Distributed filesystem forensics: XtreemFS as a case study

Ben Martini*, Kim-Kwang Raymond Choo
University of South Australia, GPO Box 2471, Adelaide, SA 5001, Australia

**ARTICLE INFO**

Article history:
Received 6 May 2014
Received in revised form 26 August 2014
Accepted 29 August 2014
Available online 11 October 2014

Keywords:
Big data
digital forensics
Distributed filesystem
Infrastructure as a Service (IaaS)
Storage as a Service (StaaS)
Distributed filesystem forensics
Cloud storage forensics

**ABSTRACT**

Distributed filesystems provide a cost-effective means of storing high-volume, velocity and variety information in cloud computing, big data and other contemporary systems. These technologies have the potential to be exploited for illegal purposes, which highlights the need for digital forensic investigations. However, there have been few papers published in the area of distributed filesystem forensics. In this paper, we aim to address this gap in knowledge. Using our previously published cloud forensic framework as the underlying basis, we conduct an in-depth forensic experiment on XtreemFS, a Contrail EU-funded project, as a case study for distributed filesystem forensics. We discuss the technical and process issues regarding collection of evidential data from distributed filesystems, particularly when used in cloud computing environments. A number of digital forensic artefacts are also discussed. We then propose a process for the collection of evidential data from distributed filesystems.

© 2014 Elsevier Ltd. All rights reserved.

**Introduction**

In recent years, the amount of data captured, stored and disseminated in electronic only form has increased exponentially (see Beath et al., 2012) and unsurprisingly, big data has constantly been ranked as one of the top ten technology trends (see Casonato et al., 2013; Chua, 2013) including by the United States National Intelligence Council (2012). A Gartner report, for example, has forecasted that “big data will generate [US]$232 billion in revenue cumulatively from 2011 to 2016” (Casonato et al., 2013, p. 4). A widely accepted definition of big data is from Gartner, which defines it as “high-volume, -velocity and -variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making” (Beyer and Laney, 2012, p. 2).

There are, however, large technology overheads and significant costs associated with the processing, storage and dissemination of big data. Businesses and governments will continue to be under pressure to deliver more with less especially in today’s economic landscape. Business and government users have recognised the advantages of cloud computing for processing and storing big data. For example, a report by the Info-communications Development Authority of Singapore (2012, p. 6), pointed out that “[e]arly adopters of Big Data on the cloud would be users deploying Hadoop clusters on the highly scalable and elastic environments provided by Infrastructure-as-a-Service (IaaS) providers such as Amazon Web Services and Rackspace, for test and development, and analysis of existing datasets”. This is unsurprising as cloud computing offers users the capacity, scalability, resilience, efficiency and availability required to work with high-volume, velocity and variety information.

Cloud computing (like other networked cyber-infrastructure) is subject to criminal exploitation (Choo, 2010; Chonka and Abawajy, 2012; Patel et al., 2013). In a digital investigation, one would need to gather evidence of an incident or crime that has involved electronic devices (e.g. computer systems and their associated networks) – a process known as digital forensics. The latter is increasingly being used in the courts in Australia and overseas. Many
conventional forensic tools have focused upon having physical access to the media that stores the data of potential interest. However, in a cloud computing environment it is often not possible or feasible to access the physical media that stores the user’s data (Martini and Choo, 2012). Distributed filesystems can support cloud computing environments by providing data fragmentation and distribution, potentially across the globe and within numerous datacentres. This presents significant technical and jurisdictional challenges in the identification and seizure of evidential data by law enforcement and national security agencies in criminal investigations (Hooper et al., 2013) as well as by businesses in civil litigation matters.

A number of researchers and practitioners have emphasised the need for cloud computing specific digital forensics guidelines (Birk and Wegener, 2011; National Institute of Standards and Technology, 2011; Zatyko and Bay, 2012), and we believe this need extends to the underlying infrastructure which supports cloud computing. While a number of published papers have provided a sound grounding for the research required in cloud forensics by highlighting the issues for digital forensic researchers and practitioners (Birk and Wegener, 2011; Martini and Choo, 2012) there are relatively few technical papers discussing the forensic collection of evidential data from cloud servers or underlying supporting infrastructure such as distributed filesystems.

Much of the existing literature has focused on the Software as a Service (SaaS) component of cloud computing (e.g. Dropbox, Skydrive and Google Drive) (Marty, 2011; Chung et al., 2012; Dykstra and Sherman, 2013; Hale, 2013; Martini and Choo, 2013; Federici, 2014; Quick et al., 2014) rather than the Infrastructure as a Service (IaaS) implementations that supports these services. Researchers such as Dykstra and Riehl (2013) and Hay et al. (2011) identified the various legal and technical challenges in conducting forensic investigation of cloud IaaS; but there has been no prevalent in-depth forensic investigation of a distributed filesystem which is used or analogous to those used in public or private cloud installations. One reason this may not have been thoroughly explored is due to the difficulties in accessing (for research purposes) a public cloud computing environment of significant scale that makes use of a distributed storage environment.

In recent years, a number of researchers have examined distributed filesystems and the implicit issues for forensic investigations on these complex systems. Cho et al. (2012) conducted a preliminary study of Hadoop’s distributed filesystem (HDFS). Hegarty et al. (2011) discuss a technique for distributed signature detection for the purpose of detecting the file signatures of illicit files in distributed filesystems. They note that existing signature techniques are unlikely to perform well in a distributed filesystem environment with a significant quantity of data stored. Almulla et al. (2013, p. 3) discuss a range of cloud forensic issues including the underlying role of ‘distributed computing’ and, in turn, distributed filesystems. The authors note the significant impact distributed filesystems have on forensics in terms of the requirement for a practitioner to ‘rebuild files from a range of filesystems’.

In this paper, we use XtreemFS as a case study to provide a better understanding of both the technical and process issues regarding collection of evidential data from distributed filesystems which are commonly used in cloud computing environments. XtreemFS, currently funded by the Contrail EU project (Contrail, 2013), is an open source example of a general purpose and fault-tolerant distributed and replicated filesystem that can be deployed for cloud and grid infrastructures to support big data initiatives (XtreemFS, 2013). To provide fault-tolerant file replication, the stored file data is generally split and replicated across multiple storage servers. In a cloud deployment, the data is also likely to be extensively distributed at the physical level within datacentres.

We chose to focus on a single distributed filesystem as this allows us to conduct an in-depth analysis of the client and, particularly, the servers to fully understand the potential evidential data that can be collected as part of a forensic investigation. XtreemFS (like most underlying infrastructure) does not receive substantial attention in mainstream technical media. However it has received significant attention in the academic community with many researchers choosing to analyse it or implement it as the underlying infrastructure in larger projects. Most commonly, in the literature, XtreemFS is implemented in cloud computing or grid computing (commonly understood to be one of the predecessors of cloud). For example Kielmann et al. (2010) describe the role of XtreemFS in supporting XtreemOS and its suitability to integrate with IaaS services. Pierre and Stratan (2012) integrate XtreemFS into their proposed ‘ConPaaS’ system which as the name describes is a Platform as a Service cloud environment. Enke et al. (2012) also implement XtreemFS (including a number of its advanced features) for the purpose of managing cloud data replication in their work to analyse distributed big datasets in Astronomy and Astrophysics. Krüger et al. (2014) note that XtreemFS has also been used to provide distributed data management in the MoSGrid science gateway (Molecular Simulation Grid), an EU-funded project. Kleineweber et al. (2014) selected XtreemFS for the underlying filesystem into which they integrated their reservation scheduler for object based filesystems (as an extension) to handle storage QoS in cloud environments. In addition to implementing or extending XtreemFS other researchers such as Dukaric and Juric (2013) and Petcu et al. (2013) have also noted its use when discussing filesystems generally in the cloud environment. This body of research demonstrates the contemporary applicability of XtreemFS in the cloud environment and, as such, makes it an appropriate choice as a case study for forensic investigation in this paper.

Cloud forensics framework

The digital forensics framework used in this paper is based on our previously published work (Martini and Choo, 2012), which we have previously validated using ownCloud (Martini and Choo, 2013; Quick et al., 2014). The framework is based upon the stages outlined by McKemmish (1999) and the National Institute of Standards and Technology (Kent et al., 2006) but differs in a number of significant
ways. The iterative nature of this framework is integral to a successful investigation in a complex client/server environment as presented in XtreemFS. The client can be used to identify the existence of cloud services and to collect any data stored by the client. Hence, forensic analysis of the client is generally carried out before analysis of the server environment. The following four stages outline the high level process and order that a forensic practitioner should follow when conducting forensic investigations in the cloud computing environment.

1. Evidence Source Identification and Preservation: This phase is concerned with identifying sources of evidence in a digital forensics investigation. During the first iteration, sources of evidence identified will generally be via a physical device (e.g. desktop computers, laptops and mobile devices) in possession of the suspect. However, in the case of a distributed filesystem used in cloud computing, the filesystem client may only exist on the cloud server nodes. This, however, does not prevent it from being the first point of identification and may lead to other components of the filesystem. During the second iteration, this phase is concerned with identifying other components of the environment or cloud which may be relevant to the case, possible evidence stored by the filesystem custodian (e.g. system administrator) and processes for preservation of this potential evidence. Preservation is integral to the integrity of forensic investigations and as such proper preservation techniques must be maintained regardless of the evidence source.

2. Collection: This phase is concerned with the actual capture of the data. There are various methods of evidential data collection suited for the various cloud computing platforms and deployment models. While IaaS may result in the collection of virtual disks and memory, and SaaS may result in an export from the relevant cloud software, the collection of distributed filesystems supporting cloud computing installations may be considerably more involved. Another consideration for distributed filesystems is the likelihood of remote hosting. If the filesystem is hosted outside of jurisdiction of the investigating LEA, they should use the appropriate legal instrument to legally gain access to the filesystem remotely.

3. Examination and Analysis: This phase is concerned with the examination and analysis of forensic data. Examination and analysis are key components of a forensic investigation dealing with distributed filesystems. Examination will be integral to gaining a complete understanding of the operating components in the filesystem, and analysis will be integral to reconstruction.

4. Reporting and Presentation: This phase is concerned with legal presentation of the evidence collected. This phase remains very similar to the frameworks of McKemmish and NIST (Martini and Choo, 2012). In general, the report should include information on all processes, the tools and applications used and any limitations to prevent false conclusions from being reached (see US NIJ, 2004).

Contributions and outline

We regard the contributions of this paper to be three-fold:

1. Provide technical insights on forensic analysis of the XtreemFS underlying infrastructure and IaaS instances;
2. Propose processes for the collection of electronic evidence from XtreemFS (see Fig. 1 in Summary section) and distributed filesystems used in the cloud computing environment based on the technical findings from the previous contribution (see Fig. 2 in Data storage section); and finally

In the next section, we provide an overview of the XtreemFS architecture and discuss the role of the various components. The findings are discussed in the context of these components. We follow the first three stages of the cloud forensic framework (see Cloud forensics framework section) to outline the recommended process for collection of forensic artefacts and potential evidence. The Collecting evidence from a distributed filesystem – a process section presents our proposed high level process for collection of electronic evidence from distributed filesystems used in cloud environments. The last section concludes this paper.

Experiment environment

XtreemFS architecture overview

XtreemFS is a virtual network-provisioned filesystem, which is used to deliver backend storage services for a cloud service provider by providing key services such as replication and striping. It is one example of a number of products available with similar feature sets (other examples include GlusterFS (Gluster, 2014), BeeGFS (Fraunhofer, 2014) and Ceph (Ceph, 2014)). It is important to make the distinction between backend and frontend storage systems in the cloud computing environment as both are commonplace. For the purposes of this research, frontend cloud storage systems refers to cloud storage that is purchased by users to store their personal files (popular examples include Dropbox, Skydrive and Google Drive) whereas backend cloud storage systems are used by the cloud provider to support IaaS, PaaS or SaaS services. For example a backend storage system would be used to store the virtual machines that are hosted as part of an IaaS cloud or the databases and other files used by a SaaS system. In most cases backend storage is not provided directly to users however it is acknowledged that in some IaaS instances users may access this type of storage directly.

Two major features provided by XtreemFS are striped and replicated filesystem services that is achieved using three main components, namely the Directory Service (DIR), the Metadata and Replica Catalog(s) (MRC) and the Object Storage Device(s) (OSD) (Stender et al., 2012). These components work together to provide the virtual...
filesystem to network users, and the various components communicate between each other and with clients as described below.

The DIR is responsible for maintaining a registry of all services and volumes provided by the XtreemFS service (Stender et al., 2012). In this respect all other parts of the XtreemFS architecture (including clients) regularly communicate status and other information to the DIR service. This indicates a possible source of forensic identification information about an XtreemFS instance, which may include the (virtual) location of the OSD storage on a local or remote network and the location and number of clients connected to the environment.

The MRC is responsible for storing and managing all of the metadata that XtreemFS generates and stores regarding the file content. Stender et al. (2012, p. 271) define metadata as “a collective term for all types of data that need to be managed by a filesystem except for file content”. Metadata forms a critical part of many forensic and civil litigation investigations (see Irwin v. Onondaga County Resource Recovery, 72 AD3d 314 – NY: Appellate Div., 4th Dept. 2010; Ellis v. Toshiba America Information Systems, Inc., Cal.: Court of Appeal, 2nd Appellate Dist., 1st Div. 2013; Voltage Pictures, LLC V. Does 1-31, Dist. Court, SD Georgia 2013), and is reportedly collected by the National Security Agency as part of the agency’s PRISM program (United States Foreign Intelligence Surveillance Court, 2011; Gellman and Lindeman, 2013; Greenwald and Ball, 2013).

As such, the MRC could be a potential source of forensic identification data and data for examination and analysis. Consideration must also be given to the preservation of the metadata stored on the MRC.

The OSD is responsible for storing the actual file data sent by the clients in the XtreemFS instance. A variety of structures are used to store the data depending on the striping, replication, and other options selected on a per volume (or per file) basis; and this is discussed further in the findings section of the paper. The OSD would likely be the focal component of the XtreemFS system for a forensic practitioner as it stores the file content data that a client has added to the virtual filesystem. This makes the OSD a key component for evidence preservation (to ensure that the data stored by an individual of interest is not modified intentionally or unintentionally) and forensic analysis. Having more than one OSD in an XtreemFS instance would be commonplace, in fact it is mandatory to make use of some of the key features of XtreemFS (e.g. replication and striping). An XtreemFS instance with multiple OSDs allows for file content to be split and replicated across a number of physical devices potentially in a number of different physical locations.

XtreemFS uses the concept of ‘volumes’ to virtually segregate data. Volumes can be used in a number of ways, for example, permissions and default policies (e.g. replication and striping) can be applied to volumes. Volumes are also the primary administrative unit for a client, for example, clients mount volumes, set policies on volumes and create snapshots of volumes, etc. An individual volume could conceivably be created for each user of the XtreemFS system, alternatively the POSIX permissions system integrated into XtreemFS could be used to segregate users. We believe that a common implementation may be to leave authentication and permissions management to higher level applications. For example, a file sync application using XtreemFS as a backend could use a single volume, with each file owned by a single service user relying on the file sync application server to enforce permissions and provide authentication.

Finally the XtreemFS client is used to connect to the XtreemFS filesystem. The client (and related tools) is responsible for all administrative operations and for facilitating local access to the virtual filesystem. Administrative operations include creating volumes, mounting volumes, setting volume and file policies, and checking status (e.g. replication and striping) of files and volumes. While it is not expected that there will be a wealth of data of forensic interest available on a host with a disconnected client, a connected client may provide a forensic practitioner with valuable identification information (see XtreemFS Client section) and potentially logical collection opportunities.

**Experiment environment overview**

The experiment environment consists of a number of virtual machines providing various XtreemFS features (e.g. DIR, MRC, OSD and client) depending on the specific experiment. The virtual machines (VMs) simplified forensic disk image collection and would simulate a cloud configuration where physical hosts house both distributed compute and storage functions. Clients were also hosted as VMs.

A number of client and server VMs were created using CentOS 6.4 as their operating system and ext4 for their local filesystem. We used the prebuilt XtreemFS packages built for the CentOS 6 operating system.

**Findings**

This section will discuss the three main architectural components of the XtreemFS system and the client in the context of the relevant phases of the cloud forensics framework with a view to understanding the filesystem and data of forensic interest available from the XtreemFS system, ultimately producing a list of high-level artefacts that should be investigated in all distributed filesystem forensic investigations. As XtreemFS is an advanced and complex environment, it is beyond the scope of a single paper to cover every possible combination of its potentially enabled features. Hence, we focus on the features most relevant to a forensic researcher or practitioner.

**Directory service (DIR)**

The DIR stores the data needed to define and locate the various technical components in an XtreemFS instance. For a practitioner commencing an investigation with an identified XtreemFS instance, this would be the logical starting point to determine the components and extent of the XtreemFS installation.

Three artefact types of potential value to a forensic investigation exist on the DIR server:
• **Volatile environment metadata** — As the directory service is responsible for maintaining a record of the various components in the XtreemFS environment, a range of environmental metadata of interest should exist. This can include the logical network location (generally an IP address) of the various filesystem nodes and unique identifiers for those nodes. Other data of interest (where available) includes data about the individual nodes such as node type/responsibilities, ownership information and configuration (especially in terms of authentication).

• **Non-volatile environment metadata** — While in many cases the DIR may store the majority of directory data in volatile storage as it is expected to change (semi) frequently, some data may be committed to non-volatile storage (e.g. disk). This includes all of the metadata listed above. The most prevalent example would be logging data but may also include backups for example.

• **Configuration files** — Configuration files can be an invaluable source of information for a forensic researcher and practitioner seeking to gain a better understanding of the operation of the system with a view to collecting evidence from the individual components. Configuration information of interest may include network information (addresses/port used, etc.), authentication information (systems/passwords, secrets, etc.) and operational information (local storage locations, database formats, etc.).

**Evidence source identification and preservation**

The most obvious use of the DIR component would be as part of identification and assisting in preservation of the distributed storage data. Within this context we are seeking to find and decode any available data on the components of the XtreemFS instance being investigated.

Before the value of the DIR component can be assessed it must be located. The most obvious method of locating the DIR server would be to analyse the currently mounted filesystem on a client (discussed in XtreemFS Client section). If this is infeasible (and the practitioner has access to the LAN segment where the DIR server is assumed to be hosted) the practitioner could potentially exploit the DIR auto detection feature available to the XtreemFS service components. The XtreemFS installation documentation notes that MRC and OSD components can “autodiscover” the DIR using a UDP broadcast. This was tested by enabling the UDP autodiscover feature on an OSD in the experiment environment and capturing the resulting packets. It was noted that the OSD sent a UDP packet to the broadcast address of the LAN on the DIR service port (32638) to which the DIR responded. While this functionality can be optionally disabled in the DIR configuration, it is enabled by default and as such replicating this UDP broadcast is a feasible method for a practitioner to locate the DIR server(s) on a LAN segment. This was tested successfully as part of our experiments. We took this concept further and sent the UDP packet contents captured from a DIR discovery to the broadcast address on the OSD service port (32640). This resulted in all OSDs on the LAN segment responding with an error message. A practitioner could make use of this method (or a similar derivation) to locate all of the XtreemFS OSDs in a LAN segment where other forms of network discovery have been intentionally or unintentionally disabled.

Once the DIR has been located and the practitioner has gained access to the operating system (with the assistance of an administrator or using standard digital forensics methods), the practitioner should focus on understanding the configuration of the DIR in this installation. The DIR service has a plaintext configuration file located (by default) at ‘/etc/xos/xtreemfs/dirconfig.properties’. This configuration file contains a range of directives for the DIR service. These include the capacity to enable debug logging, configuration of the service port (for communication with other components of the XtreemFS system) and the HTTP port (for accessing administrative status information), directives to enable SSL communication (disabled by default), monitoring configuration (email and SNMP) and ‘BabuDB’ (XtreemFS database) configuration directives. The configuration file also contains the DIR service UUID, extensive use is made of UUIDs in the XtreemFS system with most DIR addressable items being assigned a UUID.

All XtreemFS services provide a HTTP service to present system administrators with status information. The default port for the DIR HTTP status service is 30638. By default, the status service can be accessed without authentication. However, if an administrator password is enabled in the configuration file, it will be required to access the status page. When loaded in a web browser, the HTTP status service provides a range of information that could be of interest to a forensic practitioner, this includes the following:

• **Address mapping**: IP address to universally unique identifier (UUID) mapping for each network accessible service in the XtreemFS installation. This provides two methods for a forensic practitioner to ensure they preserve and collect all services (generally on different physical devices) of interest, these being IP address (for provider and device location) and UUID (which can be confirmed using the service configuration file once access has been arranged to the host).

• **Service registry**: A range of registry information for each component listed with the directory service. Generic mappings in this category include the service type (e.g. ‘SERVICE_TYPE_OSD’ or ‘SERVICE_TYPE_VOLUME’), name (generally the service type and UUID, volume name for volumes), static_status (online, locked, etc.) and last updated (in UNIX timestamp and converted to local timezone). Specific services (MRC, OSD, volumes, etc.) also have specific entries in the registry such as free disk space and total/used RAM and load. For a forensic practitioner the focus in the service registry is to determine what each UUID referenced throughout the system refers to (e.g. a specific volume, an OSD or an MRC, etc.) through the use of the type and UUID (or name) entries.

• **Configurations**: Each XtreemFS service listed with the DIR has a range of configuration details listed against its UUID in the configurations table. These can include the ports on which the service itself and HTTP monitoring can be accessed which may be useful if a practitioner is...
considering accessing the service via the network. There is also an indication if SSL is enabled on the service. One directive of particular interest in the MRC configuration entry is “authentication_provider” that notes the type of authentication enabled on the MRC. By default the “NullAuthenticationProvider” is used that makes use of local users and groups on the clients. However, other authentication schemes can be enabled which could complicate a logical extraction using the XtreamFS client.

Two of the most interesting OSD configuration entries include the “storage_layout” and the “object_dir” that indicate the layout and location on host storage of the objects stored with the individual OSD instance respectively. With this information a practitioner can begin to plan for collection from the OSD (if the practitioner is seeking to undertake a physical collection). This is discussed further as part of the high-level evidence collection process (see Collecting evidence from a distributed filesystem — a process section).

Network monitoring (packet collection) the services port is another potential source of data for a forensic practitioner in terms of monitoring the hosts connecting (or attempting to connect) to the XtreamFS services.

Collection & examination and analysis

In most cases a practitioner would seek to collect the environment directory metadata held by the DIR service for use in identifying other components of the XtreamFS installation or for rebuilding parts of the installation in a post event laboratory situation where required. Examples of this data are outlined in Evidence source identification and preservation section. Collecting the data via the HTTP status interface discussed as part of identification was, in our experience, one of the most straightforward and comprehensive methods of collecting the current state of the XtreamFS instance directory. However, it is only possible to collect this data from the running XtreamFS DIR server that may not always be available to a forensic practitioner who, for example, may be relying on offline physical/logical copies of the DIR server.

The XtreamFS DIR service stores some metadata in a “BabuDB” database (a key pair based database system closely linked with XtreamFS and the XtreamFS developers). This database keeps a binary log which could be of use if a practitioner is working with an offline copy of the DIR service or potentially looking for historical data on the components which were part of the XtreamFS instance but have since been removed. Unfortunately examination and analysis of the databases binary log is not a straightforward process for the DIR service, and we tried different methods, such as rebuilding the database using the BabuDB Java library and raw keyword searches, with varying levels of success as discussed below.

The database files for the DIR service are located by default in “/var/lib/xtreamfs/dir/” where there are two subdirectories named “database” and “db-log”. When collecting evidence from a DIR server, the “/var/lib/xtreamfs/dir/” directory and all subdirectories and files should be collected. The “database” directory contains a “config.db” file that appears to store configuration information for the DIR databases and two subdirectories named “dirdb” and “dirdbver”. These subdirectories are the names for the two databases used by the DIR service, and in our experiments they did not store any persistent files. However, the name of the database will be needed if the practitioner chooses to use the BabuDB Java library to examine the database. The “db-log” directory generally contains a number of files (the specific number depending on the length of time the XtreamFS instance has been operating and configuration settings) which appear to be snapshots of the database binary log using the extension “.dbf” with a filename of the format “.l.nnnn” where n is any number of integers presumably a type of sequence number. BabuDB can recreate the last state of the database using these snapshots, which we attempted using the BabuDB JAR library with some success. Usage examples for the BabuDB library, which can be adapted for the purposes of reconstruction, are provided by the developers (BabuDB, 2012). This provided access to the UUID and IP address of the MRC and OSD components of the XtreamFS instance in its state before the collection. Another method of analysis a practitioner could undertake on the databases binary log files is a raw keyword search (for IP addresses) on the files using a forensic tool such as Guidance Software’s EnCase. This would allow the practitioner to potentially locate all IP addresses associated with DIR listed components for the life of the database logs. We tested this method with success and found that the service type (e.g. MRC, OSD) could often be decoded by the associated port number.

The DIR server should have its system time recorded (particularly any offset from real time) as this may be needed later to verify/explain the timestamps recorded by the system. This record should also be made for any other XtreamFS system component examined.

The DIR component stores an operations log at “/var/log/xtreamfs/dir.log” which should be collected and analysed. In our experiments, this log did not contain any entries which we deemed to be of general interest to a forensic practitioner. However, this would vary on a case by case basis. It should also be noted that the DIR supports different logging (verbosity) levels which are noted in the DIR configuration file. In our experiments, we left this value at the default which does not result in particularly verbose logging. More information of interest may, however, be logged if the administrator had raised the level of logging before the commencement of the forensic investigation.

Metadata and replica catalog (MRC)

The MRC stores a broad range of metadata relating to the volumes in the XtreamFS instance and the files/directories stored within these volumes. This includes high level data such as the number of OSDs (locations) a file is striped/replicated across and the OSD identifier (UUID) down to file level metadata critical for practitioners such as filename, size, temporal information (access, creation, modification times) and permissions/ownership information. A practitioner can use this information (combined
with information gained from the DIR service) to not only locate components of files of interest but also potentially gain a better understanding of the provenance of the files in the XtreemFS instance.

Three artefact types of potential value to a forensic investigation exist on the MRC server:

- **Volatile and non-volatile construct metadata** — While the DIR service provides metadata for the service components that comprise an XtreemFS instance, the MRC stores metadata which defines internal constructs to the XtreemFS filesystem (e.g. volume metadata).

- **Volatile and non-volatile file metadata** — This includes (1) high level metadata such as the number of OSDs (locations) a file is striped/replicated across and the OSD identifiers (e.g. UUID), (2) down to low level file metadata critical for practitioners such as filename, size, temporal information (access, creation, modification times) and permissions/ownership information.

- **Configuration files** — Configuration information of interest for understanding the environment and MRC operations may include network information (addresses/ports used, etc.), authentication information (systems, passwords, secrets, etc.) and operational information (temporal information, structured metadata formats, etc.).

**Evidence source identification and preservation**

Using the data gathered from the DIR service a practitioner should be able to identify the logical (network) location of the MRC (e.g. via datacentre switch/routing examination). Once the practitioner has used standard forensics methods to gain access to the server hosting the MRC service (either remotely or locally) they should commence identification by reviewing/confirming the configuration of the service using the configuration files. The MRC service has a plaintext configuration file located (by default) at “/etc/xos/xtreemfs/mrcconfig.properties”. This configuration file has many similar directives to the DIR configuration file which we have reviewed (e.g. port numbers for the service and its HTTP monitoring facility, SSL configuration, service UUID). However, there are also MRC configuration directives which may be of interest. It should be noted that much of the information available in the configuration file is also available in the configuration section of the DIR HTTP status page (excluding any passwords or secrets). The MRC has a configuration directive for defining the DIR service hostname/IP address (dir_service.host) and port (dir_service.port) along with directives for defining replica DIR service hostname/IP and ports (e.g. dir_service2.host). This may be of interest for a practitioner (if this information has not already been collected) to determine the entities which are hosting and have access to the XtreemFS instance. A number of file temporal metadata configuration directives are set at the MRC level, these include whether file access times are updated (no.atime) and the method/conﬁgurations for determining the current time for ﬁle temporal information across the distributed filesystem.

The MRC HTTP status page (by default available on port 30636) lists a number of configuration directives which may be of use to a forensic practitioner. As with the DIR status page, the MRC status page can be accessed by default without authentication. However, if an administrator password is enabled in the MRC configuration file, the administrator password will be required to access the status page. The page lists configuration information (e.g. the network location of the directory service), various load and request statistics and volume information. The volume information is of particular interest to a practitioner if they are searching for a particular volume or are seeking data on volume sizes/configurations to determine if physical acquisition will be feasible. This information allows the practitioner to determine the type of volumes hosted by the MRC in terms of striping, access and replication policies. It also provides information on selectable OSDs (to determine which OSDs store data for the volume) and the number of files and occupied disk space for the volume. This information can be useful for the identiﬁcation stage of the forensic process when a practitioner is making decisions on the best method of evidence collection for the XtreemFS environment.

The MRC is also responsible for providing authentication services for the virtual filesystem. At the time of research, XtreemFS listed support in its installation guide for two types of authentication, “org.xtreemfs.common.auth.NullAuthProvider” (which is the default) relies upon the user information provided from the local operating system by the client. Depending on the environment in which XtreemFS is being used, this could be a security issue. For example, a user which has root privileges on any PC which has network access to the XtreemFS instance could potentially access all files in an XtreemFS volume. From a digital forensic perspective, this is a mixed blessing. If a practitioner has access to the XtreemFS system via the network, then it would be straightforward to connect as a root user and collect evidence. However, this introduces issues of provenance as other unknown individuals may have had the potential to use the same method to access/modify files without the file/volume “owners” knowledge. The other authentication provider included in XtreemFS is “org.xtreemfs.common.auth.SimpleX509AuthProvider” which makes use of X.509 certificates for client authentication. While this is a much more secure solution, it is not enabled by default. When enabled SSL is used for authentication between peer services and between the clients and services, it also (optionally but by default) encrypts all trafﬁc as it traverses the XtreemFS instance. SSL encryption and the use of X.509 certificates are discussed further in the client analysis section (XtreemFS Client section). However, it should be noted that unlike the default NullAuth configuration if X.509 authentication is enabled practitioners will not be able to connect to collect evidence using the XtreemFS client without the relevant user key, a “superuser” key or a host/service key (which reverts the authentication to trusting the local client for user credentials) — standard POSIX access denied errors are received if unauthorised access is attempted.
Collection

Collection of MRC data is critical not only as the metadata may provide a form of evidence but also for the reconstruction of the XtreemFS environment if it is not currently functioning or fully accessible. It is also critical if a practitioner seeks to reconstruct files from physical extractions of the relevant OSD components. Once the practitioner has access to the MRC server there are two methods of collecting the MRC database.

As discussed in Collection & examination and analysis section the database binary logs can be collected and replayed using the BabuDB APIs to form the last state of the database. This may be the only feasible method of collection if the MRC component of the XtreemFS instance has been shutdown or the practitioner would like to use standard forensic methods to attempt to collect previous versions of the MRC to determine if files existed in previous versions of the environment metadata. However, in our experiments, we found this method more difficult to execute and we had mixed results in collection of usable data using this method (e.g. some reconstructions of the database resulted in corrupt output). Similarly to the DIR component the MRC database files are located by default in ”/var/lib/xtreemfs/mrc” where there are two subdirectories named “database” and “db-log”. The database collection and examination techniques discussed in Collection & examination and analysis section can be similarly applied if the practitioner needs to use this method of collection.

Unlike the DIR component, the MRC component includes a much simpler method for collecting the current state of the MRC database. This collection is achieved by using the “xtfs_mrcdbtool” which is part of the “xtreemfs-tools” package. The latter would be likely installed on the MRC host but can also be used from another host on the network which has access to the MRC or could potentially be installed on and run from removable storage on the MRC. “xtfs_mrcdbtool” allows a user to dump and restore the MRC database to/from a human readable XML format. Assuming that SSL/certificate based authentication and/or an administrator password has not been enabled, the syntax for the tool is as follows: “xtfs_mrcdbtool -mrc pbrpc://localhost:32636 dump /tmp/mrcdump.xml”. This command connects to the MRC on localhost and dumps the MRC database to a file in “/tmp” named “mrcdump.xml”. The practitioner can then open this file in a text editor to determine that the dump has been successful and copy the file to external media as a method of collection.

Similarly to the DIR component (see Collection & examination and analysis section), the MRC stores an operations log at “/var/log/xtreemfs/mrc.log” which should be collected and analysed. Also similarly to the DIR log, little information of general forensic interest was found in this log from our experiments, although the logging level selected may affect this result.

Examination and analysis

Assuming that the practitioner has collected the MRC database using the “xtfs_mrcdbtool” the resulting XML file can be partially analysed using any text editor. If the practitioner has collected the database files manually the examination and analysis processes will be similar to those discussed in Collection & examination and analysis section, for the remainder of this section we will assume that the practitioner has dumped the MRC database to an XML file and is seeking to examine and analyse its contents. Although most of the XML file is plaintext, some attributes are encoded using BASE64 which makes the values unreadable until they are decoded. We wrote an application to parse the XML file, display its content and decode any BASE64 encoded attributes.

In our experiments we found the broad (non comprehensive) structure of the MRC XML file to be as follows, elements are listed as dot points and their attributes are indented:

- The root element was “FILESYSTEM”
  - “DBVERSION” – This is a numerical identifier of the database version.
  - The next element was “VOLUME”
    - “ID” – The UUID for the volume generated by the XtreemFS system.
    - “NAME” – The volume name entered by the user which created the volume.
    - “ACPOLICY” – The numerical identifier for the “Authorization Policy” (or access control policy) used on the volume. These numerical identifiers and their meanings are defined in the “XtreemFS Installation and User Guide”.
  - The next element was “DIR”, this represents directories in the volume but the first entry represents the root directory of the volume.
    - “ID” – The file ID assigned by XtreemFS for the directory. The file ID is an integer. Generally, the root directory for the volume is “1” and the integer is incremented with the addition of files/directories.
    - “NAME” – The name assigned by the user for the directory, the volume name for the root directory.
    - “UID” and “GID” – The user and group respectively which own the directory, this is represented in the XML dump as a username (e.g. “root”). Depending on the authentication service used (see Evidence source identification and preservation section), this information may not be reliable.
    - “ATIME”, “CTIME”, “MTIME” – Accessed, created and modified times respectively, these are represented as POSIX timestamps in UTC. Notably during our experiments, the CTIME and MTIME values were updated when the contents of the folder were accessed or changed. Timestamps should be accurate between XtreemFS nodes as server clocks are synced with the DIR as part of the systems operation. The practitioner should check the DIR server (and other nodes for completeness) for any offset between real time and system time.
    - “RIGHTS” – The numerical representation of the POSIX permissions applied to the directory. The numerical representation is somewhat different from the standard representation used for POSIX permissions. Owner, group and others are the three classes
of permissions with read, write and execute being the three types. For the “others” class 1, 2 and 4 represent execute, write and read respectively, for the “group” class 8, 16 and 32 represent execute, write and read respectively and for the “owner” class 64, 128 and 256 represent execute, write and read respectively. These numbers are summed depending on the permissions applied to the directory and stored as a single number. For example if the owner has read and write permissions, the group has read and others have execute (traverse for directories) the permission stored would be 417 (i.e. 128 + 256 + 32 + 1 = 417). Table 1 provides a visual summary of this representation.

- “W32ATTRS” — This was reported as the numerical value of “0” in our experiments, presumably as we did not use the Windows version of XtreemFS.
- The next element was “ATTRS” which encloses the “ATTR” element(s) that contain attributes applicable to the DIR. These “ATTR’s” were only noted in relation to the volumes root directory and not other directories or files in our experiments. The “ATTRS” element did not have attributes in our experiments, and the “ATTR” elements had four elements “KEY,” “VALUE,” “ENC” and “OWNER”. All “VALUE”’s were encoded in Base64 (as noted in the “ENC” attribute), and these values were decoded by our application and are represented in plaintext below. The “KEY”’s are listed and discussed below.
- “allowSnaps” — This true or false value represents whether snapshots have been enabled on the volume.
- “osdPol” — These comma separated numerical values represent the OSD policies applicable for this volume. These values include policies for filtering, grouping and sorting OSDs for the volume. This may be of interest to a practitioner if they wanted to determine why particular OSDs were used to store file stripes but will not allow a practitioner to determine which OSDs were selected at the time of file creation and as such the current location of the file stripes.
- “sp” — A textual representation of the striping policy which applies to the volume in a similar format to that displayed in the “xtxsutil” client application. The format of this string is described in the “XtreemFS Installation and User Guide”, it denotes the type of striping (only “RAID0” is supported at time of research), the stripe size in KB and the number of stripes to create (if that quantity of OSDs is available).
- “acPol” — This appears to replicate the “ACPOLICY” attribute in the “VOLUME” element.
- “volId” — This appears to replicate the “ID” attribute in the “VOLUME” element.
- “replPol” — This appears to store the replication policy for the volume, as we did not comprehensively study replication (and it was disabled for this experiment) this value was blank.
- “scrubber.latestscrub” — The POSIX timestamp representing the last time a scrub operation (filesystem clean-up) was run on the volume. Notably this element includes an extra element of “OWNER” which lists the username of the user which started the scrubbing operation.
- The next element was “FILE” in our experiments; this represents the metadata for the first file we added to the directory. However, it could have also been a “DIR” element. The XML representation mimics the hierarchical nature of the filesystem.
- “ID” — The file ID (an integer) assigned by XtreemFS for the file, this is a critical identifier if the practitioner is seeking to manually locate/restore files. Files in the XtreemFS system are addressed by the volume UUID and their file ID.
- “NAME” — The name assigned by the user for the file, the “filename”.
- “SIZE” — The size of the file in bytes.
- “EPOCH” — A numerical representation, set to 0 and 1 in our experiments.
- “ISSUEDEPOCH” — A numerical representation, set to 0 and 1 in our experiments.
- “UID” and “GID” — The user and group respectively which own the directory, this is represented in the XML dump as a username (i.e. “root”). Depending on the authentication service used (see Evidence source identification and preservation section) this information may not be reliable.
- “ATIME”, “CTIME”, “MTIME” — Access, created and modified times respectively, these are represented as POSIX timestamps in UTC. Timestamps should be accurate between XtreemFS nodes as server clocks are synced with the DIR as part of the systems operation. The practitioner should check the DIR server (and other nodes for completeness) for any offset between real time and system time.
- “RIGHTS” — The numerical representation of the POSIX permissions applied to the file. As discussed above, the numerical representation used to store permissions in the XtreemFS database is somewhat different from the standard representation used for POSIX permissions. File rights representations also differ somewhat from directory rights. While the same numbers are used to represent the permission types for the three classes, file permissions are subtracted from the 16-bit integer limit (32,768). As such file permissions are summed depending on the permissions applied to the file. This number is

<table>
<thead>
<tr>
<th>Class Type</th>
<th>Others</th>
<th>Group</th>
<th>Owner</th>
<th>XtreemFS directory RIGHTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E</td>
<td>W</td>
<td>R</td>
<td>E</td>
</tr>
<tr>
<td>XtreemFS permission value</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>XtreemFS value total</td>
<td>1</td>
<td>32</td>
<td>417</td>
<td></td>
</tr>
</tbody>
</table>

Note: E, W and R denote Execute, Write and Read respectively.
then subtracted from the 16-bit integer limit and stored as a single number. For example if the file owner has read and write permissions, the group has read and others have execute the permission stored would be 32,351 (i.e. 32,768 – 417 = 32,351, where 417 = 128 + 256 + 32 + 1).

- “W32ATTRS” – This was reported as the numerical value of “0” in our experiments, presumably as we did not use the Windows version of XtreemFS.
- “READONLY” – A true or false value to note if the file is read-only after creation in the XtreemFS system.
- The next element was “XLOCLIST” in our experiments, and this begins the metadata on the location of file stripes and replicas. “XLOCLIST” notes a version number and the replication policy (e.g. “WqRq”) that determines how replicas are created and if replicas are read only or read/write. This attribute is blank if replication is not enabled.
- The next element is “XLOC” that contains striping and replication attributes for the file.
- “PATTERN” – A string representation of the striping policy for the file, and this may differ from the volume striping policy as the striping policy can be changed at the file/directory level.
- The next element is “OSD” that contains OSD location for the file.
- “LOCATION” – The UUID of the OSD that stores a stripe/replica of the file. This information is critical to a practitioner looking to manually reconstruct files in the XtreemFS instance as it will indicate which OSDs need to be accessed to collect file stripes for reconstruction.

The summary of attributes of common forensic interest to a practitioner is outlined in Table 2.

### Object storage device (OSD)

The OSDs will be the core component of forensic interest as the OSD stores the data stripes which allow the practitioner to reconstruct the files. The OSD also has the potential for a practitioner to recover deleted file parts using existing forensic techniques on the underlying filesystem. The practitioner’s identification and analysis of previous components (DIR and MRC) will provide the information they need to determine which OSDs in the network contain the data they are seeking to collect and allow the practitioner to identify the relevant individual files on the OSDs.

Two artefact types of potential value to a forensic investigation exist on the OSD server(s):

- Non-volatile file data – The OSD is responsible for the content of files stored in the XtreemFS filesystem which would form the majority of non-volatile data of interest on the OSD(s). Other data that should be collected is any local metadata (e.g. local filesystem permissions) and log data.
- Configuration files – Configuration information of interest for understanding the environment and OSD operations may include network information (addresses/ports used, etc.), authentication information (systems, passwords, secrets, etc.) and operational information

<table>
<thead>
<tr>
<th>Element</th>
<th>Attribute/key</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOLUME ID</td>
<td>The volume ID (UUID) is necessary to manually reconstruct files stored on the OSDs.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>The volume name is potentially useful to determine the owner of the volume and/or its contents.</td>
<td></td>
</tr>
<tr>
<td>DIR</td>
<td>The directory ID and name can be useful in reporting. They may also assist in determining provenance.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>The UID and GID owners and their associated rights for the directory may be useful in determining who had access to a directory and data provenance.</td>
<td></td>
</tr>
<tr>
<td>UID &amp; GID RIGHTS</td>
<td>The UID and GID owners and their associated rights for the file may be useful in determining who created/had access to a file.</td>
<td></td>
</tr>
<tr>
<td>CTIME</td>
<td>The created, accessed and modified times may be useful in determining when the directories contents were accessed or modified.</td>
<td></td>
</tr>
<tr>
<td>ATIME</td>
<td>Used to determine if snapshots are enabled on the volume.</td>
<td></td>
</tr>
<tr>
<td>MTIME</td>
<td>Used to confirm the type of striping used, size of individual stripes and potential number of stripes created for files in this volume.</td>
<td></td>
</tr>
<tr>
<td>FILE ID</td>
<td>The file ID and name can be useful in reporting. They may also assist in determining provenance.</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>The UID and GID owners and their associated rights for the file may be useful in determining who created/had access to a file.</td>
<td></td>
</tr>
<tr>
<td>UID &amp; GID RIGHTS</td>
<td>The created, accessed and modified times may be useful in determining when files were last added, accessed (when enabled) and changed.</td>
<td></td>
</tr>
<tr>
<td>XLOC PATTERN</td>
<td>The specific stripe pattern used for an individual file. It should be noted that this may be different to the volume striping pattern.</td>
<td></td>
</tr>
<tr>
<td>OSD LOCATION</td>
<td>The UUID of an OSD which stores the stripe(s) of the file. There is generally more than one OSD entry for a file with striping (or replication) enabled.</td>
<td></td>
</tr>
</tbody>
</table>

(physical data storage locations, data structure formats, etc.).

### Evidence source identification and preservation

The OSD is the most likely component in the XtreemFS architecture to be installed numerous times across a number of disparate physical devices. Regardless of whether the administrator is seeking to use XtreemFS striping or replication functions (or a combination of both), multiple OSDs are fundamental to the effective operation (e.g. replication for higher IOPS – Input/Output Operations Per Second) of the XtreemFS system. Identification and preservation of these components is particularly critical for a forensic practitioner as OSDs store the actual data uploaded by the clients.

The DIR component that the practitioner will have analysed as part of an earlier phase provides the practitioner with the network location of the OSDs, which should be useful in locating the physical OSD host devices for
preservation and collection. As with the DIR and MRC components, the OSD component has configuration directives that define which ports are used for communication, how the DIR is located and a basic status page. The OSD configuration file is located (by default) at “/etc/xos/xtreemfs/osdconfig.properties”. The configuration file contains similar directives to the other two configuration files discussed (such as the listen/http ports, DIR host/port and SSL configuration). However, it also contains some important OSD specific directives.

Perhaps the most important directive from the perspective of a forensic practitioner is the “object_dir” directive, which specifies the filesystem location of the OSD storage. This directive (by default) is set to “/var/lib/xtreemfs/objs/”. However, it could and, presumably, would often be changed to suit the individual implementations. The forensic practitioner can use this directive to start the search for the physical system/media that holds the OSD data. For example, the data could be stored on a local disk (or disk array) in the server or it could be hosted on a NAS (Network Attached Storage) device or SAN (Storage Area Network). In the latter case, the “object_dir” directive would point to a mount point which the operating system provides to the remote storage. The use of this mount point (either directly or using the XtreamFS system) could be the most feasible method for a practitioner to collect a logical copy of data stored by the XtreamFS environment as access to the remote storage device may be complicated by authentication or other technical requirements.

The OSD HTTP status interface provides runtime statistics (which only require authentication when enabled in the configuration file as with the DIR and MRC) on the OSD mostly in terms of usage. However, these statistics are of limited interest to a forensic investigation. The availability of this interface (and port), however, could be a method of network based identification (as with the other XtreamFS components).

Collection

The collection procedure for the OSD will vary depending on the requirements of the practitioner. As OSDs generally store, stripes of files a practitioner may choose to use the XtreamFS client tools (discussed in XtreamFS Client section) to mount the XtreamFS filesystem and collect the relevant files. However, if for any reason (e.g. lack of network access, authentication issues, the need for a more thorough physical collection of file stripes) the practitioner chooses not to use this method, they will need to manually locate and reconstruct the files of relevance. Using the information collected from the MRC (and DIR), the practitioner can locate and collect only the relevant files’ parts, rather than needing to collect an entire XtreamFS installation (potentially consisting of many OSDs). In a medium to large XtreamFS environment, this could be the only feasible option for a forensic practitioner as physical collection of the entire XtreamFS environment (which may consist of thousands of devices) would not be feasible.

Once a practitioner has examined the metadata collected from the MRC and identified the files/volumes they need to collect, they can identify the OSD(s) that store the file stripes by UUID. This UUID can be used with the information collected from the DIR to locate the OSD on the network and, in turn, locate the device physically (or where appropriate access it remotely). Using the “object_dir” and mount point information available (as discussed above), a practitioner can determine where the data is being physically stored and make a decision as to whether they will collect the file stripes logically (e.g. by gathering the available filesystem level metadata and then copying the relevant stripes from the mount point on the OSD to external media) or by collecting a physical image of the storage device or relevant parts of the device. If a practitioner seeks to collect only individual file stripes (logically or physically), they will need to understand the hierarchy of the “object_dir” which is discussed in detail below.

Similarly to the DIR and MRC components (see Collection & examination and analysis and Examination and analysis sections), the OSD component stores an operations log at “/var/log/xtreemfs/osd/log” which should be collected and analysed. As we found that the default logging settings did not provide significant detail of interest for forensic purposes, we changed the logging level (verbosity) on one of the OSDs in our experiment environment to the highest level, 7 — “debug”. With this level of logging enabled, we found that much more relevant information was logged and, as such, would be available as part of a forensic investigation. The usefulness of the information presented would still be case specific. However, as this level of logging records data to the granularity of individual file stripes, this has the potential to be significantly more useful than standard logging. We note that it is, of course, not possible to change the logging level after an event has occurred which requires forensic investigation. Therefore, when implementing XtreamFS we advise users to select an appropriate level of logging (preferably verbose) to ensure that an evidence trail can be followed at a later date.

Examination and analysis

The “object_dir” directory contains a number of subdirectories and metadata files which relate to the individual stripes stored by the OSD. A simple directory listing on an OSD that is currently storing data will reveal a number of directories which have a name consisting of two hexadecimal characters (e.g. 7E or 22). In our experiments, we found that a further three two-hexadecimal character directories are located under the initial directory (e.g. a path of [obj_dir]/AA/BB/CC/DD/). In the last directory, a further directory exists with a name composed of the UUID of the volume followed by a colon followed by the file ID number (see Examination and analysis section) on that volume (e.g. 9be62ga4-21q1-1763-rq15-kap1021p4aq9:7). In this directory are the file stripes stored by the OSD. Configuration of the striping metadata on the volume or individual file at the MRC will determine the size and quantity of stripes stored in this directory. This can mean that all of the necessary stripes are stored within the selected directory or only a subset of the stripes may be available on this particular OSD.

This implies that individual file reconstruction is possible without relying upon the live XtreamFS environment to extract the data. However, in any reasonably sized
implementation of the environment, it would not be feasible for a practitioner to locate and reconstruct files of interest without the metadata collected as discussed. Many forensic searches are based on keywords and known file hashes, and these methods may not be useful when files are split into an arbitrary number of parts. In cases where cloud computing is involved, a practitioner may initially seek to collect all evidence from a particular user. However, it is clear that a practitioner would not be able to easily determine which files or stripes are owned by a particular user from the OSDs hexadecimal representation. To collect evidence at the low level that is commonly used with traditional devices (bitstream images), a practitioner must understand the technical method used by the storage platform to allocate file components.

In the case of XtreemFS, there are two key encodings that a forensic practitioner should understand in order to reconstruct files from a physical image of the OSD(s) — the directory path for a particular file or files (determined in part from analysis of the MRC data) and the stripe filenames.

The directory path that is created on the OSD(s) to store the file stripes is determined based upon the volume UUID and file ID. This file structure format is referred to as "HashStorageLayout" in the XtreemFS configuration directives. Based on our analysis and review of the XtreemFS source code, we were able to create an application that accepts a volume UUID and file ID (e.g. 9be62ga4-21q1-1763-rq15-kap1021p4aq9:7) and returns the path which stores the stripes on the OSD (e.g. /var/lib/xtreemfs/objs/D0/BE/B6/53/9be62ga4-21q1-1763-rq15-kap1021p4aq9:7/). This path is derived from the Java String "hashCode" function which then requires a number of other transformations before resulting in four pairs of hexadecimal characters (D0BE653 in the above example) that are then used to create the directory path.

The stripe filenames consists of three sixteen-hexadecimal character blocks concatenated together to form a 48-character filename (e.g. 00000000000000000000000000000000). Reviewing the source code, the HashStorageLayout createFileName function provides the following hints, the first block is referred to as "objNo", the second as "objVersion" and the third as "checksum". objNo appears to refer to the stripe sequence number, and in our experiments, we found the files must be reconstructed in the correct order based upon these object (stripe) sequence numbers. The decimal sequence number is encoded using a hexadecimal representation of a "long" type number (i.e. an unsigned integer in base 16) and reversed before being buffered to 16 characters by appending "0"s. For example, while stripe two is represented as "0000000000000000" (zero represents the first stripe as expected), stripe eleven is represented as "A000000000000000", stripe sixteen is represented as "F0000000000000000" and stripe seventeen is represented as "010000000000000000" (note the reversed ordering). In our experiments, we were able to successfully manually reconstruct a number of small and medium sized files proving the concept that manual reconstruction is possible if the practitioner has sufficient knowledge of the operation of the environment.

The last sequence of sixteen hexadecimal characters represents the checksum of the object (individual file stripe) when enabled. Checksums are not enabled by default and are enabled on a per OSD basis. Considering the importance of file integrity to forensic investigations, we elected to enable checksums on one of our OSDs to gain a better understanding of their operation in the XtreemFS environment. We tested both the Adler32 (default) and CRC32 checksums. While the OSD configuration file indicates that MD5 and SHA-1 are also supported, with our experiment environment configuration, we found that when these checksums were enabled we were unable to assign objects to these OSDs. As such we were unable to test the MD5 and SHA-1 checksums. Similarly to the other parts of the object identifier, the checksum is represented in hexadecimal and reversed. Other than this representation format difference, in our experiments the checksums for the objects (calculated using other tools) matched the checksums recorded in the object IDs. As such a practitioner could potentially make use of this functionality to ensure that logical copies of the objects are error free or to determine if objects have been modified after they have been stored (although presumably if the file data can be modified by an attacker the checksum can also be modified).

XtreemFS client

The XtreemFS client applications can provide significant utility to a forensic practitioner seeking to extract evidence from the distributed filesystem environment. It must be acknowledged that in a number of situations, the manual file reconstruction methods discussed above would be necessary. However, it may also be possible for a practitioner to use the XtreemFS client tools in a more automated fashion to collect the files from the environment. The various types of metadata collected can then be used, combined with the files collected, to create a relatively comprehensive (logical) representation of the data stored by a particular user in the environment.

If a practitioner has access to the mounted filesystems on a client either live (using commands such as mount) or based upon logged data or memory captures, this information can be used to simplify the location of the storage nodes in the environment (e.g. by identifying the DIR instance in an XtreemFS environment). In the standard mounted filesystem listing on a Linux system the following basic structure is used:

```
[mounted device] on [mountpoint] type [type] ([options])
```

e.g. /dev/sdal on /mnt/hdd type ext4 (rw, nosuid, noatime)

XtreemFS mount-points appeared as follows in our experiments:

```
xtreemfs@dir.local/volumeName on /mnt/volumeName type fuse (rw, nosuid, nodev, noatime)
```
An example mount command for an XtreemFS volume (named “volumeName” with a DIR hostname of “dir.local” to a local mountpoint of “/mnt/volumeName”) is as follows:

```bash
mount.xtreemfs dir.local/volumeName /mnt/volumeName
```

A number of artefacts of interest to a forensic practitioner can be determined from this listing. Firstly, the string “xtreemfs@” as part of the mounted device indicates that this is an XtreemFS connection. If a practitioner is searching a node for a connection to XtreemFS, this is one obvious method of verifying the connection. Secondly, “dir.local” is the hostname for the DIR service in the XtreemFS environment. This allows a practitioner to (logically) locate the DIR as required at the commencement of the forensic process. After the DIR hostname the volume name on the XtreemFS instance is provided, this potentially allows a practitioner to narrow their focus to the volume in question.

If a practitioner wishes to mount the volume using a locally connected machine (e.g. an acquisition laptop), they will need to use the “mount.xtreemfs” tool. If SSL and X509 permissions are not enabled, this can be as simple as “mount.xtreemfs [DIR hostname]/[volume name] [{local mount-point}]. However when SSL and X509 are enabled, the practitioner will need to have access to a certificate (as discussed in Evidence source identification and preservation section) and its passphrase as they are used with both the “–pkcs12-file-path” (pointing to the certificate file) and “–pkcs12-passphrase” (providing the passphrase) parameters with the mount command.

If a practitioner has access to a client with a mounted XtreemFS volume, they can use the “xtfsutil” utility to interrogate the volume for XtreemFS specific information. xtfsutil is a powerful tool for both gathering information from and managing XtreemFS volumes. In its most basic form, the command is run with only one parameter: the path to a mount-point or a file or directory under that mount-point. When used in this manner, xtfsutil provides a range of information including items of interest for three types of objects, a volume mount-point, a directory and a file (see Table 3):  

**Summary**

The section summarises the process a forensic practitioner may follow to conduct a successful forensic investigation of an XtreemFS instance. It should be read in conjunction with (and is adapted from) Collecting evidence from a distributed filesystem — a process section, which provides a generic process for evidence collection from distributed filesystems. Collecting evidence from the DIR, MRC and OSD components will require varying levels of access depending on the type of data and acquisition method the practitioner selects. Volatile data such as component status pages will require access to a web browser on the host or access to the (V)LAN on which the status pages are hosted and may require an administrator password (if enabled). Access to non-volatile data such as databases and object stripes may require root or administrator access on the XtreemFS hosts (depending on configuration) if a practitioner is seeking to collect them while the OS is running. However, if a practitioner cannot

---

**Table 3**

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Applicable type(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XtreemFS file Id</td>
<td>All</td>
<td>One of the most important identifiers (discussed further in Examination and analysis section) for locating files on OSDs. A unique file/directory identifier consisting of the volume UUID and file/directory number.</td>
</tr>
<tr>
<td>“XtreemFS URL”</td>
<td>Volume, Directory</td>
<td>The URL used to connect to the volume, including the protocol, hostname/IP address of the DIR, port number and volume name in the following format: [protocol://][hostname://][port number://][volume name] [directory name where appropriate]. e.g. pbpcs://DIR:32638/xtfsvolume/dirname</td>
</tr>
<tr>
<td>Owner</td>
<td>All</td>
<td>The name of the POSIX user or certificate (depending on authentication type enabled) which owns the item.</td>
</tr>
<tr>
<td>Group</td>
<td>All</td>
<td>The name of the POSIX group which owns the item (when X509 is being used, this is derived from the OU listed in the certificate).</td>
</tr>
<tr>
<td>Type</td>
<td>All</td>
<td>Textual representation of the object type, e.g. volume, directory and file.</td>
</tr>
<tr>
<td>Free/Used Space</td>
<td>Volume</td>
<td>Free space and used space on the volume delimited by a slash.</td>
</tr>
<tr>
<td>Num. Files/Dirs</td>
<td>Volume</td>
<td>Number of files and directories on the volume delimited by a slash.</td>
</tr>
<tr>
<td>Access Control p.</td>
<td>Volume</td>
<td>The access control policy applied to the volume (e.g. POSIX).</td>
</tr>
<tr>
<td>Replica Selection p.</td>
<td>Volume</td>
<td>Textual representation of the OSD selection policy (“default” in our experiments).</td>
</tr>
<tr>
<td>Default Striping p.</td>
<td>Directory</td>
<td>Textual representation of the volumes default striping policy (as discussed in Examination and analysis section).</td>
</tr>
<tr>
<td>Default Repl. p.</td>
<td>Directory</td>
<td>A textual (“yes” or “no”) representation of whether snapshots are enabled on this volume.</td>
</tr>
<tr>
<td>Snapshots enabled</td>
<td>Volume, Directory</td>
<td>A list of UID, IP address and port numbers for the currently selectable OSDs for this volume.</td>
</tr>
<tr>
<td>Selectable OSDs</td>
<td>Volume</td>
<td>The number of replicas (e.g. Replica 1, Replica 2, ..., and Replica N) with a number of sub values. These include the “Striping policy” which is a textual representation of the striping policy applicable to this individual file (that may be different from the default striping policy used on the directory or volume). The OSD(s) storing objects for each of the replicas is also listed (e.g. OSD 1, OSD 2, ..., and OSD N) with the UUID, IP address and port noted.</td>
</tr>
<tr>
<td>Replicas</td>
<td>File</td>
<td></td>
</tr>
</tbody>
</table>

---

gain access to an administrative account on the host, then
the techniques discussed in this paper should allow them
to manually reconstruct files of interest.

It must be noted, however, that this is a somewhat time
consuming practice and, as such, practitioners should take
every opportunity to collect the majority of files from the
distributed filesystem using a client connected to its
mounted volumes.

1. Identify XtremFS usage — The initial steps a practi-
tioner will undertake when investigating an XtremFS
instance will depend on the means by which they
identified the use of XtremFS. Most commonly we
expect this to be via their locating an XtremFS client or
being informed of the use of XtremFS by a system
administrator. Regardless of identification source, the
practitioner’s aim is to collect any logical data they have
access to and locate (physically and/or logically) the DIR.
In the case of an XtremFS client (with a currently
mounted remote filesystem), we highlight the method
by which the environment’s directory service (DIR) can
be located in XtremFS Client section. We also recom-
end that a practitioner take a logical copy of the data
that they are able to access via the currently mounted
volumes on the client. They should also check the client
mount logs, Linux ‘fstab’, etc. (where available) and
attempt to mount (read only) any volumes found (see
the mount example in XtremFS Client section).
Generally to access this information (and potentially to
access data on the mounted filesystems), they will
require root or administrator access to the client or they
could power off the client and conduct a standard
forensic analysis on the client device that may reveal
persistent data on mounted filesystems.
If the practitioner is satisfied with the logical data
which they have collected from the client and do not
wish to pursue further collection opportunities (which
require physical access to the XtremFS hosts or
cooperation by the system administrators), they may
choose to end the process at this point. However, if a
practitioner is looking to collect the maximum poten-
tial evidential data possible, they will need to proceed
with the next step of the process.
If a practitioner is commencing their investigation by
discussing the environment with a system administrator
(who mentions the existence of XtremFS), they should
ask where the physical environment hosts (e.g. servers
and storage systems) are located and should also request
any authentication (e.g. X.509) certificates/passphrases,
etc. and, if possible, to be setup on a client connected to
the XtremFS environment with a “superuser” key (as
discussed in Evidence source identification and
preservation and XtremFS Client sections). A practi-
tioner can then mount the volumes (as read only) using
the XtremFS client and collect logical data from the
volumes (and can use the DIR status page to ensure they
are connected to all hosted volumes).
Another potential identification opportunity we envi-
sion is via network broadcast interception (searching
for the network artefacts, such as “autodiscover”
requests, discussed in Evidence source identification
and preservation section) or sending broadcasts
(using the “autodiscover” method). However, for many
cases, this seems less feasible as an initial identification
method. This can be used to logically locate (preferably)
the DIR or other XtremFS components that can then
be used to locate the DIR.

2. Collect and Examine DIR Information — Once the DIR
has been logically or physically located, the practitioner
should collect volatile data from the DIR status page
(from a web browser on the host or via its (V)LAN). They
should also obtain root access to the host (via a remote
or local shell) and collect the non-volatile artefacts
outlined in Collection & examination and analysis sec-
tion. If they have physical access, they could also collect
the virtual machine disk or physical disks hosting the
DIR to obtain the non-volatile artefacts.
The collected artefacts from the DIR server should be
examined to determine the logical and/or physical
location of the MRC and OSD hosts in the XtremFS
environment. This will allow the practitioner to collect
and analyse the filesystem metadata and select files of
interest to ensure their stripes are collected from the
relevant OSDs.

3. Collect and Examine MRC Information — Collecting
the MRC metadata database is the next step after the
components have been identified and located using the
DIR metadata. Physically locating the MRC host should
be possible using existing techniques (e.g. datacentre
switch/routing examination) once the MRC IP address
has been determined. Collecting the database files
directly will generally require root access to the MRC
host (either console or remotely). It may be possible to
use the “xtfs_mrcdbtool” from a standard user account
or from another PC on the XtremFS (V)LAN. However, it
will not be possible if authentication is enabled unless
the practitioner possesses the keys and/or admin pass-
phrase. If the practitioner has root access to the host
then the database constituent files can be obtained (as
described in Collection section). However, once root
access has been obtained, the practitioner may be able
to collect the required authentication keys/passphrase
(as described in the MRC configuration file) to use the
“xtfs_mrcdbtool” which will simplify the collection.
Once the metadata database has been obtained, it
should be analysed to determine which volumes/di-
rectories/files are of interest and have not yet been
collected (via the client) or merit further investigation.
The artefacts that are available to make this determi-
nation are discussed in Examination and analysis sec-
tion and summarised in Table 1. The OSD GUIDs should
be noted and cross-referenced with the logical OSD
location (IP address) data collected from the DIR to
determine which OSDs need to be collected.

4. Collect, Examine and Analyse OSD Information —
Using the OSD identification data obtained from the DIR
(logical location) and MRC (stripes of interest), a prac-
titioner can begin collection of the relevant OSD data.
The primary items for collection for an OSD are the
constituent stripes that form individual files on the
filesystem. These stripes can be logically collected (e.g. copied) from the running host operating system’s underlying filesystem (e.g. ext4) if the practitioner has local or remote root access to the OSD host. However, if a practitioner is attempting to conduct a more in-depth analysis, they may wish to collect forensic images of the physical disks that support the host’s underlying filesystem. This process would obviously require physical access. However, it does not rely upon the practitioner having access to any operating system level credentials.

The process for analysing and ultimately reconstructing the filesystem stripes into files is discussed in Examination and analysis section. Checksums are also discussed in this section, and if enabled, provide a useful method for practitioners to determine that their copying process for file stripes is maintaining file integrity. Fig. 1 provides a graphical abstract of the steps outlined above. Table 2 also provides a summary of the types of stored metadata artefacts that are referenced in Fig. 1.
Collecting evidence from a distributed filesystem – a process

This research demonstrates that a process must be followed to ensure the collection of data and metadata to the furthest possible extent from a distributed filesystem environment. If a practitioner followed existing practice and attempted to acquire a bitstream image of the storage devices (in this case the OSDs), it is clear that a large amount of metadata (available at the MRC) would be missed. Metadata stored by the DIR may also be integral as part of evidence collection or environment reconstruction. For example, the practitioner may experience difficulty in locating all of the data storage devices in a medium to large sized installation without referring to the central directory source for the distributed filesystem (the DIR in this case).

With these considerations in mind, we propose the following process for collecting evidence from a distributed filesystem. This process contains elements which fall within the stages of the cloud forensics framework (Martini and Choo, 2012) outlined in Cloud forensics framework section. The process consists of the following three elements:

1. **Directory Services**: Locate directory services, collect data and examine the data to determine the filesystem components in use and their logical locations.
2. **Metadata Storage**: Using directory services data locate the metadata storage server, collect the metadata and examine it to determine which files/directories/volumes/etc. may be of forensic interest.
3. **Data Storage**: Using the environment metadata collected from the directory service and the selected target information obtained from the metadata storage, collect relevant data storage devices or logical datasets (e.g. stripes, filesystems) and reconstruct using the collected metadata.

Each of the process elements is discussed in detail below (also see Fig. 2).

Directory services

The directory services component of a distributed filesystem maintains metadata on the nodes in the filesystem. This may be implemented in a number of ways, for example, in the XtreemFS experiments discussed in Findings section, directory services were centralised to one (or a specific number of) DIR server(s). In other architectures, this could be a more distributed model where each node stores directory information or even a peer-to-peer model where limited directory information is stored with each node.

Regardless of the technical implementation, a practitioner should seek to gather the environment’s directory metadata as the first step. This would allow them to gain a better understanding of the technical operation and node logical location (e.g. IP address) in the distributed filesystem. Identification of these components will most likely be via assistance from a system administrator or via analysis of a cloud node (e.g. VM host server) or filesystem client. Once the directory information has been located, it should be preserved and collected. The specific methods required for these steps will depend on the technical implementation of the distributed filesystem.

Analysis of the directory data should consist of decoding (where necessary) and documenting all relevant details for each of the components/nodes in the distributed filesystem. Items of interest include logical location (e.g. IP address), type (e.g. metadata or data store), capacity, unique identifiers, et cetera. This information will be useful in reporting in terms of documentation, chain of custody requirements and provenance.

Metadata storage

Using the information gathered from the directory services the practitioner should now be able to identify the sources of metadata and potentially have an idea of the types of metadata available in the environment. This metadata may be stored with the data as part of the data storage component. However, it is likely in a distributed system that some or all metadata will be stored separately in a centralised or decentralised database. Once the directory information has been located it should be preserved and collected before being examined and analysed.

Examination of metadata will likely involve reducing the size of the identification and collection stages of the data storage stage of the process. The exact method used will depend on the case. However, a common reduction method would be to focus on metadata that indicates a data owner who is a suspect or related in some way to a suspect. In the XtreemFS example, this may be possible via the “owner” metadata which can indicate which POSIX user or certificate holder has access to the volume or has stored an individual file in the volume. In some cases this information is not reliable (e.g. if the default authentication provider is used in XtreemFS) or is unavailable (e.g. when permissions are stored in the application which the distributed filesystem supports). In this case the practitioner may need to rely upon permissions data available at other levels.

Other types of metadata that may be of interest include temporal data for both reporting and confirming provenance. File hashes are also useful if available for similar reasons and to allow the practitioner to search the filesystem environment for various files of known interest by hash.

Data storage

Once the necessary data has been examined and analysed from both the directory services and metadata store, the practitioner should have the required information to locate and reconstruct the files distributed throughout the filesystem. Similarly to the process for metadata location, the practitioner should be able to use data gathered from the directory service to identify the nodes which store data in the environment. Using the information collected so far,
as part of the process, the practitioner can now determine if it would be advantageous to complete a physical collection (i.e. collection of a bitstream image from the physical storage devices) of the data storage nodes or attempt to use the information gathered from systems administrators, client nodes or compute nodes in a cloud environment to connect to the filesystem and collect the evidential data logically (i.e. collected using filesystem client applications). In either event, the practitioner should ensure that they follow forensic preservation techniques (write blocking, etc.) as they would with a regular data collection. The practitioner should also ensure they collect any additional metadata which may be stored with the data objects (e.g. local filesystem temporal and permissions data) and on the data nodes (e.g. log data).

If the practitioner has undertaken a logical collection, they should be able to match the metadata and files collected to produce a complete representation of the data of interest which has been stored. If, however, a practitioner chooses to undertake a physical collection, they will most likely need to reconcile and reconstruct the individual data objects into the original data stored by the user. This is facilitated by the data collected at the metadata and, possibly, the directory services stages. An in-depth technical understanding of the distributed filesystem is likely required to reconstruct the files in a precise manner which would be required as part of a forensic investigation. These details may be discussed in the literature (in sources like this paper) but a practitioner is advised to also conduct their own experiments to enhance their own understanding and avoid any potential issues with software version changes.

**Conclusion**

With the increasing digitalisation of data and use of services such as cloud computing to process, store and disseminate big data, there will be more opportunities for exploitation of large datasets (e.g. in corporate or state-sponsored espionage) and consequently, the continued development of the digital forensic discipline is more important than ever. An effective investigative process is one that follows well-researched and documented processes, which allow digital forensic practitioners to be able to identify and preserve, collect, examine and analyse electronically stored information from information communication technologies that would be admissible in a court of law (Butler and Choo, 2013; Quick et al., 2014).

In this paper, we conducted an in-depth forensic investigation of XtreemFS, a distributed filesystem that is commonly implemented in cloud computing environments. Findings from our study contributed to a detailed understanding of the both the technical and process issues regarding collection of electronic evidence from distributed filesystems generally. Based on our findings, we put forth the following technical insights:

- Directory services provide practitioners with an overview of the internal setup of the distributed filesystem. This allows them to gain the necessary understanding of the environment configuration (i.e. both physical and logical) to undertake the collection of evidential data.
- Metadata storage holds critical data that facilitates the collection of valid and complete evidential data. Examples include temporal data which is useful in

![Cloud Forensics Framework](https://example.com/cloud-forensics-framework.png)

**Our Proposed Distributed Filesystem Process**

- **Locate directory services** – see Section 4.1 (example: Section 3.1)
- **Locate metadata storage** – see Section 4.2 (example: Section 3.2)
- **Locate data storage** – see Section 4.3 (example: Section 3.3)
- **Collect Volatile, Non-volatile and Network Data**
- **Examine directory services to locate filesystem components (metadata and storage)**
- **Examine metadata to locate data of potential evidential interest**
- **Use data collected from metadata and directory services to reconstruct files**

**Fig. 2.** Our proposed distributed filesystem forensic process.
reporting and determination of provenance in a forensic context.

- Data storage allows the practitioner to reconstruct the files of interest stored in the distributed filesystem using both environment data and metadata. This results in a complete representation of the stored data of interest.

This research also highlighted the importance of a forensically sound process such as our proposed distributed filesystem forensic process (see Fig. 2) in order to provide clear guidance to digital forensic practitioners in their investigation from evidence source identification and preservation to collection of volatile, non-volatile and network data to examining and analysing the preserved data and reporting and presenting in a court of law.

Future work includes validating our framework and the proposed process with other similar distributed filesystem products such as GlusterFS, FhGFS and Ceph. Another aspect of future work would be to develop forensic processes for cloud/distributed filesystems where APIs can be used for object storage and retrieval (e.g. Amazon S3) using a similar approach to the one presented in Martini and Choo (2014).

Acknowledgements

The first author is supported by both the University of South Australia and the Defence Systems Innovation Centre. The views and opinions expressed in this article are those of the authors alone and not the organisations with whom the authors are or have been associated/supported.

References


Federici C. Cloud data imager: a unified answer to remote acquisition of cloud storage areas. Digit Invest 2014;11(1):30–42.


