of points of reference increases, so the likelihood of false positive decreases. However, a large number of signature bytes may work against the system, especially in networked environments for two reasons. First, packets may be fragmented as they traverse the network, which would defeat the scheme if signature data were spread across packets. Second, larger signatures add some computational overhead, leading to a loss of throughput. Therefore a balance of signature size and accuracy must be achieved. For example, a signature formed from 7 bytes provides a false probability of 1.39×10^{-17}, and a signature of 8 bytes 5.42×10^{-20}. Approximately 10 bytes or more provides a robust signature whilst processing minimal data and in carrying out the experiments in the next section each signature comprised 12 bytes. Hash-based techniques often utilise MD5 or SHA1/2, which requires the comparison of 32+ bytes. The approach posited in this paper therefore provides obvious performance improvements during the analysis process and while transferring signature sets across the network. Signature sets can hold a significant number of signatures: the current Police signature library contains in excess of 200 million signatures.

4 Feasibility and Performance study

In order to evaluate the feasibility of monitoring traffic in a live network a feasibility study was conducted with *XDet* monitoring the NIC (Network Interface Card) of a computer transferring data. Two subsequent experiments were carried out to evaluate the performance of the *XDet* technique when varying numbers of files and signatures were used in the analysis process. These experiments targeted a number file sets created from a standard JPEG data set from Caltech (www.caltech.edu).

The live *XDet* analysis feasibility study monitored the NIC on a host computer and reassembled image files that were transferred over HTTP as the result of a web browsing session. Each time an image was reassembled a signature was generated from the file. The file signature was then compared with the signature set and the user informed if a match could be found. This study was carried out by downloading a selection of images from target websites prior to a browsing session. Signatures were generated from these images, and the resultant signature set used during the subsequent browsing session, which visited the target websites. This feasibility study resulted in all of the signatures in the signature set being detected when the target websites were visited during a browsing session. In a separate browsing session lasting five minutes (on a 50 Mbps connection) where the target sites were not visited no matches were detected.

To evaluate the performance of *XDet* when varying number of files are analysed we created a signature set containing 5000 signatures from files in the Caltech data set. These were used in place of the reassembled images captured from the NIC. As the focus of the experiment was on the performance of the XDet technique, rather than image reassembly time. A standard data set containing 30,608 JPEG files from Caltech was downloaded for use in the performance evaluation. File sets containing varying numbers of files from the data set where searched for matching signatures. The file sets contained between 1000 and 10,000 files with 1000 file increments.

The parameters for the signatures used in the performance experiments were as follows. Block two was selected for signature generation as this enabled a file as small as 1536 bytes to be detected. 12 byte off sets from within the block were selected to form the signatures. As discussed above, this provides enough entropy in the resultant signatures to produce results with zero false positives. Further work is required to determine how to calculate the optimum number of bytes to use in a signature, and this is discussed further in the next section.

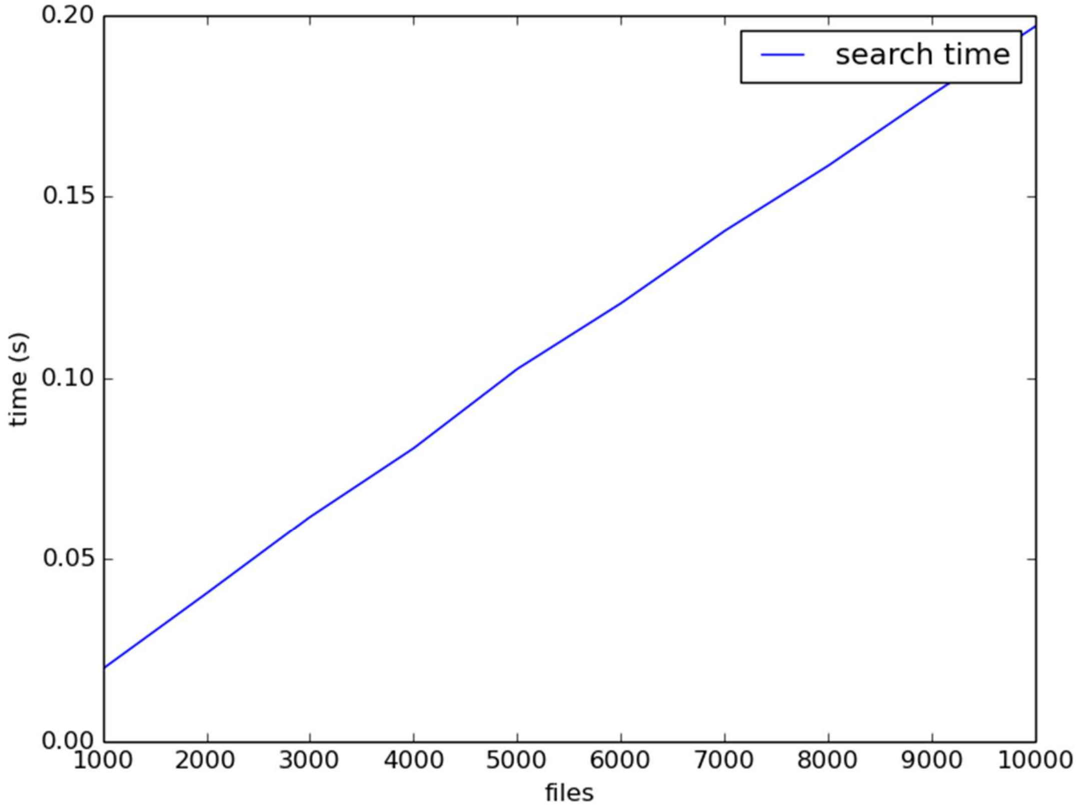

Figure 4. XDet search time for 5000 signatures with varying numbers of files

The time required to search each of the file sets is illustrated in figure 4. Each search was carried out using 5000 signatures. All files with matching signatures were identified during this search with zero false positives or false negatives recorded during the experiment. The search recorded times are minimal in comparison to the bandwidth available in modern network environments. The relationship between search time and packet capture was approximately linear. This which was expected due to the use of a Python implementation using sets for signature comparison. More optimum data structures for search are available, however the focus of this paper was on the signature technique, rather than the data structure used to store and search the signatures.

It should be noted that the search times recorded do not include the time required to reassemble the images from the packet capture. This time was consistently less than 3 seconds during the feasibility study, and entirely dependent on the implementation of Driftnet. The focus of our study is to record the time required for the actual search as more optimised image reassembly techniques would almost certainly be used in a live environment. Alternatively the data undergoing analysis will have been reassembled prior to storage. This is the case when web browsing takes place in a home environment, where the internet cache would be the target. This is also the case in cloud environments where the data is reassembled prior to being sent to the storage network. The most significant aspect of the performance was that zero false positives or negatives were recorded throughout the experiment which analysed a total of 10,000 images.).

A second experiment was conducted in order to evaluate the performance of *XDet* as the number of signatures in the signature library is increased.

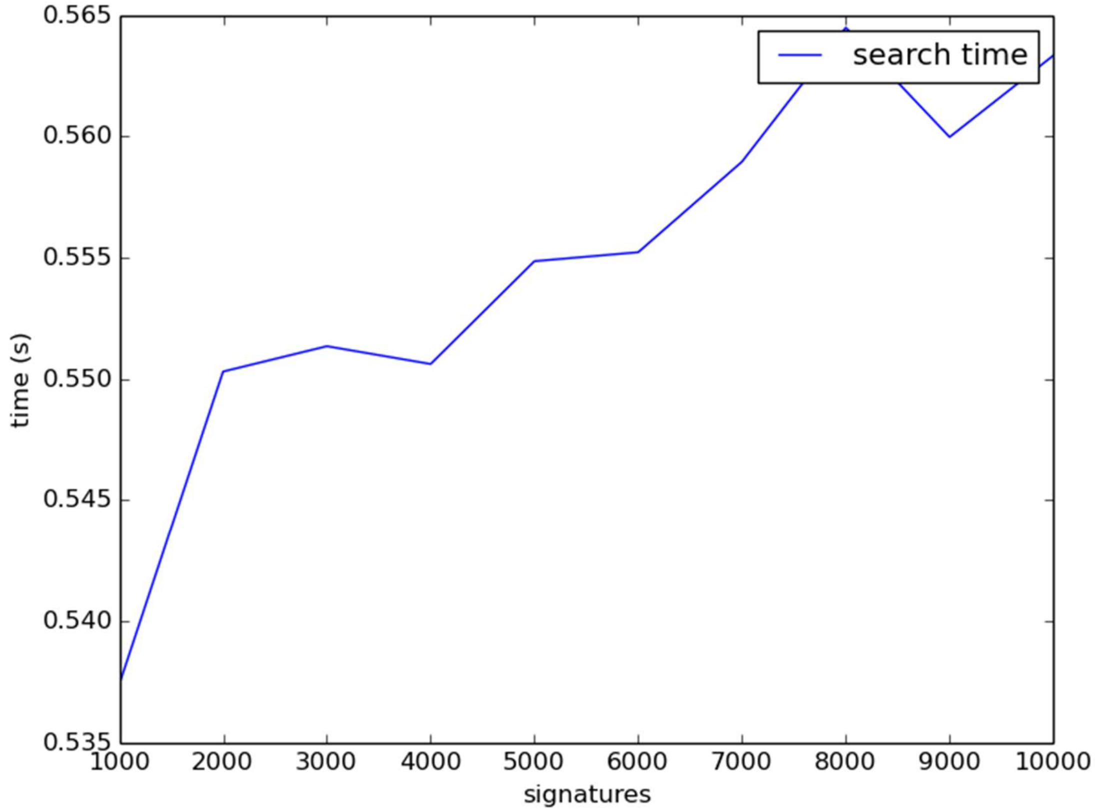

Figure 5. XDet search time for 30,608 files with varying size signature sets

During the second experiment all 30,608 JPEG images from the Caltech data set where searched using signature libraries derived from the files used in the previous experiment (file sets containing between 1,000 and 10,000 JPEG files at 1,000 file increments).The relationship between the time required carrying out the search and the number of signatures in the signature libraries was once again linear, and minimal in comparison to the bandwidth available in contemporary networks. The results are illustrated in figure 5, once again all matching files were detected with zero false positives or negatives recorded despite a total of 30,608 files being compared to 10,000 signatures..

5 Discussion

The goal of *XDet* is to detect the upload or download of illegal files to cloud systems to ensure privacy and trust of data storage in such environments. In order to achieve this, the approach focuses of application data in transit rather than the packet characteristics or identification of anomalous behaviour used by firewalls and IDS. However, it is envisaged that *XDet* can be used in conjunction with such countermeasures to provide a holistic approach to the security of user data in cloud systems. For example, it may form a discrete module in the approach proposed by Yu et al (2013). Moreover, it focuses on the detection of attacks on data privacy to enhance IDSs that protect cloud systems from network-based attacks, such as DoS, or compromises of guest operating systems through the hyper-call interface as identified by Bharadwaja et al (2012).

In order to achieve this, *XDet* takes a file-centric approach building on related approaches in security and computer forensics. Users and cloud providers are able to create a signature from the files that they wish to protect. As discussed previously, signature schemes associated with file system characteristics, such as file hashes, block-wise signatures, or

JPEG header metadata, do not provide identification features that may be used within the network environment. Therefore, *XDet* utilises a scheme that uses a single block of data from an associated file, and creates the signature by using points of reference within the block of data. The approach is then able to analyse HTTP/TCP data streams for these points of reference within the data being transmitted by reconstructing image files using Driftnet (2014) to reassemble image files and search for the relevant block offsets. The advantage of this approach is that image files are re-assembled for searching and avoids issues around compression of data in network packets. However, this architecture does incur some overhead and future work aims to mitigate this issue.

The *XDet* software is located between the cloud server and distributed file storage rather than on the perimeter of the cloud network. This architecture has three main advantages within cloud-bases systems. First, it can be protected by perimeter-based security devices, such as firewalls and IDS, employed by the cloud provider. Second, it can take advantage of the scale and collaborative working of cloud-based systems. In this way, the approach is able to share security information, such as signature data sets, collaboratively, a key advantage of security countermeasures in these environments noted by Huang and Yang (2010). Finally, the system enables the cloud providers to employ network-based encryption schemes, such as SSL, to protect data in transit as data is analysed between the cloud server and storage of the file data, which is no longer encrypted.

The feasibility and performance study demonstrates the applicability of the approach. To replicate the writing of data to storage in the cloud environment, image files transferred over HTTP were reconstructed and signature sets were created. These were then applied to network traffic to identify files of interest and search times were minimal, especially in comparison to bandwidth availability if cloud-based systems. During the performance study, zero false positives and false negatives were identified. However, given the amount of data that may exist in cloud environments, it is envisaged that false positives and false negatives may occur. Future work will aim to mitigate this issue by identifying optimum data structures for the storage and search of signatures. In addition, candidate block selection for the generation of signatures may be refined through the analysis of the entropy in each of the blocks that a file comprises; high entropy blocks result in more accurate signatures than low entropy blocks. Further research is required to fully understand the relationship between entropy and accuracy.

In order to evaluate the scalability of *XDet* and its ability to monitor high bandwidth networks, a distributed version of the techniques will be produced in further work. As each *XDet* device is capable of working in isolation, without the requirement to communicate with other devices, the development and deployment of a distributed *XDet* system is easily achievable. However, it is desirable to enable collaborative working to provide a more holistic approach and to take advantage of the benefits of cloud architectures. In addition, the impact of file encryption on the approach will need to be assessed. As discussed above, network-based encryption schemes do not have an impact on the approach. However, end-to-end encryption, whereby the files themselves are encrypted at both endpoints may prove problematic and it is envisaged that schemes such as Ming Li et al (2013) whereby encrypted data may be searched in cloud-based systems may be adopted by the *XDet* system.

6 Conclusions

Cloud computing has become an important paradigm in organisational network infrastructures due to its flexibility and scalability. Moreover, cloud computing architectures provide an attractive solution for data storage due to the accessibility of data from multiple computing devices and the flexibility in memory availability. However, cloud computing is not without its security issues, as recent attacks have demonstrated. Primarily, challenges remain around the way in which files are uploaded to or downloaded from the cloud or what files are actually stored by providers, which directly affect privacy and trust in such environments.

The *XDet* approach has been developed to identify data leakage from cloud networks. In particular, it aims to complement existing approaches, such as firewalls and IDS, which detect network attacks against cloud-based systems. To achieve this aim, it generates signatures from private files and stores them for comparison with signatures derived from files being transferred across a network. In this way, unauthorised uploads or downloads of potentially confidential data may be detected and prevented. Through a number of experiments on real world data this paper has demonstrated the feasibility of applying *XDet* to live networks. Furthermore, it has shown that *XDet* performs well in terms of the temporal requirements for network-based data searches and has a high degree of accuracy. Therefore, *XDet* demonstrates the potential to detect the illicit extraction of information from cloud networks. Future work aims to refine and test the approach when applied to different cloud environments.

References

Alqahtani, S.M., Al Balushi, M. and John, R. (2014), "An Intelligent Intrusion Detection System for Cloud Computing (SIDSCC)", *Proceedings of the 2014 International Conference on Computational Science and Computational Intelligence*,Las Vegas, NV, USA, 10-13 March 2014, pp. 135-141.

BBC (2014), “Apple to tighten iCloud security after celebrity leaks”, http://www.bbc.co.uk/news/technology-29076899, 5 September, 2014, accessed 2 September, 2014.

Bharadwaja, S., Sun, W., Niamat, M. and Shen, F. (2011), "Collabra: A Xen Hypervisor Based Collaborative Intrusion Detection System", *Proceedings of the 8th International Conference on Information Technology: New Generations (ITNG)*, Las Vegas, NV, USA, 11-13 April, 2011, pp. 695-700.

Chawathe, S.S. (2012), "Fast Fingerprinting for File-System Forensics", *Proceedings of the Conference on Technologies for Homeland Security (HST)*, Waltham, MA, USA, 13-15 November, 2012, pp. 591-596.

Crossley, R.,Asimakopoulou, E.,Sotiriadis, S. and Bessis, N. (2013), "A study on metadata tagging for tracking original file information within the cloud", *Proceedings of the 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, Compiegne, France, 28-30 October, 2013, pp. 453-456.

da Cruz Nassif, L.F. and Hruschka, E.R. (2013), "Document Clustering for Forensic Analysis: AnApproach for Improving Computer Inspection", *IEEE Transactions on Information Forensics and Security*, 8 (1), January 2013, pp. 46-54.

Driftnet (2014), http://freecode.com/projects/driftnet , accessed 2 September, 2014.

Edmundson, D. and Schaefer, G. (2013), "Fast Mobile Image Retrieval", *Proceedings of the International Conference on Multimedia and Expo Workshops*, San Jose, CA, USA, 15-19 July, 2013, pp. 1-6.

Ficco, M.,Tasquier, L. and Aversa, R. (2013), "Intrusion Detection in Cloud Computing", *Proceedings of the 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, Compiegne, France, 28-30 October, 2013, pp. 276-283.

Haggerty, J. And Taylor, M. (2007), "FORSIGS: Forensic Signature Analysis of the Hard Drive for Multimedia File Fingerprints", in *IFIP International Federation for Information Processing, Volume 232, New Approaches for Security, Privacy and Trust in Complex Environments*, Venter, H., Eloff, M., Labuschagne, L., Eloff, J. & von Solms, R. (eds.), (Boston, Springer), pp. 1-12.

HTTP Acrchive (2014), http://httparchive.org/interesting.php?a=All&l=Aug%2015%202014&s=Top1000 , accessed 2 September, 2014.

Huang, W. and Yang, J. (2010), "New Network Security Based On Cloud Computing", *Proceedings of the 2ndInternational Workshop on Education Technology and Computer Science*, Wuhan China, 6-7 March, 2010, pp. 604-609.

Kelion, L. (2014), “Microsoft tip leads to child porn arrest in Pennsylvania”, http://www.bbc.co.uk/news/technology-28682686, 6 August, 2014, accessed 2 September, 2014.

Khakpour, A.R. and Liu, A.X. (2012), "First Step Toward Cloud-Based Firewalling", *Proceedings of the 31stInternational Symposium on Reliable Distributed Systems*, Irvine, CA, USA, 8-11 October, 2012, pp. 41-50.

Kurdi, H.,Enazi, M. and Al Faries, A. (2013), "Evaluating Firewall Models for Hybrid Clouds", *Proceedings of the 2013 European Modelling Symposium*, Manchester, UK, 20-22 November, 2013, pp. 514-519.

Liu, S-T. and Chen, Y-M. (2012), "MalPEFinder: fast and retrospective assessment of data breaches in malware attacks", *Security and Communication Networks*, 5 (8), August 2012, pp. 899-915.

Liyanage, G. and Fernando, S. (2013), "Firewall Model for Cloud Computing", *Proceedings of the 8th International Conference on Industrial and Information Systems,*Peradeniya,Sri Lanka, 18-20 August, 2013, pp. 86-91.

Ming Li, Shucheng Yu, Kui Ren, Wenjing Lou and Y. Thomas Hou (2013), "Toward Privacy-Assured and Searchable Cloud Data Storage Services", *IEEE Network*, July/August 2013, pp. 56-62.

Mohamad, K.M. and Deris, M.M. (2009), “Detecting JPEG JFIF Header Using FORIMAGE-JPEG”, *Proceedings of the 5th International Conference on INC, IMS and IDC*, 25-27 August, 2009, Seoul, Korea, pp. 1693-1698.

Meng, Y., Li, W. and Kwok, L-F (2013), "Towards Adaptive False Alarm Reduction Using Cloud as a Service", *Proceedings of the 8th International Conference on Communications and Networking in China (CHINACOM)*, Guilin, China, 14-16 August, 2013, pp. 420-425.

Nikolai, J. and Wang, Y. (2014), "Hypervisor-based Cloud Intrusion Detection System", *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, Honolulu, HI, USA, 3-6 February, 2014, pp. 989-993.

Oktay, U. and Sahingoz, O.K. (2013), "Proxy Network Intrusion Detection System for Cloud Computing", *Proceedings of the International Conference on Technological Advances in Electrical, Electronics and Computer Engineering*, Konya, Turkey, 9-11 May, 2013, pp. 98-104.

Perera, S., Kumarasiri, R., Kamburugamuva, S., Fernando, S.,Weerawarana, S. and Fremantle, P. (2012), "Cloud Services Gateway: A tool for exposing Private Services to the Public Cloud with fine-grained Control", *Proceedings of the 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, Shanghai, China, 21-25 May, 2012, pp. 2237-2246.

Pierris, G. and Vidali, S. (2012), "Forensically Classifying Files Using HSOM Algorithms", *Proceedings of the 3rd International Conference on Emerging Intelligent Data and Web Technologies*, Bucharest, Romania, 19-21 September, 2012, pp. 225-230.

Schaefer, G.,Edmundson, D., Takada, K., Tsuruta, S. and Sakurai, Y. (2012), "Effective and Efficient Filtering of Retrieved Images based on JPEG Header Information", *Proceedings of the 8th International Conference on Signal Image Technology and Internet Based Systems*, Naples, Italy, 25-29 November, 2012,pp. 644-649.

Shabtai, A., Potashnik, D., Fledel, Y., Moskovitch, R. and Elovici, Y. (2011), "Monitoring, analysis, and filtering system for purifying network traffic of known and unknown malicious content", *Security and Communication Networks*, 4 (8), August 2011, pp. 947–965.

Sportiello, L. and Zanero, S. (2011), "File Block Classification by Support Vector Machines", *Proceedings of the 6thInternational Conference on Availability, Reliability and Security*, Vienna, Austria, 22-26 August, 2011, pp. 307-312.

Yu, S., Doss, R., Zhou, W. and Guo, S. (2013), "A General Cloud Firewall Framework with Dynamic Resource Allocation", *Proceedings of IEEE ICC 2013 - Communication and Information Systems Security Symposium*, Budapest, Hungary, 913 June, 2013, pp. 1941-1945.