NDLERF

Cloud Authentication and Forensics

Kim-Kwang Raymond Choo
Jill Slay
(Editors)

Monograph Series No. 69

Funded by the National Drug Law Enforcement Research Fund
An Initiative of the National Drug Strategy

Cloud Authentication and Forensics

Kim-Kwang Raymond Choo
Jill Slay
(Editors)

Funded by the National Drug Law Enforcement Research Fund, an initiative of the National Drug Strategy

Produced by the National Drug Law Enforcement Research Fund (NDLERF)
GPO Box 1936, Canberra, Australian Capital Territory 2601

© Commonwealth of Australia 2016

ISSN: 1449-7476

The research on which this report is based was funded by the National Drug Law Enforcement Research Fund, an initiative of the National Drug Strategy.

Contents

Figures

Tables

Acknowledgements

This report is based on research funded by the National Drug Law Enforcement Research Fund (NDLERF).

We would like to thank the following for their input and assistance on the research:

- members of the Project Reference Group (ACT Police, Australian Federal Police, NDLERF, Queensland Police, South Australia Police); and
- Dr Maria Borzycki (NDLERF), Monica Boyle (NDLERF) and Katalina Bradley (NDLERF Secretariat).

Chapter 1: Introduction

Ben Martini, Kim-Kwang Raymond Choo and Jill Slay

Cloud computing is becoming increasingly prevalent, with both individuals and enterprises using cloud services for numerous purposes on a daily basis. The growth in the use of cloud computing has resulted in an increased need for forensic investigations involving cloud technologies. This need has spawned the growth of research in cloud forensics. Although this domain remains somewhat immature, several authors have undertaken research in the field.

A 2014 draft report by the National Institute of Standards and Technology entitled *NIST Cloud Computing Forensic Science Challenges* highlights the challenges experts face in responding to cloud computing incidents. The document identifies 65 challenges, many of which might be eased by further research. The large number of open challenges presented in the report demonstrates that, at the time of the research, cloud forensics remains an unresolved and somewhat under-researched area of enquiry.

Challenges faced by forensic investigators in cases involving the use of cloud services which include technical, legal and policy issues are well documented (Shariati, Dehghantanha & Choo 2016), but Martini and Choo (2013 2014a) noted that the number of academic publications with a technical focus on cloud forensics is relatively small compared to those in established disciplines such as information and network security. It was also pointed out that only:

> …in recent months, a small number of papers discussing the forensic collection of cloud storage products have appeared, and their focus is on the client side digital forensic process assumedly due to the difficulties in obtaining access to a cloud providers data centre to conduct server analysis (Martini & Choo 2013).

The first cloud forensic framework was proposed by Martini and Choo (2012), and was subsequently validated using ownCloud (Martini & Choo 2013), Amazon EC2 (Thethi & Keane 2014), vCloud (Martini & Choo 2014b), XtreemFS (Martini & Choo 2014c), Ubuntu One (Shariati, Dehghantanha, Martini & Choo 2015), SugarSync (Shariati, Dehghantanha & Choo 2016), OneDrive, Box, GoogleDrive and Dropbox (Daryabar, Dehghantanha, Eterovic-Soric & Choo 2016). Another cloud forensic framework was proposed a year later by Quick and Choo (2013a) and validated using Dropbox (Quick & Choo 2013b), SkyDrive (Quick & Choo 2013a) and Google Drive (Quick & Choo 2014). These two frameworks were subsequently merged into one (Quick, Martini & Choo 2014).

Shariati, Dehghantanha and Choo (2016) posited that:

> The cloud forensics 'space' can be seen as a race, not only to keep up with device (ie hardware) and software releases by providers, but also to keep up with new cloud services that may result in a variation of the type of data that can be collected. The challenges faced by digital forensic researchers and practitioners are compounded due to the constant evolution of cloud security threats and vulnerability, partly in response to defensive actions or crime displacement.

It is undeniably important for the forensic practitioner and research communities, as well as policing communities, to keep pace with technological advancements. In this research we therefore seek to address the following research questions:

1. What are the contemporary challenges in the cloud computing environment?
2. How can modern cloud-based authentication schemes be exploited for the purpose of forensic investigations?

Chapter 2: Literature review

Heba El-Fiqi and Jill Slay

Martini and Choo (2014a) reviewed some of the most prominent technical publications to date in the field of cloud forensics. Their review noted that some authors discuss 'forensics on the cloud' (ie using cloud computing as a service to conduct digital forensics) as opposed to 'forensics in the cloud' (ie cloud computing as an evidence source for forensic investigations). The literature review in this chapter report focuses on the latter, and as such we do not discuss the former because the techniques and challenges are mostly dissimilar.

Cloud forensic investigations face multiple challenges which cannot be addressed directly by traditional digital forensics techniques. Ruan et al. (2013) conducted a survey about cloud forensics in which 257 IT professionals participated. Based on the participants' responses, and using the NIST Cloud Computing Reference Architecture (Liu et al. 2011), Ruan et al. (2013) proposed a definition for cloud forensics:

> Cloud forensics is the application of digital forensic science in cloud computing environments. Technically, it consists of a hybrid forensic approach (e.g., remote, virtual, network, live, large-scale, thin-client, thick-client) towards the generation of digital evidence. Organizationally it involves interactions among cloud actors (i.e., cloud provider, cloud consumer, cloud broker, cloud carrier, cloud auditor) for the purpose of facilitating both internal and external investigations. Legally it often implies multi-jurisdictional and multi-tenant situations.

In a normal digital forensic investigation, the investigator has full access to the suspect machine. Network logs, router logs, process logs, and hard disk are physically accessible to the investigator. Evidence collection and preservation can be managed according to the investigation's requirements. On the other hand, if the crime occurred through the cloud, there are a number of issues that create obstacles to traditional digital forensic procedures. Examples of such obstacles are (Ruan et al. 2013):

- distributed physical locations of the infrastructure and storage;
- physical locating and controlling of the data;
- the absence of a standard interface;
- handling legal issues relating to multiple ownerships, multiple jurisdictions and multiple tenancies;
- CSP collaboration cannot be guaranteed; and
- evidence extraction and data recovery.

The following sections summarise the most common challenges that are faced by those conducting investigations in cloud-based environments.

2.1: Forensic data acquisition

Evidence acquisition is a critically important step in any investigation. The complex nature of the cloud has added further challenges to this process. Traditional digital forensics acquisition tools may not be sufficient in the investigation of some cloud-based crimes. With the increase in cloud related crime, the need to acquire evidence types which are located in the cloud becomes more important. 'There is no doubt that the admission of evidence obtained from the cloud will be of even greater significance in the future.' (Mason and George 2011).

There are three different possible states for evidence data in a cloud environment: at rest, in motion or in execution (Birk 2011). The data is considered 'at rest' if it is allocated to disk space, either as database or any other file format. De-allocated disk space caused by deletion but which has not been overwritten is still considered as data at rest, as it can be reconstructed and accessed by the investigators. Data is considered as ‘in motion’ while being transferred over a network. Investigators can track data in motion by tracking the traces that have been left by the encapsulated protocols used for transferring the data. Finally, the data is considered ‘in execution’ if it is neither in rest nor in motion. Usually, that refers to processing the data. In order to investigate this type of data, a snapshot of the current state of the system is taken and then analysed by the investigators.

2.1.1: Physical inaccessibility

Unlike other types of forensic investigation, locating forensic evidence in a cloud computing environment is challenging due to the virtual and complex nature of that environment. The data can be distributed on different servers, where each of these servers may exist in a different country (Zawoad and Hasan 2013). In Birk 2011, the researcher discussed the impact that the lack of physical access to data servers has on collecting evidence in a cloud computing environment.

2.1.2: Incomplete control over evidence

In a traditional digital forensics case, the investigators have complete access to and control over the evidence once it has been located, as they can seize the electronic device that hosts the evidence. In cloud forensics, the investigators only have access to what the CSP provides them with. A considerable amount of the evidence can therefore only be acquired through CSPs, who have sole control over collecting and providing the evidence data to the investigators. This task is done by a CSP employee, who is not a certified investigator (Zawoad et al. 2013). That raises issues of trust, since it is the CSPs who have full control over the log files and customer details, which are crucial pieces of evidence in the investigation process. However, if CSPs link an account to an IP address and provide this information to the investigator, this information should be used as trusted information, as there is no other way to verify this information. Another aspect of CSP dependency is that investigators cannot always get the information that they need from CSPs, since CSPs are not always obligated to provide that information, especially if they are located in different countries and are under different legal regimes (Zawoad et al. 2013).

2.1.3: Volatile data

Rapid elasticity is one of the essential services of cloud computing, and raises a further challenge for forensic investigators in the collection of evidence. In traditional digital forensics, volatile data—which are data that cannot remain after powering the machine off—is big concern for investigators and needs careful and quick handling. In a cloud computing environment, the rapid elasticity feature that enables quick provision and release of services, can result in ‘deleted data being made irrecoverable very quickly’ (Martini et al. 2013).

The use of the virtualization feature to create a virtual machine allows the possibility of losing evidence such as registry entries or temporary internet files, as these will be lost when the user turns off their machine. If an attacker launches an attack using a virtual machine, and then turns it off and deletes the data, that places an extra challenge on the investigator’s shoulders. Logging files in virtual machines have attracted the interest of a number of researchers (Birk 2011, Slusky et al. 2012, Poisel et al. 2012) and will be discussed further under the Logging section below.

Logging is limited either by time or by logging file size. For example, logs of actions that happened before a certain date may be deleted; or, once the logging files reach a certain size limit, older logging entries may be deleted to provide space to accommodate newer logging entries.

2.1.4: Trust issues

One of the conditions of the admissibility of evidence in court is reliability. Both the authentication and the integrity of the evidence need to be demonstrated to the court. In a research paper by Dykstra and Sherman, they discussed the trust issue associated with evidence acquisition in two hypothetical case studies (Dykstra and Sherman 2011). They identified a number of questions that the defence may raise against evidence which has been collected in a cloud environment. These questions include: How can they tell if the CSP has provided a complete and authentic forensic copy of the data? Can the cloud technician and his tools be trusted? Is the integrity and authenticity of the data in question? With regard to the physical location of the data, they may ask: Where was the data located, and who had access to it? Additionally, there may be questions raised about how to prove the consistency of date and time stamps of data collected from different systems.

Assuming that the court trusts the cloud service provider and the technician who collected the evidence from the cloud, the authentication and integrity of evidence collected over the cloud are still crucial matters that need to be demonstrated to the jury. In another research study conducted by Dykstra and Sherman, researchers investigated trust issues in the case of acquiring forensic evidence from an infrastructure-as-a-service cloud computing environment (Dykstra and Sherman 2012) with a customer using a virtual machine scenario. In order to avoid the inadmissibility of evidence gathered in a cloud environment, there was a need to ensure that the acquisition method was trusted over all of the accumulative layers. For example, if the evidence provided is a packet captured on the network level, it is only necessary to validate the trust over the network level. If the evidence is acquired via virtual machine introspection, then the investigator needs to demonstrate to the court that the evidence acquisition was done under trusted accumulated layers started by hypervisor, then the host OS, the hardware, and finally the network layer.

2.1.4.1: Integrity of evidence

A number of digital signature based techniques have been suggested by researchers for integrity preservation of evidences in the cloud computing environment. Hegarty et al. (2009) proposed a distributed signature detection framework as a forensic analysis tool. This framework has three main components: (1) an 'initialiser' that contains the hash value of the target buckets, a (2) 'forensic cluster controller' that allocates the task of finding the matching files through the third component, which is (3) 'analysis nodes' . Another solution had also been proposed by Song et al. (Xiuli Song 2013) that encrypts the electronic evidence using Chaos Block Encryption Algorithm (CBEA). This encrypted/ciphered evidence is sent to the cloud storage server. Users can then verify the integrity of the evidence through a Homomorphic Signature Algorithm (HSA). It was not clear in that research (Xiuli Song 2013) how a third party auditor or investigator would be able to decipher the encrypted evidence in the place of the user.

As a part of Zawoad et al. (2013) solution Secure Logging-As-A-Service (SecLaaS) for logging as forensic evidence, they introduced the Proof of Past Log (PPL) scheme to address the issue of the integrity of logs as evidence. This scheme can be used to prove that the data has not been altered, either by investigators or by CSPs. This is ensured by saving an accumulated proof of log file. One limitation of this solution is that only the auditors can read all of the logged data, and not the users themselves (Hogben and Royer 2013) .

The Trusted Platform Module (TPM) had been originally proposed by Santos et al. (Santos et al. 2009) as a part of a trusted cloud computing platform (TCCP) to accommodate the needs for confidentiality and integrity in the Infrastructure as a Service (IaaS) environment. Later on, Birk and Wegener (Birk and Wegener 2011) suggest TPM as a solution that can detect changes to previous hardware configurations and provide to customers the integrity of a running virtual instance, trusted logs, and trusted deletion of data. Additionally, TPM can provide machine authentication, hardware encryption, signing, secure key storage, and attestation (Zawoad and Hasan 2013, Dykstra and Sherman 2012). Whereas TPM had been designed to support a single OS on a single platform, a similar module, Trusted Virtual Environment Module (TVEM), had been proposed by Krautheim et al. (2010) to support trust in a virtual environment, independently of authentication of the the hosting platform.

Dykstra and Sherman (2012) criticized the use of TPM for use in cloud forensics for a number of reasons: (1) the original purpose for these modules was to provide trusted guests VMs rather than as an attestation of the host platform; (2) a number of TPM limitations, such as being able to modify a running proof without being detected by TPM; and (3) for TPM to be effective, it needs to be installed in each cloud server, which is currently difficult to achieve with the large amounts of heterogeneous and commercial hardware used by cloud vendors.

A TrustCloud framework to address the trust issue in the cloud has been proposed (Ko et al. 2011). This framework suggests detection rather than prevention, via accountability over five layers: System layer, Data layer, Workflow layer, Laws and regulations, and Policies layers. The aim of this framework is the integrity and accountability of data stored in the cloud, which aligns with the requirements of cloud forensics. However, this framework has yet to be implemented into a system where it can be validated against real life scenarios (Ko et al. 2011).

2.1.4.2: CSP dependability

Abbadi (2011) suggested the resolution of the trust issue in the cloud environment through two mutually dependent elements: (1) mechanisms and tools that facilitate the automation of managing, maintaining, and securing the infrastructure, and (2) establishing the trust by providing both customers and CSPs with techniques that facilitate the continuous evaluation of the operations of the cloud infrastructure. Abbadi and Alawneh (2012) have therefore proposed a framework that addresses the second element above, suggested by Abbadi (2011) through a virtual resource management tool called Virtual Control Centre (VCC). This tool provides resource management of VMs via the establishment of communication channels with physical servers. Through VCC, a chain of trust can be established between the end user and the physical layer. One of the advantages of that framework is the aspect of pushing more management power towards the users rather than towards CSPs. Such direction can help future forensic investigation and minimise the issues caused by dependency on CSPs.

Similarly, a cloud management plane has been proposed by Dykstra and Sherman (2012) that enables CSPs, customers, and investigators to collect VM images, networks, processes, database logs, and other digital evidence via this tool. Although a cloud management plan minimises the number of layers that need to be trusted for the investigation, it requires trust in the management plane itself, which is a potential vulnerability when compared to physically accessible data acquisition.

2.1.5: Bandwidth limitation for time-critical investigations

In an investigation scenario, a number of VM instances may need to be analysed. Each of these VMs may be a couple of gigabytes in size. This entire amount of data needs to be transferred through the internet to be analysed locally by the investigators. In time-critical investigations, there is a need for a large bandwidth to accommodate the required data transfer. The two major factors associated with the process of transferring the disk image are the file size and bandwidth speed limits. Additional costs associated with dedicated high-speed bandwidth should be considered, in order to facilitate forensics over the cloud environment.

2.1.6: Multi-tenancy and privacy

The shared-storage nature of cloud computing raises several issues related to forensic investigations: validity-of-the-warrant, privacy, and authenticity. In order to issue a warrant, the identification of a specified location that has a significant probability of containing evidence is a necessary step. However, such identification becomes a challenge in a cloud environment. A single physical disk space on a server may contain data which belongs to multiple customers, and data belonging to a single file of a single customer may be physically distributed among different servers that are located in different countries.

Trying to overcome such a problem by attempting to issue overly broad warrants is not legally acceptable. In some cases, the warrant may be suppressed; for example, a warrant which affects the privacy of other tenants who are not involved in the investigation, or a warrant which allows for searching of other data belonging to the suspect that is not related to the investigation (Orton et al. 2013). Additionally, there is a need to show that authenticity has been preserved—that is, that the data obtained from that multi-tenancy shared storage belongs to the defendant and only the defendant (Orton et al. 2013). Furthermore, multi-tenancy also involves an additional risk—that of side-channel attacks.

A multi-tenancy model has been proposed to maintain the privacy of users in multi-tenant environments by means of data encryption and information disassociation (Zhang et al. 2009; Shi et al. 2010). However, this model aims to protect users' privacy in the case of intruders trying to access their data. Others have suggested the isolation of cloud instances to protect evidence from containment by means of moving other clouds from the same node that contains the suspicious activities (Zawoad and Hasan 2013; Delport et al. 2011).

2.2: Logging

Despite the importance of analysing logs for forensic investigations, collecting logging information for forensic analysis faces a number of challenges because of the special characteristics of the cloud. Different researchers have discussed a number of these logging challenges (Marty 2011; Zawoad et al. 2013).

Logging information is distributed across multiple files on multiple servers. In addition, the logging information of multiple users may exist on a single server, and the logging information of a single user may be distributed among over multiple servers (Marty 2011; Zawoad & Hasan 2013). A further challenge for cloud forensic investigators is in collecting required evidence.

Logged records can be lost for one of two reasons: first, due to the termination of a virtual machine or the unexpected termination of a server due to inactivity (Marty 2011), or second, due to the fact that logs cannot be stored forever due to size limitations. Log file size is controlled either by the elapse of a specified length of time or by a prescribed size limit. Reaching one of these conditions causes older records to be deleted.

Logs are generated in each layer of the cloud, and a single software service provided by the CSP causes all the stacks of the underlying cloud structure to generate logs all the way from the application layer up to the network layer. Collecting each of the logs from these multiple layers is a challenging task for the investigators.

Although application developers, system administrators, and security analysts each need to access only a subset of the logging details, forensic investigators need to have access to all of these logs. Some of the logs are not available through the customer, as in the case of Amazon AWS PaaS (Marty 2011). In these cases, investigators need to access these logs through the CSP, along with all of the challenges associated with that task. This leads to the next challenge, which is CSP dependency.

Marty (2011) proposed a solution that addresses a number of logging challenges at the same time—that is, decentralization of log information, multiple tiers and layers, and limited access to or absence of logs. This solution goes through three steps: (1) enabling logging on all infrastructure components able to collect logs; (2) establishing a reliable, efficient, encrypted transport layer, in addition to a central log collector, to which all logs are transferred; and (3) ensuring that the required information exists in the collected logs. However, the log management solution proposed by Marty (2011) lacks the ability to log network usage, file metadata, or process usage, among other important forensic information, as reported by Zawoad and Hasan (2013).

In order to collect log-based evidence, forensic investigators must rely on CSPs to provide them with logging information, as the investigators do not have access to validate the evidence data themselves. They need to trust the CSP technician yet again (Zawoad et al. 2013). Defendants may then claim that CSPs have altered the log, or that investigators have planted evidence.

Birk and Wegener (2011) have suggested that customers of public Software as a Service (SaaS), if involved in a forensic investigation, should be able to retrieve important information through an API offered by their CSP.

This API should provide the customers with access, error, and event logs. In PaaS environment, a central logging server that the customer controls can be applied as a solution. However, the verification of data alteration or the risk of eavesdropping while data is being transferred may still be in issue. To addres this, Birk and Wegener (2011) have suggested adding authentication and encryption to prevent attackers from viewing or altering the logs while they are being transferred to the central sever. A very similar approach had been proposed by Damshenas et al. (Damshenas et al. 2012) suggesting the use of APIs in both PaaS and SaaS models in order to provide the customers with status data related to their usage in read-only and on-demand modes.

As every service by a different CSP is documented through logs generated by that CSP, some of these logs miss important pieces of information, such as when, where, why, and by whom these actions have been taken (Zawoad & Hasan 2013). Such information, while not important enough for that particular CSP to document, may nonetheless be crucial for a cloud-based crime investigation.

If all of the previous problems have been addressed, investigators still face the critical issue of generating a unified digital forensic tool, due to the different and random log formats generated by different CSPs.

2.3: Preserving chain of custody

As cloud evidence may in many cases have to be collected by the CSP technicians rather than a forensics investigator, this step may create a weakness in the chain of custody (Orton et al. 2013) and may raise questions about the original state of the evidence before the collection time (Taylor et al. 2010).

Data provenance has been suggested by a number of researchers as a solution to this challenge of preserving the chain of custody (Lu et al. 2010, Birk and Wegener 2011, Zawoad and Hasan 2013, Li et al. 2013). Gaining the history of an object via its provenance can deliver valuable forensic information, such as the possession of data at a certain point in time and the chronological access history of the data.

Group signature technique has been used by Lu et al. (2010) to propose a secure provenance scheme that is specially designed for a cloud computing environment. That model lacked the flexibility of accommodating multiple attributes for a single user, but this was addressed later on by Li et al. (2013), who employed both group signature and attribute-based signature techniques to address fine-grained access control systems. The idea of attribute-based signatures allowed the identification of the user (the signer) by a set of attributes rather than by a single string representing the signer's identity (Li et al. 2013). Due to the value and advantages of secure provenance in the cloud environment, researchers are criticising CSPs for not implementing provenance techniques (Birk and Wegener 2011, Zawoad and Hasan 2013). 2 2 Nevertheless, employing a trained, trusted, and qualified technician to collect the evidence data in the course of an investigation should be considered (Grispos et al. 2012, Simou et al. 2014).

2.4: Limitation of current forensics tools

In 2012, Dykstra and Sherman conducted a comparative study of the performance of existing forensic tools (Dykstra and Sherman 2012). They evaluated the performance of the remote acquisition feature of two forensic tools, Guidance EnCase and AccessData FTK, and three memory acquisition tools, Fastdump, Memoryze, and FTKImager, in collecting evidence in an IaaS-based crime scenario using Amazon EC2. Although Encase and FTK were technically able to acquire evidence remotely and to produce a correct timeline of user activity, technology alone was not sufficient to provide trustworthy collection of the evidence due to the dependence of multiple layers of trust through the acquisition. Additionally, existing tools cannot be installed in other cloud environments or in models such as Microsoft's Azure and Google's AppEngine (Dykstra and Sherman 2012). Another issue with existing forensic tools is that, although it is possible to collect virtual disk images from Amazon web services, validation of the downloaded images is not possible due to the unavailability of a hash key from the original source (Dykstra and Sherman 2013).

In 2012, Dykstra and Sherman proposed a cloud management plane that aims to facilitate data forensics acquisition over cloud computing environments (Dykstra and Sherman 2012). This management plane would be able to provide CSPs, users, and investigators with log files, disk images, and packet captures on demand. However, it would also add a potential vulnerability if not protected by CSPs, as attackers could use it to access, alter, and manage the virtual assets.

In 2013, Dykstra and Sherman then introduced a new forensic tool, FROST, to overcome some of the limitations of existing forensic tools in addressing cloud-based challenges in evidence collection (Dykstra & Sherman 2013). Virtual disks, API logs, and guest firewall logs can be acquired through this tool. The new tool is able to offer an alternative to the current solution to this issue of layers of trust in traditional digital forensic tools, which is by means of a single layer of trust—the need to trust the CSP. The FROST tool is designed for the OpenStack platform. Although the advantages of speed and trust solutions proposed by FROST, there still some limitations in the logging mechanisms such as: (1) There is no way to know if there intentionally missing entries in the log without accessing it. (2) Logs can be accessed and modified by users without being detected later on by the investigator who collects the data. Data preservation is still another issue that need to be handled separately as well (Dykstra and Sherman 2013). Furthermore, the functionality of FROST cannot be guaranteed with the expected and continued development of OpenStack.

2.5: Cross-border law (multi-jurisdictional)

Servers used by cloud service providers are distributed worldwide. They may exist in the same country where the CSP head office exists, or they may be located in one or more other countries. Researchers have suggested a globally unified legal regime for cloud-based crimes (Biggs and Vidalis 2009) and that international laws be developed that do not violate any laws or regulations in any jurisdiction (Ruan et al. 2011, Sibiya et al. 2012, Simou et al. 2014). However, there is as yet no clear proposal as to how this could be made a reality (Zawoad and Hasan 2013).

2.6: Compliance issues (privacy and service level agreements)

A Service Level Agreement (SLA) provides the contract between the cloud customer and the cloud service provider. Interestingly, possible forensics investigations have not been considered in existing contracts. Ruan et al. (Ruan et al. 2011) highlighted a number of terms relating to cloud forensics which are not part of current agreements but would need to be included in future SLAs. These include: (1) What are the types of services, support and access that the CSP will provide to this customer in the case of a forensic investigation. (2) What are the boundaries of trust, roles, and responsibilities of the CSP and the customer in the case of a forensic investigation? (3) What types of legal regulations, confidentiality, and privacy provisions are there for customers in the case of a cloud forensic investigation being conducted in a multi-jurisdictional environment? (4) What types of legal regulations, confidentiality, and privacy of provisions are there for customers in the case of a cloud forensic investigation being conducted in a multi-tenant environment?

SLAs should be revised in a way that facilitate the requirements of forensics investigations without violating customers' rights, confidentiality, or privacy (Zargari and Benford 2012). Damshenas et al. (2012) suggested updating SLAs according to the additional technologies or procedures that may be needed for investigations. For example, if persistent storage has been used as a solution for volatile data as evidence, then a term could be included assuring that all customers' data will be triple-wiped within a week of end of contract. For more privacy preservation, customers' data could in addition be kept encrypted on the persistent storage. Birk and Wegener suggested employing trusted third-party auditing to ensure the quality of SLAs (Birk and Wegener 2011).

Chapter 3: The decline of passwords and the need for legal reform

Ben Martini, Quang Do and Kim-Kwang Raymond Choo

This chapter is based on materials published in Martini B, Do D, and Choo KKR, Digital forensics in the cloud era: The decline of passwords and the need for legal reform, a forthcoming paper in the *Trends & Issues in Crime and Criminal Justice* series. Online and cloud computing services are increasingly prevalent; for many people, they are integral to communication in daily life. From a criminal justice perspective, this makes these services key sources of evidence for prosecuting both traditional and online crime (Martini & Choo 2014b; Quick, Martini & Choo 2014). However, the successful prosecution of individuals who commit crimes involving electronic evidence relies upon two major factors. The first is appropriately resourced law enforcement agencies with forensic practitioners who are able to collect, analyse and present the evidence (Quick & Choo 2014). The second is a legislative framework that facilitates the collection of evidence in the modern era, particularly where much of the relevant electronic evidence may be stored beyond the jurisdiction of the investigating law enforcement agency, and more generally beyond the nation's borders. Given the relatively recent advent and changing face of cloud computing technologies and their widespread use, it is important to discuss these factors when looking not only at the challenges of collecting evidence from cloud computing systems in the current statutory environment but also the technical challenges of authentication in forensic collection—particularly as cloud service providers continue to enhance the security of their services.

This chapter first discusses the various provisions for the search and seizure of evidence which are currently available to Australian law enforcement agencies. It then focuses on the increasing emphasis being placed on the security of online services in recent times, and the effect this has had on authentication. Where digital forensics are used to collect evidence of a crime within a law enforcement agency's physical jurisdiction, it is common practice to take a physical bit-stream image of the storage in the devices to be forensically analysed. The methods used to collect this image do not generally require authentication, as the process requires physical control of the device. In the online environment, users—and, generally, forensic practitioners—do not have physical access to the storage devices hosting their data.

However, there is still a need for a copy of the electronic evidence to be analysed and, ultimately, presented.

It has become relatively common for forensic practitioners, particularly those outside the jurisdiction of the cloud service provider, to obtain copies of electronic evidence using similar technical means to the user (eg a client-visible application programming interface, or API). There is some doubt as to the forensic soundness of this process, particularly in terms of the lack of preservation measurements (eg cryptographic hashes–acting as a unique identifier–matching for the source and the forensic image, where the forensic image is an identical duplicate of the data source) and the increased potential for contamination, depending on the software tools used. As such, the most appropriate means of undertaking the collection of such evidence remains undetermined.

Regardless of the tools or methods used, most online services require authentication for every data access request. This relies on the practitioner gaining access to a user's credentials. These credentials have traditionally been a username and password, and law enforcement agencies have developed various methods, such as extracting them from application caches, to obtain them from suspects or their devices.

As service providers continue to improve the security of their services, there are fewer avenues available for collecting these credentials. This is, at least in part, due to the increase in token-based authentication systems in contemporary apps. A token-based system typically requires the user's credentials only once, at initial logon. These credentials are then used to obtain one or more tokens that are used for future authentication as required. This is discussed in greater detail in the Authentication Systems section of this paper.

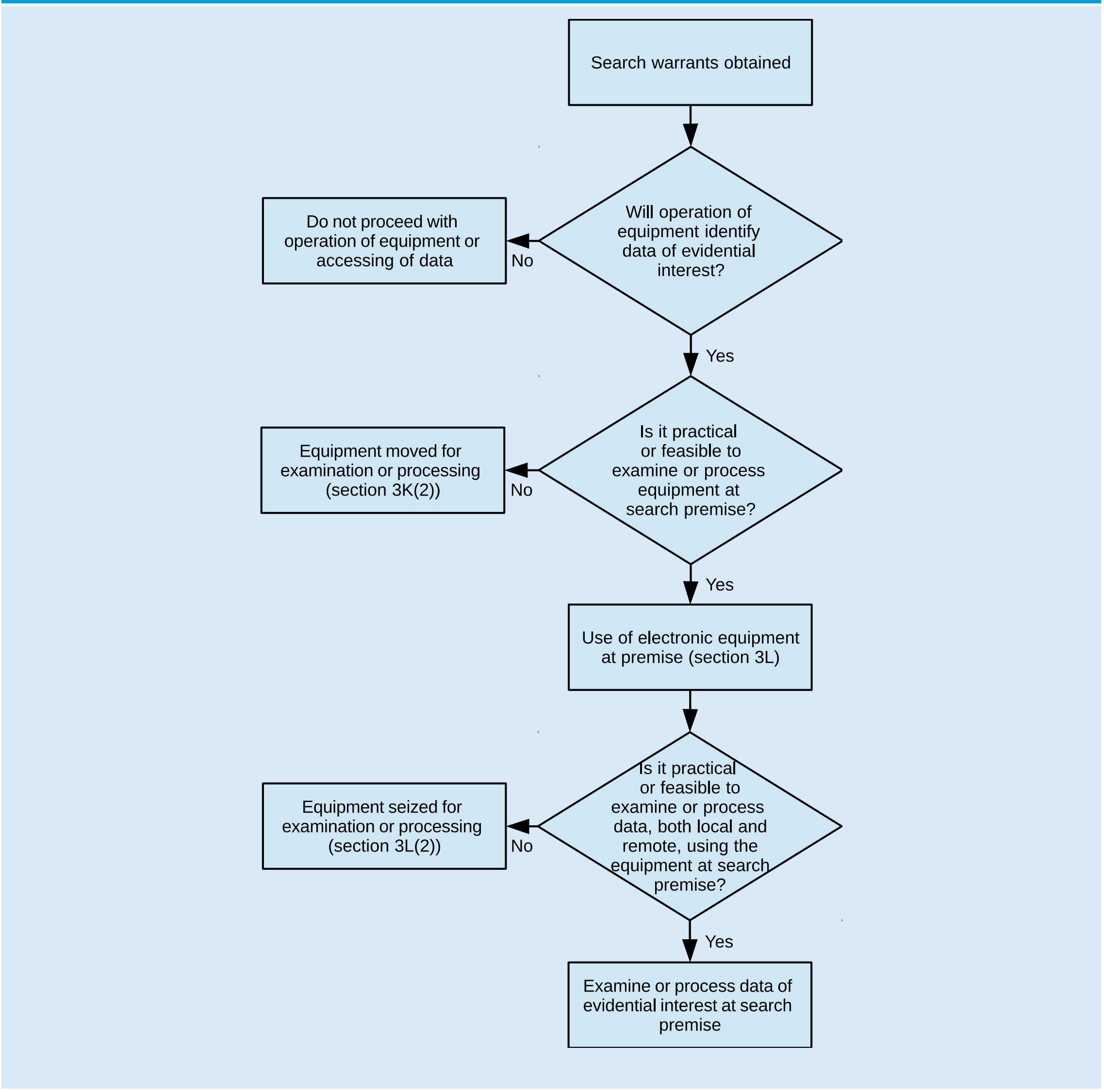
To ensure law enforcement agencies are able to maintain and improve their current capabilities in evidence collection from online and cloud services, they must understand the operation of contemporary authentication systems. This paper assists in reducing the gap in documented knowledge of this area.

3.1: Evidence collection: The Australian legislative perspective

Before data can be forensically analysed, it must be identified and preserved. As noted in an earlier work (Hooper, Martini & Choo 2013), law enforcement agencies can access certain data without the need for a formal warrant—for example, cloud service providers are obliged to render 'reasonable assistance' under section 313 of the *Telecommunications Act 1997* (Cth) for the purposes of, among other things, enforcing the criminal law and laws imposing pecuniary penalties, assisting to enforce the criminal law of a foreign country, protecting the public revenue and safeguarding national security. It is also possible for Australian law enforcement agencies to apply for a warrant under the Telecommunications (Interception and Access) Act 1979 (Cth) to intercept telecommunications in real time, which may allow access to data stored on a cloud server.

Australian law enforcement agencies can also apply for a search warrant requiring service providers to disclose content data which they store. The Crimes Act 1914 (Cth) contains various provisions concerning the search and seizure of evidence in federal criminal matters, with specific provisions relating to electronic searches, including: when search warrants can be issued (s 3E), what the search warrant authorises(s 3F), the specific powers available to constables executing warrants (s 3J), the use of equipment such as laptops with forensic-imaging programs to examine or process items (s 3K), and the use of electronic equipment at the premises (s 3L). A constable is defined in section 3 as a member or special member of the Australian Federal Police or a member of the police force or police service of a state or territory. In this chapter, the terms 'executing officer' and 'constable' are used interchangeably.

Figure 3.1: Current Australian search and seizure powers under the Crimes Act 1914 (Cth)


Search warrants obtained
Will operation of equipment identify data of evidential interest?
No
Do not proceed with operation of equipment or accessing of data
Yes
Is it practical or feasible to examine or process equipment at search premise?
No
Equipment moved for examination or processing (section 3K(2))
Yes
Use of electronic equipment at premise (section 3L)
Is it practical or feasible to examine or process data, both local and remote, using the equipment at search premise?
No
Equipment seized for examination or processing (section 3L(2))
Yes
Examine or process data of evidential interest at search premise

When evidence is likely to be found during the execution of a search warrant, the executing officer will state this in the affidavit supporting the warrant application and in its conditions. Other relevant information in the affidavit may include what type of remote data is expected to be found, the name of the company or entity hosting it, and where the data is hosted.

When a section 3E search warrant is executed, the suspect's electronic equipment, such as desktop computers and smart mobile devices, is examined at the search premises to identify and record evidence of historical access to remotely stored data—for example, a cloud storage app on the equipment may indicate that potentially incriminating data is stored in an online cloud storage account. If this is indeed the case, section 3L allows the executing officer to use the electronic equipment to access the remotely stored data, if he or she is satisfied that the equipment can do so and that the process for doing so meets the conditions of the warrant.

Where assistance is required to operate the equipment or to access specific data held on it, the executing officer can apply to a magistrate under section 3LA(1) for an order requiring a specified person to provide reasonable and necessary information or assistance to allow the executing officer to:

(a) access data held in, or accessible from, a computer or data storage device that:

  (i) is on warrant premises; or

  (ii) has been moved under subsection 3K(2) and is at a place for examination or processing; or

  (iii) has been seized under this Division;

(b) copy data held in, or accessible from, a computer, or data storage device, described in paragraph (a) to another data storage device;

(c) convert into documentary form or another form intelligible to a executing officer:

  (i) data held in, or accessible from, a computer, or data storage device, described in paragraph (a); or

  (ii) data held in a data storage device to which the data was copied as described in paragraph (b); or

  (iii) data held in a data storage device removed from warrant premises under subsection 3L(1A) (*Crimes Act 2014*).

Section 3K also permits equipment to be brought to the search premises in order to examine and process items, including any electronic devices, found there to determine whether they may be seized. As discussed below, it might be possible to search for authentication artefacts, such as tokens, during this process.

In the event that it is not feasible or practical to access the data using equipment located at the search premises (eg due to bandwidth constraints or concerns about the forensic soundness of the equipment), section 3L(2) of the Crimes Act permits the seizure of the equipment and any disk, tape or other associated device. Section 3K(2) allows items to be moved from the search premises to another place for examination for up to 14 days if it is reasonably suspected that the items constitute evidence. An executing officer may apply to an issuing officer for one or more extensions of that time, if he or she reasonably believes the item cannot be examined or processed within 14 days or within a previously extended time frame (s 3K[3B]).

If something is moved to another place to be examined or processed under section 3K(2) the executing officer must, if it is practical to do so:

(a) inform the person referred to in paragraph (2)(b) or (c) (as the case requires) of the address of the place and the time at which the examination or processing will be carried out; and

(b) allow that person or his or her representative to be present during the examination or processing (*Crimes Act 1914*).

However, the executing officer does not need to comply with these provisions if they believe on reasonable grounds that doing so might:

(a) endanger the safety of a person; or

(b) prejudice an investigation or prosecution (*Crimes Act 1914*).

Sections 3LAA and 3ZQV permit any items seized or moved to be used to access data, including data held elsewhere—for example, in an online cloud storage account—to determine whether data (ie evidence) is stored on or remotely accessible from the electronic equipment, and to obtain access to that data. The equipment can be used either before or after the expiry of the warrant. The legislation explicitly allows only equipment located at, moved or seized from the search premises to be used to access remotely stored data—in other words, a forensic copy of the data, or equipment belonging to the investigating law enforcement agency, must not be used to access the remotely stored data.

This restriction presents some issues in terms of forensic soundness best practice. It is commonly recognised that, where practical, the first step in handling an electronic device that may contain evidence is to preserve the evidence. This is most often achieved by taking a forensic image (a bit-for-bit copy) of the device's storage. Further analysis of the evidence is then conducted on the forensic image, rather than on the source.

This helps avoid accidental modification of the evidence source and allows for more flexible handling of the evidential data—for example, it would allow a practitioner to boot a copy of the seized device in a controlled virtual environment.

Where they are available, it is also preferable to use tools that have been verified as forensically sound and are designed for use in forensic collection and examination. Such tools ensure evidence—including less obvious elements such as metadata and temporal data—is not inadvertently modified during the collection process. These tools may also use functions such as cryptographic hashes to demonstrate that a true copy of the remote evidence has been made. The only input data generally required by these tools to conduct a collection are the user credentials of the suspect user. These credentials are then used to authenticate access to the online or cloud service.

With this in mind, utilising the client application for the cloud service installed on the seized device—as may be required by sections 3LAA and 3ZQV—is not the most forensically sound option. The best solution would be to take the relevant authentication artefacts from the client device, and conduct the collection using law enforcement-controlled, forensically sound hardware and software. However, it is not clear whether law enforcement equipment—that is, equipment brought to the search premises—can be used to process and preserve remotely stored data or whether the executing officer must use the software on the moved or seized equipment to process and preserve the remotely stored data.

3.2: Authentication systems

As previously noted, the best way to collect forensically sound remote cloud data is to use the law enforcement agency's collection environment, with collection credentials sourced from client devices. To collect remote cloud data, practitioners must thoroughly understand how the authentication systems used by cloud services work, both to design a collection system and to understand the limitations of the authentication system (eg credential expiry) and its use in common collection operations.

Password-based authentication has been the de facto standard in computer system authentication since the introduction of multi-user computer systems in the 1960s. Initially, developers of online and cloud mobile apps often stored (or cached) password-based user credentials on the device. This allowed the app to access the user's credentials and make continuous requests for up-to-date data without user intervention. This caching was necessary primarily due to the 'always-on' nature of mobile apps, where users expect continuous updates from an online service, often requesting data that is protected by authentication. Recently, however, developers have had to consider other approaches to authentication.

The approach of storing user credentials in plaintext (ie unencrypted data) on the device presents a security risk to the user. Malicious parties, or apps they control, could be used to obtain the user's stored credentials. Using these credentials, the malicious party could impersonate the user when communicating with the service.

An example of this is the LinkedIn app for Android, which originally stored the user's credentials, including their password, in plaintext on the device (Ante 2011). This made it easy for appropriately resourced malicious parties to access the protected data the service stored for the user. As a result, developers were forced to deviate from authentication schemes that verified the user's username and password each time the app and the service communicated.

Token-based authentication is a popular contemporary authentication system that does not require the user's password-based credentials to be stored on the device. Rather than requiring the user's username and password for each session with a particular service, the user authenticates once with their username and password and receives a token (Satyanarayanan 1990). This token can be time-limited; for example, it may expire after 24 hours. The app can use the token to authenticate the user with the service as long as the token remains valid.

The user is generally required to reauthenticate with their password-based credentials to generate a new valid token upon expiration of the old token, or to utilise a 'refresh' token, as discussed later in the paper.

The advantages of token-based authentication schemes include:

- Users authenticate with their credentials only when their tokens expire, providing fewer opportunities for man-in-the-middle attacks to obtain the user's password-based credentials.
- User passwords are typically short in length and can be guessed, whereas authentication tokens are often comprised of a long string of characters. This makes it computationally infeasible to perform brute force attacks on services utilising token-based authentication. The reduced frequency with which users need to authenticate using their password also allows developers to harden this authentication interface.
- Tokens can be rescinded by the service at any time for security reasons. Tokens can also be given a subset of the user's privileges—for example, an instant messaging app authenticating with the user's Facebook account to retrieve the user's friends list can be given an access token with limited access to resources. In this case, the app would only be able to access the user's friends list and not all other user data available from Facebook. This process would be either impossible or very difficult to achieve if the user's password was shared.

3.2.1: Prominent authentication systems

To determine what the most prominent contemporary authentication systems were for online and cloud applications, this study analysed 10 popular Android apps (and their Windows equivalents where available), selected for their popularity and their use of remote authenticated services. The apps analysed are listed in Table 3.1.

Table 3.1: Apps selected for analysis (at 7 October 2014)

App	Version (Android)	Version (Windows Store)	Category (and Category Rank) in the Google Play Store
Box	3.2.1	2.0.0.12	Business (13th)
Facebook Messenger	10.0.0.17.14	N/A	Communication (1st)
Skype	5.0.0.49715	3.1.0.1005	Communication (2nd)
OneDrive	2.8.1	6.3.9600.16384	Productivity (9th)
Dropbox	2.4.5.10	2.0.0.0	Productivity (10th)
OneNote	15.0.3130.1014	16.0.3030.1024	Productivity (28th)
Facebook	15.0.0.20.16	1.4.0.9	Social (1st)
Instagram	6.4.4	N/A	Social (2nd)
Twitter	5.23.0	1.1.13.8	Social (4th)
LinkedIn	3.4	N/A	Social (6th)

On the Windows platform, official Windows Store apps were selected where available. Where apps were available on Windows but not available via the Windows Store, desktop apps were used. Instagram and LinkedIn were not available on Windows as desktop or Windows Store apps at the time of research. Facebook Messenger functions were integrated into the Windows Store Facebook app

3.2.1.1: Android

To determine the prominent online authentication systems used on Android devices, the selected popular apps were forensically analysed. The study utilised the latter examination and analysis stages of the method outlined in Martini, Do and Choo (2015) for the forensic analysis of cloud apps on Android. This primarily involved analysing the files stored by the apps on the Android device, examining the account management system and, where necessary, analysing the app's operation and/or code.

Android apps typically store authentication data in two locations: in the app's private storage folder on the device's internal storage, and by using Android's AccountManager API. AccountManager API is considered more secure than internal storage due to the additional layer of hardware-backed protection on credentials stored this way (if available on the device).

The first step in ascertaining which authentication systems are used by the selected apps was determining where each app stores the authentication data (ie the username and password or access tokens). The details of this examination are presented in Table 3.2.

Table 3.2: Locations and nature of authentication data on selected Android apps

App	Stored using AccountManager	Stored in internal storage	Token obfuscation	OAuth version
Box	No	Yes	No	2.0
Dropbox	No	Yes	No	1.0
Facebook	No	Yes	No	Derivative of 2.0
Facebook Messenger	No	Yes	No	2.0
Instagram	No	Yes	Yes	2.0
LinkedIn	No	Yes	No	2.0
OneDrive	Yes	No	No	2.0
OneNote	Yes	No	Yes	2.0
Skype	No	Yes	No	2.0
Twitter	Yes	No	No	2.0

The study's findings show that only three of the 10 apps selected utilised the AccountManager API to store their authentication data. The findings also show that none of the 10 selected apps stored the user's credentials (ie username and password) directly; rather, they stored authentication tokens. In addition, two of the 10 selected apps further obfuscated their stored authentication tokens in an attempt to thwart malicious attackers seeking to obtain and use these tokens.

All 10 Android apps appear to use the OAuth authentication protocol, with nine of the apps appearing to use OAuth version 2.0. Dropbox was the only service examined still using version 1.0 of the OAuth protocol. Based on the popularity of the selected apps and the fact that all 10 utilised the OAuth protocol, it is clear that OAuth has become the standard for the online authentication of Android apps. The OAuth protocol is discussed in greater detail later in this chapter.

3.2.1.2: Windows

The study utilised a similar approach on the Windows platform to determine which authentication systems are used by Windows apps. Apps may store their credentials in a number of locations on a desktop computer running the Windows OS. One commonly used location is the app's 'appdata' directory, located within the user's home directory.

Windows does not enforce suitable protections by default on folders or files in the appdata directory where credentials are stored. This makes the appdata directory a poor choice for storing user credentials. A more secure method of storing user credentials on Windows, commonly used by Windows store apps, is the Windows Credential Manager.

Credential Manager is the inbuilt credential storage system available in recent versions of Windows. It can securely store information such as passwords, usernames and access tokens. By default, credentials can be stored in Credential Manager as either web credentials or Windows credentials, and are saved to storage in an encrypted file.

One major difference between these types of credential storage in later versions of Windows is that the contents of the web credentials store can be viewed within the Credential Manager if the account password is entered, but a user cannot view the contents of Windows credentials natively. Web credentials are also stored by Internet Explorer when the user has opted to save a set of credentials for a particular website. Apps may store their own credentials as either web or Windows credentials if they wish to utilise the Credential Manager.

To access the contents of the web credentials (without access to the account password) and Windows credentials directories in plaintext, the Credential Manager dynamic-link libraries must be used. These libraries include 'vaultcli.dll', 'vault.dll' and 'advapi32.dll', which can be utilised via API calls in order to read both Web and Windows credentials in plaintext. This was the technique utilised by the study to obtain the stored credentials. However, the user needs to be logged into the device being analysed; practitioners attempting to recover credentials from an offline forensic image will need to use other approaches.

The Windows Credential Manager system should be further analysed to provide forensic practitioners with the knowledge required to extract credentials from offline images and/or systems where the user's password is unknown. Unfortunately, the study found the Credential Manager system to be somewhat lacking, particularly the sandboxing security improvements introduced with Windows 8.

Using the data obtained by the study, it was found that the majority of apps on the Windows platform also used of OAuth versions 1.0 and 2.0 (see Table 3.3).

Table 3.3: Locations and nature of authentication data on selected Windows apps

App	Stored using Credential Manager	Token obfuscation	Authentication system
Box	No	No	OAuth 2.0
Dropbox	Yes	Yes	OAuth 1.0
Facebook	Yes	No	Derivative of OAuth 2.0
OneDrive	Yes	Unknown	Appears to be WS-Security
OneNote	Yes	Unknown	Appears to be WS-Security
Skype	Yes	Unknown	WS-Security
Twitter	Yes	No	OAuth 1.0

3.3: The OAuth protocol

Having found that the majority of apps in the selection of popular online and cloud apps utilised the OAuth protocol, OAuth was chosen as a case study for further discussion, with the intention of providing the reader with a high-level overview of how a token-based authentication system operates, from the perspective of a forensic practitioner.

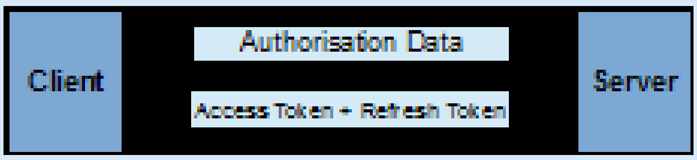
The OAuth protocol is a relatively recent specification. The final draft was released in 2007 and version 1.0 of the protocol was published in 2010 by the Internet Engineering Task Force (Hammer-Lahav 2010). Version 2.0 of the protocol was published in 2012; this second version of the standard was not backwards-compatible with OAuth 1.0 (Hardt 2012). The original OAuth protocol was therefore made obsolete by version 2.0, which has significantly simplified the operation of the token-based authentication system.

One of the most common uses of the OAuth protocol is to provide third-party apps with limited access to a user's resources (known as a 'scope'). A third-party app could, for example, request access to a user's Facebook photos. The user would be prompted to authenticate with Facebook's servers directly in order to allow or deny this request. If the user allows this request, Facebook's servers will return an access token allowing the third-party app limited access to the user's resources (in this case, only the user's photos). As this research focused exclusively on authentication for first-party apps, there was little need for an app's servers to return a resource-limited access token—all access tokens stored by these apps should be allowed to access all of a user's resources. The remainder of this discussion is based on this first-party token implementation, as explained further later in this chapter and in Figures 3.2–3.4.

3.3.1: Initial authorisation

In the initial stages of the OAuth protocol, a client (eg an app) transmits authorisation data to the service's servers. This authorisation data can contain a number of items, known as parameters, and commonly includes the user's username and password and, optionally, a scope (see Figure 3.2).

Figure 3.2: Initial authorisation in OAuth 2.0



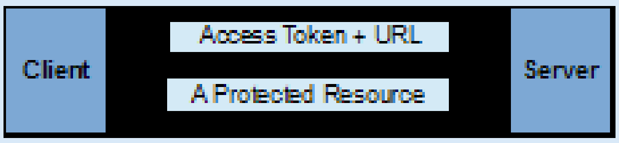
The scope parameter is used to specify the resources requested and, ultimately, which resources the access token can be used to access. Typically, where a username and password are presented with the initial authorisation data in OAuth 2.0, the access token returned by the server will provide full access to all of the user's resources—because the user's password is generally only provided to first-party client apps, as users may not trust third-party clients.

When using password-based authorisation in the OAuth 2.0 protocol, the parameters must be transmitted using Transport Layer Security (TLS). This is mandatory for secure operation, as the HTTP transmission contains the user's credentials in plaintext. After successfully authenticating with the server, the client receives an access token and a refresh token, generally in JavaScript Object Notation (JSON) format. The client may also receive an 'expires_in' parameter denoting the period of time the access token is valid for, along with any other parameters specified by the service provider's implementation.

3.3.2: Accessing a resource

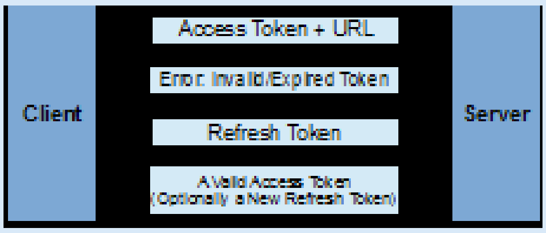
In order to access a protected resource (eg bank account details or private messages on a social networking website), the client is required to transmit the valid access token along with the requested resource's URL on the server. Typically, no further parameters are required in order to obtain the protected resource.

Figure 3.3: Accessing a protected resource with a valid access token in OAuth 2.0



If an access token is invalid (eg the access token has been invalidated or has expired), then the server will inform the client. The client then sends the refresh token obtained from the original password-authentication to the authentication server. If this refresh token is valid (as refresh tokens may have an expiry period), the authentication server then returns a new valid access token, and, optionally, a new valid refresh token. This access token can then be used in a new request to retrieve the protected resource.

Figure 3.4: Accessing a protected resource with an invalid access token in OAuth 2.0



However, if the service's implementation of the OAuth authentication standard does not utilise a refresh token, the user must re-send their authorisation data.

3.3.3: Generic OAuth authentication process

In summary, the generic process for authenticating a user to an app's online service via OAuth is as follows:

1. The client app prompts the user for their username and password credentials.
2. The app securely sends these credentials and any other necessary data such as the app key or unique ID to the authorisation URL for the service.
3. The authorisation service returns an access token (used to authorise individual resource requests) and a refresh token (used to request new access tokens). The app stores both of these tokens on the client device.
4. When the app needs to make an authorised request for a protected resource, it sends the stored access token as the authorisation. If the access token is valid, the protected resource data is returned and/or the requested operation is executed. Otherwise (i.e. the access token is invalid or expired), the app will use the stored refresh token with an authorisation URL to request a new access token. The new access token will then be used to request the protected resource, as described in this step.

The usefulness of this process for forensic practitioners seeking to collect evidence from online and cloud services is clear. The access and/or refresh tokens can be collected from a device and used by the practitioner to access user data, even when usernames and passwords are unknown. The use of access tokens can be embedded in forensic collection applications designed to obtain evidence from online and cloud services in a forensically sound manner.

Another potential advantage of access tokens is they may remain valid even after the user changes their password-based credentials, depending on the service provider's implementation. The access token may also be useful—again, depending on implementation—when access to the user's data is required without their knowledge; meanwhile, the user may continue using the service.

3.3.4: OAuth 1.0 vs OAuth 2.0

The primary difference between versions 1.0 and 2.0 of the OAuth protocol is that OAuth 2.0 relies entirely on HTTPS in order to secure its transmission to and from servers (Hardt 2012). On the other hand, OAuth 1.0 requires the use of cryptography on both the client and the server, in order to secure all authorisation of and access to protected resources (Hammer-Lahav 2010). Both versions of the OAuth protocol require the app's unique secret (as assigned by the authentication provider) upon initial authentication with the user's username and password. However, OAuth 1.0 requires that the app's secret be transmitted with each protected resource request (along with the access token), whereas OAuth 2.0 only requires the access token. In addition, as it does not rely on HTTPS for secure transmission, OAuth 1.0 requires the transmission of a nonce (a randomly selected value), the signature method and version numbers during initial authentication.

Another significant difference between the two versions of the protocol is that OAuth 1.0 does not utilise refresh tokens. Any access tokens generated are likely to be valid for an extended period of time—potentially indefinitely. The use of refresh tokens in OAuth 2.0 allows for scope-limited tokens with short lifespans and for new tokens to be obtained without user involvement.

3.4: Concluding remarks

To keep pace with the growth and changing face of criminal activity, it is essential that evidential material can be identified and preserved, regardless of whether it is held domestically or overseas.

A number of governments have sought to enhance their technical capabilities (and, in some instances, to circumvent or weaken existing security measures) and introduced legislation allowing national security and law enforcement agencies to conduct online surveillance.

As demonstrated in the analysis of popular cloud apps presented in this paper, forensic practitioners must adapt to the use of tokens as the most common means of stored authentication. Practitioners who have seized and analysed client devices such as laptops and smartphones will often be in a position to locate authentication tokens as part of this process.

Forensic practitioners will have to adapt their procedures for examining electronic evidence to ensure they obtain these authentication tokens where available. They will also need to maintain a working knowledge of the authentication token system if they are to be able to exploit the tokens to collect further evidence stored on remote systems.

One major challenge is the time-consuming process of establishing whether tokens exist for particular products and platforms, along with their type and usage. Researchers working in this field can assist practitioners by thoroughly analysing popular software products and platforms to guide practitioners. With this information, cloud-hosted data—an evidence source of significant and growing importance—will be accessible to forensic practitioners in most cases, and evidence hosted in the online environment will, at least technically, be available for presentation to court.

However, it is unclear whether existing Australian law permits the real-time use of remote evidence preservation and collection processes, or the use of tools to preserve evidence stored or held overseas, without a mutual assistance request.

If this direct-collection approach is deemed infeasible, Australian law enforcement agencies may also have access to alternative evidence collection approaches. For example, agencies can issue a domestic preservation notice to a carrier requiring the carrier to preserve all stored communications they hold which relate to the person or telecommunications service specified in the notice issued under section 107G of the *Telecommunications (Interception and Access) Act 1979* (Cth). Domestic preservation notices cover any stored communications that might relate to either the contravention of certain Australian laws or security; see Division 2 of the *Telecommunications (Interception and Access) Act 1979* (Cth).

Where evidence is determined to be hosted or stored outside Australia, Australian law enforcement agencies make a mutual assistance request via the federal Attorney-General to obtain the evidence in a form suitable for admission in Australian court proceedings under the *Foreign Evidence Act 1994* (Cth); see the Australian Government Attorney-General's Department (2013). Mutual assistance is a reciprocal process. Where a foreign nation has requested the preservation of evidence held in Australia, only the Australian Federal Police can serve the foreign preservation notice on an Australian carrier, and only if the foreign country requested the preservation in accordance with section 107P of the *Telecommunications (Interception and Access) Act 1979* (Cth). Foreign preservation notices cover stored communications that might relate to the contravention of certain foreign laws; see Division 3 of the *Telecommunications (Interception and Access) Act 1979* (Cth). Access to the information is then regulated by the *Mutual Assistance in Criminal Matters Act 1987* (Cth) which, in principle, affords sufficient safeguards to ensure access is consistent with Australian values.

The question of whether accessing evidence stored or held remotely (eg in overseas cloud storage accounts) could result in the violation of a foreign law is a grey area. It is therefore important that digital forensic researchers collaborate with legal and policy scholars and practitioners to ensure the legal effectiveness of any remote evidence preservation and collection processes.

From a technical perspective, work on the remote collection of evidence should continue. Future work should include an in-depth analysis of some of the less prominent authentication systems in use, and a discussion of the most appropriate ways of exploiting their functionality for the purposes of digital forensics.

Chapter 4: Platform as a service forensics

Ben Martini, Quang Do & Kim-Kwang Raymond Choo

Platform as a Service (PaaS) services have historically had less customer uptake in comparison to the other two cloud service models defined by the NIST. However, the use and popularity of PaaS services is now starting to rise, with many major technology/cloud vendors supporting PaaS services (Natis 2015a). As PaaS services begin to enter the mainstream, they must be studied from a forensic perspective to ensure that law enforcement agencies (LEAs) are in a position to investigate these services, as is the case with any emerging technology.

Recent trends show continued customer growth for this model. As the PaaS platform approaches maturity and we see greater customer uptake (Natis 2015a), we will in turn see a greater involvement of PaaS services in forensic investigations. Also, from a digital forensics research perspective, when considering its place as a service model between the infrastructure and software, PaaS has the potential to provide us with the opportunity to research and describe forensic artefacts that could be of interest across multiple cloud service models.

One category of such artefacts, integral in forensic investigations of PaaS along with other cloud service models, is authentication artefacts. Chapter 3 outlined the growing use and importance of authentication artefacts to successful cloud computing investigations, focusing on SaaS and mobile applications. This chapter aims to further explore authentication tokens from a digital forensic perspective using a PaaS cloud product as a case study.

4.1: PaaS forensics

PaaS forensics has many commonalities with other types of cloud forensics. One of these is the decision, which must be made by forensic practitioners when commencing PaaS forensics, as to whether to undertake a remote evidence collection process, or whether to seize and (at least partially) acquire the physical devices (servers) providing the service(s). It is likely that in the highly volatile environment of cloud computing it will be impractical for forensic practitioners to undertake a collection of all data stored in the cloud (Martini and Choo 2012). Instead, techniques targeted on the individual and/or dataset of interest, either remotely or locally, are far more likely to be successful and cost and/or time effective, as has been demonstrated in previous cloud forensics research (Martini and Choo 2014c). With this in mind, authentication and access control become very important factors for forensic practitioners, as they form common filter criteria when targeting a cloud platform. In other words, targeting all objects stored or running on the cloud platform that were created, accessed, modified or deleted by a particular user, perhaps within a particular time frame, would be a common and sensible means of targeting a forensic collection operation.

In the case of PaaS, logical collection (ie the collection of evidential artefacts via request to the cloud software, rather than the physical collection of raw data from storage and/or memory) would likely be the most feasible means of collection, due to the scale of cloud environments and the complex technologies used to store data in these environments. This is in contrast to traditional disk based and even mobile based collections. If a logical collection is to be undertaken, it is most likely the case that the forensic practitioner will need to use some form of authentication credentials to undertake the collection, as discussed in the previous chapter.

Commonly this is one of two types of credentials: either cloud administrator credentials, which will allow for the collection of all data exposed via the cloud service APIs, or user/organisation administrator credentials, which should allow access to most of the available data for the specified user/organisation. In cases where CSPs are providing cooperation, obtaining the former type of credentials may be feasible. However, in many cases CSPs are outside of the jurisdiction of Australian LEAs. This means that legislative measures such as mutual assistance requests must be relied upon in order to receive CSP assistance (Hooper, Martini and Choo 2013). In cases where timely evidence retrieval is integral, or where mutual assistance requests are otherwise impractical, CSPs may need to undertake more direct collection approaches, such as connecting to the cloud service directly using the user's credentials and extracting data in a forensically sound manner, where this is legally permissible.

Successfully achieving much of this is reliant upon the forensic practitioner having a sound knowledge of the various authentication artefacts available, how these artefacts operate and how they can be effectively exploited to facilitate collection. This chapter will explore these topics in greater detail, using the popular PaaS cloud product Cloud Foundry (CF) as a case study.

The next section outlines the experiment environment used to conduct the study outlined in this chapter and provides a brief outline of the CF components that are relevant to this study. The Findings section will outline the key learnings for forensic practitioners from studying the authentication components of CF. Finally, the *Conclusion* section will outline potential areas for future work.

4.2: Experimental environment

We selected Cloud Foundry as the case study platform for this research due to its popularity and its support for modern authentication. It is noted as one of the major PaaS frameworks in Gartner reports (Natis 2015b), and was developed by Pivotal, a company launched by big names in the industry including VMware and EMC (Jackson 2013). Pivotal boasts customers including Comcast, Verizon, Lockheed Martin, Garmin (Kearns 2015) and more recently Ford (Dignan 2015). CF is also one of the few examples of commercial cloud products which provide open source access to the key components of the product, which helps to facilitate forensics research.

Cloud Foundry can be provisioned in a number of different configurations. As this research is concerned with the authentication component of CF, we elected to deploy CF via BOSH Lite. BOSH Lite is designed to support the development of BOSH (the software deployment system used by CF) and CF (Cloud Foundry Foundation 2015a). A BOSH Lite deployment allows for the installation of CF on a single development workstation, rather than deploying CF to a cloud infrastructure, as would be the case in a production environment. At a high level this results in the deployment of a single virtual machine on the development workstation, which utilises a form of software isolation to separate the various services utilised to support a CF environment.

BOSH Lite also emulates a typical cloud networking environment, using the bosh-lite.com domain. All requests on the development workstation for *.bosh-lite.com resolve to the IP address of the “HAProxy” (high availability proxy server); however, in production environments this would be substituted for the environment's high availability load balancing solution (Pivotal Software 2015). The requests are then forwarded to the CF router component, which is responsible for forwarding the request to the appropriate CF component (eg in the case of api.bosh-lite.com) or CF hosted cloud app (eg myapp.bosh-lite.com). In this chapter, URLs are specified using the bosh-lite.com examples, but in a production environment they would need to be substituted, as appropriate, with the production domain name(s).

A full description of the various components used to support CF's PaaS services is provided in the CF documentation (Cloud Foundry Foundation 2015b). At the time of writing, there are more than 14 distinct components in the CF architecture. As such, it is impractical to analyse all of the components from a forensic perspective simultaneously. Considering the complex nature of this system, logical evidence acquisition may be the most practical method for collecting evidence from a CF environment. This is supported by the fact that, as with most cloud environments, a CF environment likely holds a proportion of its evidence of interest in volatile memory, complicating physical collection and analysis.

With these factors—and the likely necessity for logical collection—in mind, this research focuses on the authentication, and specifically the OAuth2 Server (known as UAA), component of CF. An understanding of the operation of the CF OAuth and authentication systems will assist practitioners in gaining and utilising credentials for connection to and extraction of logical evidence from a CF installation, either locally or remotely. Although CF is the case study platform in this research, it is hoped that some of the authentication research presented in this chapter will also be applicable to other similar cloud systems.

Supported by these components is CF's logical architecture, much of which is also specified in some detail by CF (Cloud Foundry Foundation 2015c). This logical architecture again consists of a range of components. However, the main components of interest to us are organisations, spaces, apps and users. One of the higher level organisational units within the CF logical architecture is the organisation or org. Orgs represent entities in CF, be they individuals or, as the name suggests, entire organisations. Some logging is completed on an organisational basis, for example, billing, which makes this component of interest to us from a forensic perspective. Spaces are a sub-unit of organisations, and can be used to separate projects within an organisation. For example, a space could be created for the CRM software within our organisation, and the multiple apps that are associated with the CRM software would co-exist within this space.

Apps are the applications deployed by users within spaces. These apps are supported by 'buildpacks', which are essentially scripted build environments that include the basic applications required to run the user's app. There are a number of built-in buildpacks for major development languages (eg Python, Java, PHP), and others that serve specific purposes (eg for static websites and pre-compiled binary applications). CF installations can also utilise third party buildpacks and develop their own. App deployment is a simple affair, and, while somewhat buildpack-specific, can be as simple as issuing the command cf push from the root directory of the app's source tree.

Users transcend these organisational units and, while they may often be associated with only a single organisation, they can just as easily be the manager of all of the organisations on the CF installation. Both orgs and spaces have three main roles (Cloud Foundry Foundation 2015c). Both have manager and auditor roles (eg Org Manager, Space Auditor). Manager privileges provide administrative control over the org/space (e.g changing users or their roles, creating/deleting spaces). Auditors have read-only access to relevant information in their org or space. Orgs also have a billing manager role which allows users to manage billing and payment information, and spaces have a space developer role, which allows for the development and management of apps within that space. The latter role is the most likely to be deployed for the average user, whereas admins of the org, space or entire CF installation are likely to have manager privileges as well. These roles are stored in a relatively simple manner within the cloud controller's database, as discussed in greater detail in the next section.

4.3: Findings

Cloud Foundry utilises tokens based on the OAuth2 standard for authenticating user requests after the user's initial (username and password-based) login. Most CF operations are conducted via a REST (REpresentational State Transfer) API, and an access authentication token is sent with each request as a bearer scheme HTTP(S) Authorization header in a HTTP(S) GET/POST/etc. request. This is a fairly standard REST/OAuth implementation. However, for obvious usability reasons, CF provide applications (which utilise this API) for users to access and manage their CF installation. The two applications that are of particular interest to us for the purposes of this research are cf (described as "[a] command line tool to interact with Cloud Foundry") and uaac (described as the "UAA Command Line Interface"). The former application allows users to configure and manage the CF installation generally, including components such as apps, organisations, spaces, services, buildpacks and limited user operations. The latter application permits the administration of users, groups and clients in the OAuth context. It also provides a range of token debugging capabilities, including a decode function for CF OAuth tokens.

The existence of either of these CF user apps on a seized device or forensic image may be not only a valuable evidence source but also an integral artefact, should a forensic practitioner wish to connect to a CF installation as the user. By default, the cf app stores its configuration in a hidden subdirectory of the user's home directory at ~/.cf/config.json. As the file extension implies, the configuration file is a JSON format and stores a range of information including the targeted CF instance, the default targeted organisation and space for the user. Perhaps most importantly, this file also contains the user's OAuth access and refresh tokens. Similarly, uaac stores a plaintext configuration file at ~/uaac.yml which contains the last access token obtained via the app, along with a decoded version of some of its parameters (eg the user's unique ID and their scopes). In addition, if the server's token signing key (see the CF Authentication Tokens section below) has been retrieved (via the command uaac signing key) this will also be included in the configuration file.

If a practitioner is successful in obtaining these tokens, it is important that they are able to understand the utility and the limitations of the tokens, so that they can be used effectively to retrieve evidence, and also to avoid arousing suspicion in certain cases where the practitioner's access to the token is not known to the suspect. The following subsections explain the CF OAuth authentication and token system in greater detail, starting with the relevant databases in the CF system that store authentication data, followed by an explanation of the authentication token architecture, type and format used by CF.

4.3.1: Databases

One of the VMs deployed within the BOSH environment for Cloud Foundry is a PostgreSQL database server. In our installation, the database server hosted two databases of interest, ccdb and uaadb. As the names imply, the former stores data relating to the cloud controller and CF environment, while the latter stores data relating to the CF users and the UAA service. Both databases store data relating to users and/or roles, and it is for this reason that they are of interest to us from a forensic perspective. In our case, the Postgres server was accessible from our development machine at the IP address of the Postgres BOSH VM on port 5524. Credentials for this connection could be obtained from the CSP, or by analysing the configuration files on the relevant BOSH VM (depending on the database being analysed). Practitioners cannot assume that they will be able to establish a network connection with the database server in a production environment, and they will need to undertake further investigation of the network security mechanisms used by the CSP (eg VLAN separation and firewalls) to determine whether network access to the Postgres server is feasible. If the practitioner has credentials/network access to the BOSH Director (a probable case if the CSP is cooperative), another potential option would be to use the bosh ssh command to open an SSH shell connection with the Postgres server, from which the database could be collected and analysed on the practitioner's machine.

The UAA supports both internal (ie users are created and managed within the UAA using CF tools) and external (eg LDAP) user databases. We focused on the former implementation, although the latter is unlikely to have a significant forensic impact (other than a potential need to extract further data from the linked LDAP server to gain a complete understanding).

4.3.1.1: The uaadb database

Within the uaadb database, there are a few tables of potential forensic interest, including users, oauth_client_details, and groups/group_membership. The users table contains a range of user-specific information which may assist practitioners in linking the users in the CF environment with persons of interest. Values of potential interest in this table include the user's id (a Globally Unique Identifier [GUID] for the user); created, last modified and password last modified timestamps (stored in UTC); and the user's username, email address, phone number and name (some values are optional). A hashed version of the user's password is also stored in this table using the bcrypt hash algorithm. The user's id is used in numerous places in CF, particularly where users need to be uniquely identified, including importantly in its authentication tokens.

The oauth_client_details table stores data relating to the various OAuth clients created within the CF installation. OAuth requests often require client (eg app) credentials in addition to or instead of user credentials. More than 10 clients were registered by default in our installation, with most representing one or more components in the CF architecture (eg cloud controller, go router and login). Values of interest in this table include the client_id (a unique string value representing the client), the client secret (hashed using bcrypt similarly to the users' passwords), a last modified timestamp and the client's authorities (the permissions granted to the client when authenticated using its client_credentials). Two other fields of particular interest to OAuth that are stored in this table are authorised_grant_types (which identifies the types of credentials that can be presented in order to obtain a token using this client, eg client credentials, a user's username and password, a refresh token) and scope (a list of permissions that this client has the potential to obtain for a properly authenticated and authorised user, which could potentially be embedded in a requested token) (Cloud Foundry Github 2015a). Requested scopes will only be assigned to a user's token when the client and the user both possess the scope (Cloud Foundry Github 2015b).

In practice, we used two of the built-in OAuth clients, cf and admin. The cf client is used by the cf app along with a password grant type to present the user's username and password credentials and obtain access and refresh tokens. The cf client is also used by uaac by default, with an implicit grant type to obtain an access token on behalf of a user. The admin client has a range of administrative authorities (beyond those allocated to the default admin user) such as creating and modifying clients and administering the UAA. We utilised the admin client's credentials with uaac for these purposes. The cf client does not use a client secret and is therefore limited in its capabilities, including having no authorities. The admin client does have a secret, which the practitioner would need to obtain in order to request a client credentials token. In our case, we obtained this secret from the UAA configuration file on the UAA BOSH VM at /var/vcap/jobs/uaa/config/uaa.yml; however, there is no guarantee that client secrets will be pre-provisioned within this file in a production environment.

The groups and group_membership tables store the grant information for the users within the UAA. For each group in the groups table a GUID is once again used for the id field. A display name and created and last modified timestamps are also included for each record. Further information on some of the UAA-related grants is available from Cloud Foundry documentation (Cloud Foundry Github 2015a). The group_membership table correlates the groups (grants) with users via their GUIDs and also includes an added timestamp value, which may be of interest in some investigations.

4.3.1.2: The ccdb database

The ccdb database contains a wealth of data on the cloud environment, including its constituent apps, services, organisations and spaces. As this study is focusing on authentication, we will be discussing only the authentication components in detail. However, it is worth noting that forensic practitioners may find collection and analysis of this database to be valuable during a CF investigation. The aim is to provide practitioners with access to much of the data stored in this database via API logical acquisition, once a valid access token has been obtained.

As noted in the Experiment Environment section above, organisations and spaces both have roles (eg manager and auditor) and role memberships are stored within tables in the ccdb database. However, this database chooses to use an alternative set of user IDs in comparison to the uaadb database and, as the UAA is responsible for authentication, there must be mapping between these IDs. This mapping exists within the users table of the ccdb database. The users table has an id field (a simple incrementing integer), followed by the respective user's GUID, along with other basic metadata fields relating to the user in ccdb.

Elsewhere in ccdb a user's (integer) id is used to reference the user. For example, the organizations_managers, organizations_auditors, organizations_billing_managers, organizations_users, spaces_managers, spaces_auditors and spaces_developers tables all store two values for each record, the organisation or space id (an integer in the spaces or organizations table, respectively) and a user_id (the user's integer identifier). Using this information, practitioners can determine the role(s) held by a given user within all of the organisations and spaces in the CF environment. Interestingly, the organizations_users table above does not appear to relate to an explicit role; however, when a role is granted to a space within an organisation, the user who was granted a role within the space appears to be added to the relevant organisation's organizations_users table.

Regardless of whether a practitioner chooses to, or is able to, collect and analyse these databases, it is important that a practitioner has a basic knowledge of the data that is potentially available. Where it is impractical to collect this data directly, it may be possible to collect the data via another means, such as via the application's public API with the appropriate credentials. An understanding of these concepts is also important in order to understand the data being presented in the authentication tokens generated by this system, as discussed in the next section of this chapter.

4.3.2: Authentication tokens

4.3.2.1: Authentication token requests

As outlined in the Experiment Environment section of this chapter, most authenticated operations in the CF environment (ie via the CF API) require a bearer access token to be presented as an Authorization header in the request. Both the cf and uaac apps obtain and store these tokens after the first login by the user, however the uaac app only stores an access token by default whereas cf stores both an access token and a refresh token. As outlined in the previous chapter, a refresh token can be used to obtain a new access token once the access token expires, an operation that is conducted transparently by the cf app as required. Once the access token stored by the uaac app has expired, the user is presented with an "invalid_token" error message indicating that the token has expired. At this point, they will need to request a new token, likely via their username and password or app-secret credentials.

The UAA is designed to be extensible, and useful in a number of circumstances within a CF—and potentially any other—environment. As such, we will not seek to address every authentication capability of the UAA, but rather focus on those that would be commonly used and of interest to forensic practitioners, based upon our case study implementation. Equally, this section is designed to be understood by experts in digital forensics, and as such our focus is on explaining the practicalities of the implementation, rather than reiterating the UAA, JWT and OAuth2 specifications verbatim (although it is recommended that practitioners refer to these specifications during their research).

With this in mind, there are two key entities that can authenticate to a CF instance, users and clients. Clients may also have the ability to obtain tokens on behalf of users, if they have the appropriate credentials and permissions. A user-based access (and refresh) token can be obtained using a client holding the password grant by sending a HTTP POST request to a URL similar to the following: https://uaa.bosh-lite.com/oauth/token?grant_type=password& response_type=token&client_id=client&username=user&password=pass. A valid client ID and user credentials must be specified, and the client ID and secret must be sent in a HTTP Basic Authorization header.

In our case, the returned access token is only valid for 600 seconds, or 10 minutes (this is imposed by the configuration of the cf client in the uaadb, whereas the default access token expiry time is 43,200 seconds, or 12 hours, as specified in the UAA configuration file); however, the refresh token is valid for 2,592,000 seconds, or 30 days. Once the access token expires, the refresh token can be used via a POST to the following URL:

https://uaa.bosh-lite.com/oauth/token?grant_type=refresh_token&refresh_token=[refresh token] – substituting [refresh token] for the actual refresh token returned from the previous request. This request will return a new access token, and potentially a new refresh token, if necessary.

A user-based access token can also be obtained by a client with the implicit grant, however this will not return a refresh token. Similarly to a password grant, a POST should be sent to https://uaa.bosh-lite.com/oauth/authorize?client_id=client &response_type=token&redirect_uri=http%3A%2F%2Flocalhost.localdomain. The response will be a 302 Found redirect to the request URL followed by the access token and metadata in www-form-urlencoded format. The access token can be parsed from the 302 response, or a script can potentially collect the access token at the redirected URL.

The client IDs and secrets for the above requests will need to be valid. However, in our implementation the built-in cf client, which does not have a client secret, was successful in obtaining tokens.

A token with the authorities of a client can be obtained if the client has the client_credentials grant. In our implementation, we found that the admin client had many administrative authorities, as discussed in the previous section. Obtaining an access token using a client credential grant is straightforward: a POST request is sent to http://uaa.bosh-lite.com/oauth/token?grant_type=client_credentials, with the client ID and secret sent in a HTTP Basic Authorization header. An access token for the client is returned along with relevant metadata.

The UAA APIs that are used to obtain tokens—and many other APIs—are defined and discussed in further detail in the UAA API specification (Cloud Foundry Github 2015c).

4.3.2.2: Authentication token format

CF authentication tokens are based upon the JSON Web Token (JWT) authentication token format (Cloud Foundry Foundation 2015d). The JWT specification is published in RFC 7519 (Jones, Bradley and Sakimura 2015) and practitioners working with JWT tokens are encouraged to familiarise themselves with the specification and operation of these tokens. This section will provide a brief overview of their implementation within CF.

A CF authentication token consists of three distinct parts: a header, a payload and a signature (Jones et al. 2015). In the token's stored and transmitted form, these three components are separately base64 encoded and then concatenated using periods as a delimiter. This encoding may give the practitioner the impression that the token is a pseudorandom string; however, this is not the case. Each section of the token can be base64 decoded without any additional information, although only the header and payload sections will decode to plaintext.

Decoding the header component of the token will return the value {"alg":"RS256"}, which denotes that the token signature was generated using an asymmetric RSA algorithm with SHA-256 hashing (Jones et al. 2015). CF may also support HMAC SHA-256 symmetric key signatures, as is required by the standard (Jones et al. 2015). However, this algorithm was not enabled by default on our installation (RS256 was instead) and it seems unlikely that it would be enabled in favour of RSA on production systems. The optional JWT typ header does not appear to be included in the CF implementation.

The payload section of the token is the component that is most likely to be of interest to a forensic practitioner. Within the payload is a JSON object that contains a number of 'claims' and associated values (Jones et al. 2015). They can be considered similar in operation to a key value pair. These claims work on the principle that, after receiving a valid authentication request (eg using a password, refresh token or client credentials), the authentication service will issue a signed token indicating the identity and capabilities of the user (ie the claims). This token is then presented to resource servers, which then need to validate the token (Cloud Foundry Github 2015b) to ensure that it originated from the legitimate authentication server (eg by checking the token signature, expiry timestamps, and revocation signature). Once validated, the resource server can make a determination as to whether the presented token has sufficient authority to complete the requested task (based on the one or more claims presented).

Table 4.1 outlines some of the common claims that we found in four types of authentication tokens issued by the CF UAA (implicit and password grant access tokens, a refresh token and a client credentials access token). It also outlines the types of token that featured the claim, a brief description of the claim, and an example. Some of the claims are 'registered claim names' outlined in the specification, and where that is the case their description is based on the definition provided in the specification or IANA documentation (IANA 2015, Jones et al. 2015).

Table 4.1: CF authentication token payloads

Claim	Type	Description	Example
jti	All	The JWT ID is a unique identifier for the token[1]. UAA utilises a GUID for the jti. In the case of refresh tokens the jti is a GUID with "-r" appended.	aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee-r
nonce	Implicit access	The nonce value that was used as part of the implicit token request. This claim is not present if a nonce was not used.	0123456789abcdef0123456789abcdef
sub	All	The subject identifies the principal (user or client) for which the token was issued1. In the case of a user, their GUID from the users table in the uaadb is used for this value. In the case of a client, the client ID is used.	aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee (user) clientid (client)
user_id	All except for client credentials access	The user's GUID ID, as in sub.	aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee
authorities	Client credentials access	Authorities are scopes (permissions) that a client is permitted to use when authenticated using its client credentials[2].	clients.read password.write scim.write
scope	All	The scope (permissions) granted to a particular token, based on those granted by the admin, and potentially a subset requested by the user. In the case of user credentials, this is limited to the permitted scopes shared between the client acting on behalf of the user and the user's assigned scopes. Both of these values are defined in uaadb. In the case of client credentials tokens, the scope will generally match the authorities[2].	cloud_controller.read password.write cloud_controller.write openid
client_id cid	All	The client ID of the client that processed the user's request (in the case of password or implicit grants) or the client's ID (in the case of client credentials grants).	clientid
azp	All access	The authorised party 'to which the token was issued'[3]. The client ID in all relevant UAA tokens.	clientid
grant_type	All except for implicit access	The grant type used to issue the (initial) token.	password client_credentials
origin	All except for client credentials access	Presumably the authentication source for the user's credentials. Only uaa was used in our environment.	uaa
user_name	All except for client credentials access	The user's username.	admin

Table 4.1: CF authentication token payloads cont.

Claim	Type	Description	Example
email	All access	The user's registered email address.	admin@bosh-lite.com
auth_time	All access	The time that authentication occurred[3]. Generally matches the iat, and is represented as a UTC POSIX timestamp.	1451606400
iat	All	The issued at time for the token. Generally matches the auth_time (when present), and is represented as a UTC POSIX timestamp.	
exp	All	The time at which the token will expire. Represented as a UTC POSIX timestamp.	1451607000
rev_sig	All	The revocation signature is a 32-bit hash value, computed using 'MurmurHash3', based on inputs including the client ID and secret, and user's ID, hashed password, email and username[4].	1234abcd
iss	All	The token issuer1. The UAA token URL in the CF UAA implementation.	https://uaa.bosh-lite.com/oauth/token
zid	All	Identity zone ID[5]. Always uaa in our examples.	uaa
aud	All	The audience for the token. Consists of the 'base name' of all token scopes[6].	cloud_controller password openid

1 Jones et al. (2015)

2 Cloud Foundry Github (2015a)

3 IANA (2015)

4 Cloud Foundry Github (2015d)

5 Cloud Foundry Github (2015c)

6 Cloud Foundry Github (2015b)

Table 4.1 demonstrates that the payload of a UAA access token is quite straightforward, and contains a range of values that could be of potential interest to forensic practitioners, depending on the case. Values such as the iss can be used to determine which cloud or clouds a suspect has been connecting to. The username and email address of the user is included in the token, along with timestamps indicating quite accurately when the service was last used. Permissions values such as scopes and audience may help to identify the level of access obtained by a suspect to the cloud environment, and may assist the practitioner to focus their collection and analysis on components of the cloud that were accessed or modified by the suspect.

One value contained within the token that has a less obvious use is the rev_sig or revocation signature (Cloud Foundry Github 2015e). Although it seems innocuous, it is in fact a security measure which gives the user the capability to revoke previously issued tokens, and inherently limits a forensic practitioner's capability to covertly use the user's tokens for the purpose of forensic collection. This value is based on a 32-bit hash value comprised of a number of inputs, including the username and password of the user, and the ID and secret of the client (Cloud Foundry Github 2015d). If the user changes their password, their revocation signature will implicitly change and, as this is checked during token validation, all of their previous tokens will be invalidated. Fortunately, if the user doesn't change any of the hash inputs, the revision signature remains relatively stable. However, as the revision signature is based on values stored in the uaadb, particularly the bcrypt hashed user password, offline generation of the revocation signature is impractical without access to a dump of the uaadb.

The token signature is the final component of the CF authentication token. It is used to verify the authenticity of the token by checking the signature using the public token verification key. The public token verification key can be obtained without authentication via a GET request to https://uaa.bosh-lite.com/token_key. Any attempt to modify the payload of the token without updating the signature will result in an invalid token response when used, with the error description indicating that the token could not be decoded (in fact, the underlying exception indicates that the signature did not match the content, but this is not returned to the client).

McLean (2015) highlighted some interesting vulnerabilities in the signature checking process for some JWT libraries. As noted above, the header component of the token stores the algorithm used to derive the signature component. McLean (2015) notes that some JWT libraries trust this value when checking the signature, which means that an alg value of none may result in no signature verification. He also notes that if a HMAC alg is provided in the header, and a HMAC signature is derived using the server's public key as the input, the signature would successfully validate, as the HMAC verification key on the server would be the public key. Our testing showed that CF does not appear to be vulnerable to these exploits, likely due to the fact that the signature verification algorithm is determined based on the key stored on the server, rather than on the header value in the token. Regardless, if practitioners need to generate tokens offline, vulnerabilities such as these may simplify the process, should the practitioner possess the relevant legal authority to do so.

We obtained a copy of the UAA server's private key from its configuration file on the UAA BOSH VM at /var/vcap/jobs/uaa/config/uaa.yml. Using this key, we attempted to re-sign modified tokens or create new tokens, to see if they would be accepted by the server. We found that a correctly signed token was accepted by all of the interfaces that we tried via both the CF API and that UAA API. Successful modifications included extending the expiry time of a token arbitrarily, and increasing the scopes within a token (ie giving a standard user the scopes of an admin user). Our testing was not fully comprehensive; however, we were successful in escalating our privileges so that an unprivileged user could undertake administrative operations.

From a forensic perspective, this could allow a practitioner to use an unprivileged user to collect evidence from parts of the system which they could not ordinarily access. The ability to arbitrarily extend the exp value may also allow a practitioner to utilise an expired access token recovered from a suspect device to collect evidence from a live CF instance. Modifying tokens may also have the potential to enhance the forensic soundness of the collection process. For example, depending on the type of collection operation/APIs used, it may also be practical to modify a token to limit its scope to only read operations (eg cloud_controller.read). Of course, at least in the case of CF, this will require the practitioner to obtain the server's private signing key, something that may or may not be feasible depending on the case and available resources.

If the practitioner is able to modify and re-sign tokens, the revision signature would generally present another barrier to their successful use, as the user could change their password, thereby changing the signature (as outlined above). However, we found that simply removing the revision signature claim from the token entirely (and re-signing) resulted in the token being accepted. This appears to be due to a backwards compatibility check in the token verification code (Cloud Foundry Github 2015d), but is a great advantage to a forensic practitioner looking to use user authentication tokens for forensic collection.

4.4: Concluding remarks

Cloud services continue to present a valuable evidence source for forensic practitioners. However, due to the variety of platforms available, and their often esoteric nature, forensic practitioners cannot always have a complete understanding of the operation of every cloud platform. This practical limitation means that evidential artefacts, or even entire evidence sources, may be missed. Forensic research, such as that presented in this chapter, which uses practical platforms as a case study, can serve to highlight the types of evidence that may be available, not only in the case study application itself but also in other similar applications. This can hopefully provide practitioners with an enhanced knowledge of the types of evidence that are available, along with potential means for their collection, without those practitioners having to personally analyse the application in question.

With this in mind, and based on the above findings, a suggested process for understanding and utilising cloud authentication tokens in forensic investigations is presented below.

1. **Identify the existence of authentication tokens:** In most cases, a seized client (computer, phone, etc.) will be the means of identifying a suspect's cloud service usage (Martini and Choo 2012). As such, practitioners should be seeking out cloud computing artefacts on client images and, when found, searching for modern credentials such as authentication tokens, in addition to traditional authentication and forensic artefacts. In the case of CF, this may include uaac.yml or CF config.json files.
2. **Determine the token type and format**: Once tokens are located, their type and format should be determined. In some cases, tokens may be based upon published standards, as was the case in this research (JWT standard). However, in other cases the token format may be custom or proprietary, which will mean that practitioners need to undertake further research/analysis of the source application to determine the token's structure and format.
3. **Decode the token:** Assuming that the practitioner is able to determine and map the token format, the next stage is to decode the token's constituent parts. In this research, this consisted of a header, payload and signature. However, each token format is likely to vary somewhat. Even tokens that follow the JWT standard, for example, can use their own private claims (as CF does). The practitioner will need to understand the role of each claim.

   This is necessary from a forensic point of view, so that the token can be reported on accurately as an evidential artefact. However, it may also be necessary if the practitioner seeks to use a token as an authentication artefact for the purposes of collection.
4. **Locate token flaws:** Tokens are generally designed to be a secure authentication scheme. In some cases it may be possible for practitioners to use tokens obtained from client devices to authenticate as the user and obtain evidence; however, in other cases security features will prevent this. For example, in this case study the revocation signature and token expiry times could both prevent effective use by a forensic practitioner. For this reason it is necessary to locate flaws in the token architecture, for example, in the authentication and/or access components of the token. This could include finding flaws in the validation process which allow for modified tokens, or tokens that are created by the practitioner offline, to be presented as valid.

In addition to the findings related to tokens, we also gained a better understanding of the operation of the authentication system of a mainstream PaaS cloud product. The UAA authentication system present in CF is fully featured, and provided a good case study environment to convey the complexities of an OAuth installation. This includes, for example, the relationship between groups in the authentication database, client scopes and the scopes that were ultimately included in the user's token. We also looked at the importance of OAuth clients, and their two practical roles in CF: acting on their own behalf using client credentials, and obtaining tokens on a user's behalf using password-based or refresh token grants.

Chapter 5: Conclusion

Ben Martini and Kim-Kwang Raymond Choo

It is well known that criminally motivated individuals will take every available opportunity to avoid prosecution. This applies equally in the online space, where criminals will make use of the latest technologies to enhance their operations and to evade law enforcement. It is the role of digital forensics practitioners to keep up with the changes in this incredibly fast-paced environment, in order to ensure that evidence sources are collected, analysed and presented in a complete and effective manner. Researchers in the digital forensics space can assist by conducting in-depth analysis of emerging technologies and by publishing their findings to assist forensic practitioners.

This report has sought to highlight and provide a better understanding of some of the key contemporary areas impacting on successful cloud forensic investigations. Chapter 2, Literature Review, highlighted that there remain many challenges in the area of cloud forensics. These include both technical and legal impediments to successful forensic investigations involving cloud computing. A number of the issues highlighted—for example, trust regarding evidence integrity when a CSP needs to be involved, the inherent volatility of cloud data escalating the urgency of collection, and the lack of physical accessibility and control—lead us to consider remote forensic collection of cloud evidence. Remote collection mitigates a number of these issues, as the practitioner is in control of the process. However, one significant impediment to successful remote collection is access control. Naturally, cloud resources are protected, and while in some cases CSPs may be cooperative, this is not always the case. In fact, suspects may specifically select CSPs in jurisdictions without mutual assistance agreements and where they know that the CSP will be uncooperative with LEA requests. In these cases practitioners need the capability to collect evidence remotely, where it is legally permissible to do so, and this will often require bypassing these access control restrictions.

Authentication in the cloud computing and contemporary online environments and the emerging use of authentication tokens—in place of traditional username and password-based authentication—have been understudied from a digital forensics perspective. Yet if practitioners are to maintain an effective cloud forensics capability, these areas must be well understood. This has been the focus of this report's findings.

In Chapter 3, the scope of the issue was assessed with an analysis of several popular cloud apps on both Windows and Android platforms. It was found that many popular cloud and online apps stored user credentials in an authentication token format. This helps to prevent the theft or misuse of the user's username and password credentials, and can also be used to limit the capabilities of an app to a subset of the user's total capabilities. The most popular of these formats was found to be the OAuth2 standard, although some of the stored tokens were further obfuscated or protected by operating system features.

The OAuth2 standard was briefly outlined in Chapter 3. It involves the concept of an access token and a refresh token. The user's credentials are initially used to obtain an access token and a refresh token from an OAuth2 authentication server. The tokens are then stored, with the access token having a relatively short life and the refresh token having a longer life before expiry. The access token can be presented to resource servers to obtain access to resources on behalf of the user for which the token was issued. Once the access token expires, the refresh token can be sent to the authentication server in order to obtain another access token.

In addition to these technical findings, Chapter 3 also discussed the issue of legal reform in the area of remote evidence collection. For example, the question of whether accessing evidence stored or held remotely could result in the violation of a foreign law is a grey area. It is therefore important that digital forensic researchers collaborate with legal and policy scholars and practitioners to ensure the legal effectiveness of any remote evidence preservation and collection processes.

In Chapter 4, an in-depth case study of authentication tokens in the cloud environment was conducted using the PaaS platform Cloud Foundry (CF) as a case study. The authentication component of CF was analysed in some detail, focusing on the user and OAuth client data stored by CF, and the type, format and utility of the tokens issued by CF. It was found that the tokens issued by CF were based on the published JSON Web Token (JWT) format. These JWT tokens consist of three major components: a header, a payload and a signature. The payload is base64 encoded and, once decoded, can provide forensic practitioners with a significant amount of potentially evidential information. This includes the authentication service the token belongs to (which will likely point to a particular cloud instance), the username and email address for the token, its issued time and the user's scope (permissions) within the cloud environment. Techniques for exploiting this authentication token implementation for the purpose of digital forensic collection were also discussed, including re-signing tokens with changed attributes (eg extended expiries and escalated privileges). However, as would be expected, security measures somewhat limited the available capabilities in this area.

Future work should continue to analyse the various authentication tokens available and their implementation within the cloud computing environment. This will provide practitioners with a more complete understanding of the authentication artefacts that may be available and of their value in a forensic investigation. Research should also continue on the PaaS model; for example, including a discussion of the types of PaaS evidence that is available, and the best practice for its collection and analysis.

We hope that the findings in this report will be valuable to the digital forensic community in better understanding the types of contemporary authentication artefacts available and how they can best be utilised in a forensic investigation.

References

All URLs correct at December 2015

Abbadi IM 2011. Operational trust in Clouds' environment. *IEEE Symposium on Computers and Communications (ISCC) 2* IEEE Computer Society: June 28 - July 1 2011: 141–145

Abbadi IM & Alawneh M 2012. A framework for establishing trust in the Cloud. *Computers & Electrical Engineering* 38: 1073–1087

Ante SE 2011. Some Top Apps Put Data at Risk. *Wall Street Journal* 8 June. http://blogs.wsj.com/digits/2011/06/08/some-top-apps-put-data-at-risk/

Attorney-General's Department 2013. *Fact sheet—Mutual assistance overview*. https://www.ag.gov.au/Internationalrelations/Internationalcrimecooperationarrangements/MutualAssistance/Pages/default.aspx

Biggs S & Vidalis S 2009. Cloud Computing: The impact on digital forensic investigations. *International Conference for Internet Technology and Secured Transactions* 2009. ICITST 2009: 9-12 Nov. 1–6

Birk D 2011. Technical Challenges of Forensic Investigations in Cloud Computing Environments. *Workshop on Cryptography and Security in Clouds*. Zurich, Switzerland

Birk D & Wegener C 2011. Technical Issues of Forensic Investigations in Cloud Computing Environments. *2011 Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering:* 1–10

Cloud Foundry Foundation 2015a. *Run a Local Cloud Foundry Instance*. http://docs.cloudfoundry.org/deploying/run-local.html

Cloud Foundry Foundation 2015b. *Cloud Foundry Components*. http://docs.cloudfoundry.org/concepts/architecture/

Cloud Foundry Foundation 2015c. *Orgs, Spaces, Roles, and Permissions.* http://docs.cloudfoundry.org/concepts/roles.html

Cloud Foundry Foundation 2015d. *Understanding Cloud Foundry Security.* http://docs.cloudfoundry.org/concepts/security.html#auth

Cloud Foundry Github 2015a. *UAA Security Features and Configuration*. https://github.com/cloudfoundry/uaa/blob/master/docs/UAA-Security.md

Cloud Foundry Github 2015b. *User Account and Authentication - A note on Tokens*. https://github.com/cloudfoundry/uaa/blob/master/docs/UAA-Tokens.md

Cloud Foundry Github 2015c. *User Account and Authentication Service APIs.* https://github.com/cloudfoundry/uaa/blob/master/docs/UAA-APIs.rst

Cloud Foundry Github 2015d. *uaa/UaaTokenServices.java.* https://github.com/cloudfoundry/uaa/blob/master/common/src/main/java/org/cloudfoundry/identity/uaa/oauth/token/UaaTokenServices.java

Cloud Foundry Github 2015e. *uaa/Claims.java*. https://github.com/cloudfoundry/uaa/blob/master/common/src/main/java/org/cloudfoundry/identity/uaa/oauth/Claims.java

Damshenas M, Dehghantanha A, Mahmoud R & Bin Shamsuddin S 2012. Forensics investigation challenges in cloud computing environments. *2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec 2012)*: 190–194

Daryabar D, Dehghantanha A, Eterovic-Soric B and Choo K-KR. Forensic Investigation of OneDrive, Box, GoogleDrive and Dropbox Applications on Android and iOS Devices. *Australian Journal of Forensic Sciences*, in press. DOI: http://dx.doi.org/10.1080/00450618.2015.1110620

Delport W, Olivier, MS & Köhn, M 2011. Isolating a cloud instance for a digital forensic investigation. *Proceedings of the Information and Computer Security Architecture (ICSA)*, IEEE: Johannesburg, South Africa: 25–33

Dignan L 2015. *Ford teams with Pivotal, bets on Cloud Foundry.* http://www.zdnet.com/article/ford-teams-with-pivotal-bets-on-cloud-foundry/

Dykstra, J & Sherman AT 2011. Understanding issues in cloud forensics: two hypothetical case studies. *Journal of Network Forensics* 2011b, 1–10

Dykstra J & Sherman AT 2012. Acquiring forensic evidence from infrastructure-as-a-service cloud computing: Exploring and evaluating tools, trust, and techniques. *Digital Investigation* 9: S90–S98

Dykstra J & Sherman AT 2013. Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform. *Digital Investigation* 10: S87–S95

Grispos G, Storer T & Glisson WB 2012. Calm Before the Storm: The Challenges of Cloud Computing in Digital Forensics. International *Journal of Digital Crime and Forensics (IJDCF)* 4: 28–48

Hammer-Lahav E 2010. *The OAuth 1.0 Protocol.* The Internet Engineering Task Force. http://tools.ietf.org/html/rfc5849

Hardt D 2012. *The OAuth 2.0 Authorization Framework*. The Internet Engineering Task Force. http://tools.ietf.org/html/rfc6749.

Hegarty R, Merabti M, Shi Q & Askwith B 2009. Forensic Analysis of Distributed Data in a Service Oriented Computing Platform. 1*0th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting (PGNet 2009)* 2 Liverpool, UK: 22–23 June

Hogben . & Royer J-C 2013. D12.1 *Project horizons report,* in Wainwright N, Papanikolaou N, Tountopoulos V, Oliveira ASD, Jaatun MG, Fischer-Hübner S, Pulls T, Hogben G, Catteddu D & Önen M (eds) Cloud Accountability Project (A4Cloud)

Hooper C, Martini B & Choo K-KR 2013. Cloud computing and its implications for cybercrime investigations in Australia. *Computer Law & Security Review* 29(2): 152–163

IANA 2015. *JSON Web Token (JWT).* http://www.iana.org/assignments/jwt/jwt.xhtml

Jackson J 2013. Pivotal launched from *VMware, EMC technologies.* http://www.pcworld.com/article/2036305/pivotal-launched-from-vmware-emc-technologies.html

Jones M, Bradley J & Sakimura N 2015. RFC 7519: *JSON Web Token (JWT)*. https://tools.ietf.org/html/rfc7519

Kearns A 2015. *Ten Pivotal Customers to Speak at Cloud Foundry Summit* May 11-12. https://blog.pivotal.io/pivotal-cloud-foundry/news/ten-pivotal-customers-to-speak-at-cloud-foundry-summit

Ko RKL, Jagadpramana P, Mowbray M, Pearson S, Kirchberg M, Qianhui L & Bu Sung L 2011. TrustCloud: A Framework for Accountability and Trust in Cloud Computing. 2011 IEEE World Congress on Services: 4-9 July. 584–588

Krautheim FJ, Phatak D & Sherman A 2010. Introducing the Trusted Virtual Environment Module: A New Mechanism for Rooting Trust in Cloud Computing, in: Acquisti, A, Smith S & Sadeghi A-R (eds), *Trust and Trustworthy Computing.* Berlin, Heidelberg: Springer

Li J, Chen X, Huang Q & Wong DS 2013. Digital provenance: Enabling secure data forensics in cloud computing. *Future Generation Computer Systems* 37: 259–266

Liu F, Tong J, Mao J, Bohn R, Messina J, Badger L & Leaf D 2011. *NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology.* National Institute of Standards and Technology Special Publication 500-292: Washington, DC

Lu R, Lin X, Liang X & Shen X 2010. Secure provenance: the essential of bread and butter of data forensics in cloud computing. *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security.* Beijing, China: ACM

Martini B & Choo K-KR 2013. Cloud storage forensics: ownCloud as a case study. *Digital Investigation* 10(4): 287–299

Martini B & Choo K-KR 2014a. Cloud forensic technical challenges and solutions: A snapshot. *IEEE Cloud Computing* 1(4): 20–25

Martini B & Choo K-KR 2014b. Remote Programmatic vCloud Forensics: A Six-Step Collection Process and a Proof of Concept, in *Proceedings of 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. Piscataway, NJ: IEEE Computer Society Press: 935–942

Martini B & Choo K-KR 2014c. Distributed filesystem forensics: XtreemFS as a case study. *Digital Investigation* 11(4): 295–313

Martini B, Do Q & Choo K-KR 2015. Conceptual evidence collection and analysis methodology for Android devices, in Ko R & Choo K-KR (eds), *Cloud Security Ecosystem*. Waltham, MA: Syngress: 285–307

Martini B, Quick D & Choo K-KR 2013. Cloud Computing and Digital Forensics, in Kim, I-S (ed), Information Society and Cybercrime: Challenges for Criminology and Criminal Justice, *Research Report Series* 13-B-01. Korean Institute of Criminology: 111–125

Marty R 2011. Cloud application logging for forensics. *Proceedings of the 2011 ACM Symposium on Applied Computing*. TaiChung, Taiwan: ACM

Mason S & George E 2011. Digital evidence and 'cloud' computing. *Computer Law & Security Review* 27: 524–528

McLean T 2015. *Critical vulnerabilities in JSON Web Token libraries.* https://auth0.com/blog/2015/03/31/critical-vulnerabilities-in-json-web-token-libraries/

Mell P & Grance T 2011. *The NIST definition of cloud computing.* http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Natis Y 2015a. *The Key Trends in PaaS 2015.* Gartner

Natis Y 2015b. *Gartner on the State of PaaS: Recent Research.* Gartner

Orton I, Alva A & Endicott-Popovsky B 2013. Legal Process and Requirements for Cloud Forensic Investigations. *Cybercrime and Cloud Forensics: Applications for Investigation Processes*. IGI Global.

Pivotal Software 2015. *Zero Downtime Deployment and Scaling in CF.* https://docs.pivotal.io/pivotalcf/concepts/high-availability.html

Poisel R, Malzer E & Tjoa S 2012. Evidence and Cloud Computing : The Virtual Machine Introspection Approach. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 4 (1): 135-152

Quick D, & Choo K-KR 2013a. Digital droplets: Microsoft SkyDrive forensic data remnants. *Future Generation Computer Systems* 29(6): 1378–1394

Quick D, & Choo K-KR 2013b. Dropbox analysis: Data remnants on user machines. *Digital Investigation* 10(1): 3–18

Quick D, & Choo K-KR 2013c. Forensic collection of cloud storage data: Does the act of collection result in changes to the data or its metadata? *Digital Investigation* 10(3): 266–277

Quick D, & Choo K-KR 2014. Google Drive: Forensic analysis of data remnants. *Journal of Network and Computer Applications* 40: 179–193

Quick D, Martini B & Choo K-KR 2014. *Cloud storage forensics.* Waltham, MA: Syngress

Ruan K, Carthy J, Kechadi T & Baggili I 2013. Cloud forensics definitions and critical criteria for cloud forensic capability: An overview of survey results. *Digital Investigation* 10: 34-43

Ruan K, Carthy J, Kechadi T & Crosbie M 2011. Cloud forensics: An overview, in: Peterson G. & Shenoi S (eds) *Advances in Digital Forensics* VII. Berlin, Heidelberg: Springer

Santos N, Gummadi KP & Rodrigues R 2009. Towards trusted cloud computing. *Proceedings of the 2009 conference on Hot topics in cloud computing.* San Diego, California: USENIX Association

Satyanarayanan M 1990. Scalable, secure, and highly available distributed file access. *IEEE Computer* 23(5): 9–18

Shariati M, Dehghantanha A and Choo K-KR 2016. SugarSync Forensic Analysis. *Australian Journal of Forensic Sciences,* in press, DOI: http://dx.doi.org/10.1080/00450618.2015.1021379

Shariati M, Dehghantanha A, Martini B and Choo K-KR 2015. Chapter 19: Ubuntu One Investigation: Detecting Evidences on Client Machines, in Ko R and Choo K-KR (eds), *Cloud Security Ecosystem.* Syngress: 429–446

Shi Y, Zhang K & Li Q 2010. A New Data Integrity Verification Mechanism for SaaS, in: Wang F, Gong Z, Luo X & Lei J (eds) *Web Information Systems and Mining. Berlin*, Heidelberg: Springer

Sibiya G, Venter HS & Fogwill T 2012. Digital Forensic Framework for a Cloud Environment, in Cunningham PCAM (ed), *IST-Africa 2012 Conference Proceedings*. 2 Tanzania: IIMC International Information Management Corporation

Simou S, Kalloniatis C, Kavakli E & Gritzalis S 2014. Cloud Forensics Solutions: A Review, in Iliadis L, Papazoglou M & Pohl K (eds), *Advanced Information Systems Engineering Workshops.* Springer International Publishing

Slusky L, Partow-Navid P & Doshi M 2012. Cloud computing and computer forensics for business applications. *Journal of Technology Research* 3: 1–10

Taylor M, Haggerty J, Gresty D & Hegarty R 2010. Digital evidence in cloud computing systems. *Computer Law & Security Review* 26: 304–308

Thethi N, & Keane A (2014). Digital forensics investigations in the Cloud. 2014 *IEEE International Advance Computing Conference* (IACC): 1475–1480

Song X, Chen L & Xiao M 2013. An Efficient Encryption and Verification Scheme for Preserving Electronic Evidence in Cloud Computing. *Journal of Information & Computational Science* 10: 911–922

Zargari S & Benford D 2012. Cloud Forensics: Concepts, Issues, and Challenges. 2012 *Third International Conference on Emerging Intelligent Data and Web Technologies*: 236–243

Zawoad S, Dutta AK & Hasan R 2013.SecLaaS: secure logging-as-a-service for cloud forensics. *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security* 2 Hangzhou, China. 2484342: ACM 219–230

Zawoad S & Hasan R 2013. Cloud Forensics: A Meta-Study of Challenges, Approaches, and Open Problems. arXiv preprint arXiv:1302.6312, 1–15

Zhang K, Shi Y, Li Q & Bian J 2009. Data Privacy Preserving Mechanism Based on Tenant Customization for SaaS. *International Conference on Multimedia Information Networking and Security 2009* (MINES '09),18–20 November: 599–603