Concept of a Server Based Open Source Backup Process with an Emphasis on IT Security

Bachelor's Thesis

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Software and Information Engineering

by

Florian Pritz
Registration Number 1125452

elaborated at the
Institute of Computer Aided Automation
Research Group for Industrial Software
to the Faculty of Informatics
at the TU Wien

Advisor: Thomas Grechenig
Assistance: Florian Fankhauser

Vienna, March 12, 2016

Kurzfassung

Mehr und mehr Menschen sowie Firmen hängen immer stärker von IT-Infrastrukturen und den darin verarbeiteten Daten ab. Diese Abhängigkeit wird oft erst sichtbar, wenn die IT nicht funktioniert. Dies kann beispielsweise durch Hardware-Ausfälle, aber auch durch Angriffe auf die IT-Sicherheit passieren.

Um die Verfügbarkeit dennoch auf einem ausreichendem Niveau halten zu können, sind heutzutage unterschiedliche Lösungen verfügbar. Eine dieser Lösungen ist die Durchführung von Backups. Allerdings werden oft keine, oder nur inkorrekte und somit nutzlose, Datensicherungen erstellt.

Insbesondere für unerfahrene Nutzer kann es besonders schwierig sein eine Entscheidung zu treffen, welche Software genutzt werden soll um Sicherungen zu erzeugen. Es ist für sie eventuell nahezu unmöglich mehrere Produkte korrekt miteinander zu vergleichen und die jeweiligen potentiellen Schwächen in der Art und Weise, in der sie Sicherungen erstellen und speichern, zu erkennen. Solche Vergleiche werden besonders durch die vielen Details erschwert, die beachtet werden müssen. Diese werden aber nicht immer von den Herstellern der Produkte erwähnt. Ohne hinreichendes Hintergrundwissen ist es schwierig eine gute, und vor allem eine informierte, Entscheidung zu treffen.

Diese Arbeit enthält einen umfassenden Überblick über die unterschiedlichen Aspekte der sicheren Erstellung und Verwaltung von Backups. Neben theoretischen Grundlagen-Aspekten von Backups wird gezeigt wie ein funktionierendes Sicherheitskonzept für ein kleines Netzwerk erstellt werden kann.

Diese Arbeit beschreibt zuerst die Grundlagen von IT-Sicherheit sowie die Grundlagen von Sicherungen indem existierende Literatur und Forschung evaluiert und komibiniert wird. Nachfolgend werden die notwendigen Teile zur Erstellung eines Sicherungskonzeptes behandelt. Abschließend werden einige Beispiele für Software die zur Erstellung von Sicherungen genutzt werden kann dargestellt und es wird ein Beispiel eines Sicherungskonzeptes für ein kleines Netzwerk erstellt.

Schlüsselwörter

Datensicherung, Sicherungskonzept, IT-Sicherheit

Abstract

Increasingly more people and companies heavily rely on IT infrastructures and the data they process. This dependency often only becomes visible once the IT no longer works. Examples for this include hardware failure as well as attacks on IT Security.

Nowadays, there are multiple solutions to to keep availability sufficiently high despite these problems. One of these solutions is the creation of backups. However, many systems are not backed up properly or, even worse, not at all.

It can be difficult, especially for an inexperienced user, to determine which software to use to create them. It may be near impossible for them to accurately compare different products with each other and to determine potential weaknesses in the way in which they create and store their backups. This is especially difficult because there are many nuances that need to be considered and that may not be explicitly mentioned by software vendors which means that without sufficient knowledge of the field it may be impossible to make a good - and certainly informed - choice.

This work contains a comprehensive overview of different aspects of securely creating and managing backups. Alongside theoretical basics of backups, it also shows how a working backup concept, that is suitable for use in small networks, can be created.

This work first describes some of the basics of IT Security as well as the basics of backups by evaluating and combining existing literature and research. Later it examines the parts that need to be considered when a backup concept is created. Finally, some examples of backup software are discussed and an example concept for a small network is created.

Keywords

data backup, backup concept, security

Contents

List of Figures

1 Introduction

Not only increasingly more businesses but also private homes heavily rely on their IT infrastructure. IT systems can fail for numerous reasons, including attacks, user mistakes or natural disasters, but also simple hardware or software failure. Garfinkel, Spafford, and Schwartz[14] illustrate that in such cases, backups can be necessary to restore operations since hardware can be bought again, but the data stored on that hardware, like customer data, custom developed software or holiday pictures, is probably lost. Keeping up to date backups is, therefore, important to almost everyone who uses computer systems yet the field is rather broad and it can be easy to miss important details when making backups for the first time.

1.1 Problem

Especially in a business environment, losing data and not being able to restore it from a backup can have a serious impact. Customers may lose confidence in the company if they learn that the company lost their address or other data. Merchants may lose their order history and be unable to ship requested products while sales personnel may be demoralised if the hard work they put into acquiring these sales is lost. Finally, there is also the issue of time and money spent to acquire or create the data that is lost. This obviously also applies to private computers that store pictures, recipes, configuration for many programs, save files of video games, passwords and similar unique information. Such data has been created during countless hours of work and it would be rather horrifying to lose it all at once.

Finding a good backup solution is complicated because there is a lot of software available and there are little to no comprehensive overviews of actual backup concepts so it is time consuming to get all the necessary information and it is easy to overlook important nuances. What is likely to be the most overlooked, yet important issue in the scope of computer backups is to actually have a backup that can be used to restore data because as Preston[31] aptly quotes a friend: ‘No one cares if you can back up – only if you can recover.‘

Even if backups are made, there are many reasons why recovery might not be possible. Sometimes important data is simply missing from the backup, at other times the backup is corrupted or data has been backed up in an incorrect fashion and is thus unusable. This issue is especially devastating when databases are affected since those can easily become entirely unusable when the backup is created while the database is changing its files [31].

1.2 Motivation

There is a vast number of backup software available on the market, but for an inexperienced user it is often difficult to determine which software to use and more importantly how to use it. It is important to understand certain problems one faces when making backups and address them in a backup concept. The problems that are addressed include how to store backup data, how often and when to create backups and how long to keep them. A backup concept also explains how backup correctness and completeness are verified, how fast data can be restored and which threats the backup process is designed to guard against. Depending on the environment and the type of data

being backed up users will have different requirements for the backup. A collection of personal holiday pictures might be important enough to warrant keeping a couple of copies, but the time needed to restore likely will not matter as long as the data is available. On the other hand a backup of the systems in a super market likely needs to be easy to restore to minimise downtime and therefore the inability to sell goods. Such a backup may also contain a lot of private information about customers with membership cards, which should be protected properly.

In some cases, like the one that Preston[31] describes in his book, mistakes in the backup concept can have a disastrous outcome. In his example a database server broke down after it had been migrated from one machine to another just over six weeks prior. When he tried to restore its data he noticed that the latest backup had errors in the log file and thus searched for one that did not. When he found one without errors and tried to retrieve it, he ascertained that it had been created on the old server just over six weeks ago. The retention cycle for backups was exactly six weeks so the only good backup had been overwritten just days earlier. Fortunately they were able to recover the data from the broken server's disk directly, but such solutions are highly unreliable and thus having a good backup concept is key.

1.3 Goal

This thesis compares existing backup concepts and tries to provide a comprehensive overview of known concepts as well as their strengths and weaknesses. First the foundations of IT Security and the basics of backups are discussed. Relevant threats like hardware failure, user mistakes, natural disasters and attacks on IT systems are evaluated, possible mitigation or avoidance strategies for those are proposed and the potential damage is explored. Additionally, different aspects and requirements of the backup creation processed are examined in detail. This document should help the reader to be able to determine which concept best fits their requirements and which issues arise when using it. Finally a concept well suited for a small network with desktops and laptops is described.

2 Foundations of IT Security

Anderson[2] explains that IT Security is a term that is rather difficult to define. The most common definitions describe Confidentiality, Integrity and Availability (CIA), but security can also be defined as the balance between risk and controls that reduce that risk [2]. Anderson[2] defines it as "A well-informed sense of assurance that information risks and controls are in balance" which he claims is better suited for the corporate world because a CEO knows when he is well-informed and has sufficient assurance, but it is rather difficult to determine good metrics for a CIA approach.

2.1 IT Security

Mellado and Rosado[26] describe Information Systems Security (here called IT Security or Computer Security) as a process that tries to establish security policies and the resulting procedures and control elements over information assets. The goal of this process is to ensure the authenticity, availability, integrity and confidentiality (see Section 2.2 to Section 2.4) of the protected information. The CIA triple as defined by Guttman and Roback[18] is one of the first if not the first definitions of Computer Security while more recent work, like that of Parker[29], sometimes lists additional elements or even uses an entirely different definition.

For the purpose of this thesis, the original CIA definition is used and one based upon risk is also described because it applies better to a real world scenario. The CIA definition helps to find possible attack vectors, but does not associate them with any kind of weight. It considers all vectors equally important and strives to prevent all of them even though this is hardly always possible because in a real world setting they sometimes contradict themselves. For example using encryption reduces availability because the key needs to be available to read the data, yet it raises confidentiality and since both are security goals either decision to use or not to use encryption harms the other goal. With the addition of a risk based approach it is possible to determine the significance of each vector and figure out which ones threaten the system in question the most and how many of them need to be addressed to gain sufficient security.

2.2 Confidentiality

Confidentiality means that only certain users are allowed to access specific information [26]. For example in a sales environment only billing staff may access the billing history of a particular customer while shipping staff may only access the shipping address and the list of ordered goods.

This not only means that the complete set of information should be protected, but also individual pieces that may seem harmless on their own. These pieces could become harmful if combined with information from other sources [14]. With information like a telephone number, a postal address, the birthday or credit card number it might be easily possible to impersonate someone and convince support staff reveal additional information or perform restricted actions like resetting a password.

Confidentiality can be ensured by restricting access to information to certain authenticated users and by using cryptography to protect data when it is stored or sent elsewhere as well as using it to protect credentials when authenticating users.

Examples

Data Confidentiality Encryption like Advanced Encryption Standard (AES) can be used to ensure confidentiality for stored or transmitted data. Cryptographic algorithms like AES work by using substitution, transposition and mathematical functions on the message so that the normal content will read like gibberish after it has passed through the algorithm. It can then only be read again if the symmetric/private key is known and the operation is reversed [14].

Password Confidentiality Passwords should not be stored or sent in plain text because doing so would allow an attacker to either attack the server and extract the stored data or intercept the password during transit [14]. Dierks and Rescorla[10] explain that nowadays network traffic can be readily encrypted by using a library that implements the Transport Layer Security (TLS) standard which is a collection of cryptographic algorithms that "prevent eavesdropping, tampering, or message forgery" and thus it is easy to protect password information in transit. Nonetheless, it is important to protect stored credentials against leaks by only storing hashes of passwords rather than the plain text password itself because passwords are often reused [14]. Hashes do not protect against brute force attacks, but it is possible to select parameters in such a way that computing the hash is deliberately slowed down. Currently algorithms that allow this kind of behaviour include PBKDF2, Bcrypt and Scrypt. Chang et al.[7] explain that those algorithms use lots of CPU time or memory to make it harder to compute many hashes in parallel or generally in a short amount of time. By using such algorithms brute force attacks can be made more costly and thus they pose less of a threat, but the algorithm cannot protect users that use the same password since a hash function always outputs the same hash for the same input.

This determinism of a hash functions obviously leads to the problem that an attacker could look at the database dump and decide whether they want to attack more common or least common passwords first. To prevent an attacker from seeing identical hashes for identical passwords a salt can be added to the password before hashing it. The salt does not have to be kept secret but it should be unique for each stored password because the goal is to make every password salt combination unique so that even if many users share the same password the hashes saved in the database will all be different and an attacker cannot prioritise hashes that appear more often when performing a brute force attack. Using a salt also protects against rainbow tables that list every possible password with a certain length and its hash and can be used for fast lookups even of expensive hashes. When a salt is used they can still be calculated, but since the salt is different per password the table is only valid for one specific salt value. If the salt is long enough and thus the search space is large enough this ensures that it is infeasible to create and store a rainbow table for all possible password and salt combinations even with restricted length and character sets [14].

Physical Security of Storage Media Physical security of the storage media also needs to be considered especially when the media is transported and might be lost or stolen. The likely most secure way of protecting backup data's confidentiality is to encrypt it before it is stored on the media. Some devices provide encryption support in hardware, but nowadays it is easy and fast enough for most workloads to perform encryption in software and thus regardless which storage media is used it is possible to encrypt data. However, it is important that the decryption key is stored securely and available because if it is lost the encrypted data cannot be restored [14].

If data is not encrypted, then the only protection of the data on the media is its physical security since file system permissions or user passwords are only validated by the running system and not when accessing its backup data [14].

2.3 Integrity

The principle of integrity says that information will not be modified by third parties and in general that correctness and completeness is ensured [26]. In the case of a bank transaction this means that the information about the transaction will not change after its creation and that when requested at a later date one will receive the same information that was put in before.

However, in general the term integrity is not clearly defined. Some literature uses it to describe that information has to be timely, complete, accurate and consistent [18] while others use it in a broader sense that includes the fact that a system should protect its data and programs from being deleted or altered without permission of the owner [14]. Yet others also consider it a loss of integrity if a distributor sells a DVD with software from a publisher, but removes the name of that publisher from the DVD even though he does not alter the software itself [29].

For the purpose of this thesis data is considered to possess integrity if it has not been altered without authorisation, if it is complete and if it matches the data that was saved and is thus accurate. The same definition also applies to programs since these are just a special form of data.

Cloud Backup for MER Chang et al.[8] explain how NASA's Jet Propulsion Laboratory (JPL) created a backup concept using cloud storage to store encrypted data backups for the Mars Exploration Rover and be able to restore them quickly when necessary. They use the AES algorithm to ensure data confidentiality, but they do not report on whether or not they take measures to ensure data integrity apart from simple encryption. Their system keeps one local backup copy and one copy in the Amazon S3 cloud storage service.

To audit their remote backup JPL use MD5 to create a list of hashes of the files stored on S3 and compare that list with one created in parallel from the locally stored backup. They do not provide an explanation as to why they create two hashes in parallel rather than simply comparing the data used to create the hashes bit by bit. They claim that this audit process ensures data integrity of the remote backup, but according to Turner and Chen[41], MD5 is known to be weak and should therefore in general not be used when strong collision resistance is desired. Strong collision resistance is an expected property of a cryptographic hash function and means that it is infeasible to find a second item with the same hash as a given item but different content so e.g. an attacker cannot easily replace the backup's content and ensure that the hash of the replaced files matches that of the original ones. Given JPL employ AES encryption it is infeasible for an attacker to change the encrypted archive to produce files with matching hashes after decryption even though MD5 should not be relied upon for strong collision resistance.

Sasaki and Aoki[34] describe an attack on MD5 that can generate a pre-image with a complexity of $2^{123.4}$ while a brute force attack that tries every possible input has an average complexity of 2^{128} since the output of the MD5 function is a 128bit hash. Due to this reduction in attack complexity MD5 is considered weak. However, MD5 is good enough if the goal is to detect unintentional changes of data caused by e.g. transmission errors, but the purpose of use has to be stated clearly to determine if this is the case [41].

2.4 Availability

As the name suggests availability means that information is readily available when it is necessary [26]. This means that for example the backup data stored on a storage system will still be readable when it is necessary to restore the system months or years after it has been stored.

Garfinkel, Spafford, and Schwartz[14] state that the effects can be just as bad as if the information was deleted when information is either not available due to services being degraded or made unavailable without authorisation. This view is backed by Parker[29] who gives an example of a rejected programmer who removed the file name of an important file thus rendering bank staff unable to find and use its content even though only the name was removed, but the content was still there. He also highlights the importance of having backups and multiple ways to access the same content by using programs that can search for the content directly without requiring a file name. Furthermore he discusses mirrored file storage or nowadays more commonly called Redundant Array of Independent Disks (RAID) which is discussed in Section 3.3.1.

2.5 Threat Models

According to Hasan et al.[19] threat modelling describes the process of "organizing system threats and vulnerabilities into general classes to be addressed with known protection techniques". They state that storage systems can be attacked in a variety of different ways and system administrators often try to find possible vectors by performing general brainstorming or reacting to attacks, but such an approach is not reproducible and therefore "likely to leave large portions of the attack space unprotected". A much better solution is to break the system into smaller parts that can be more easily understood and then perform brainstorming of these components. Hasan et al.[19] provide two processes which can be used to create threat models for storage systems in a systematic and repeatable fashion and they also discuss identified attacks. Their first process is called the Confidentiality, Integrity, Availability, Authentication (CIAA) process because it organises the attacks into the four groups from which the name is derived. Confidentiality, integrity and availability are already defined in Sections 2.2, 2.3 and 2.4. Authentication means that the user or an automated system presents some form of credentials which allow another system to confirm it as a legitimate user and further use that information to permit specific actions. The second process is the Data Lifecycle Threat Model process which works by determining where data is being stored or transferred and evaluating the CIAA process for each of those steps separately. Their process is described in further detail in Section 3.5.

2.6 Risk

Security is a process and perfect security can never be achieved because only the risk of threats can be reduced, but it is impossible to reduce that risk to zero [14]. According to Anderson[2] risk is often difficult to measure even though Garfinkel, Spafford, and Schwartz[14] argue that numbers can be obtained from industry organisations or insurance companies, but they also note that doing so is difficult work. If numbers are available for example because the event happens often enough and occurrences are recorded by the organisation then probabilities can be calculated. The probability multiplied by the expected loss gives the annual loss expectancy (ALE) which can be compared to the cost of the protection to determine whether or not the benefit is worth the cost [2]. However, such values should be used with care since the uncertainty for some estimations can be pretty high and therefore predictions may be incorrect [2].

It is important to note that since perfect security is impossible there are always trade-offs to be made. How those look in practice is something that everyone has to figure out for themselves, but the definition by Anderson[2] given at the beginning of Section 2 provides a good guideline for such decisions.

2.7 Security Aspects of Open Source

Open Source Software (OSS) describes software for which the source code is available for anyone to read and probably the most prominent example of OSS is the Linux operating system kernel. Linux is commonly called "Open Source", although it is important to note that "Open Source" on its own only really means that the source code is available, not that it may be modified or even redistributed. To avoid such ambiguity the term Free Libre Open Source Software (FLOSS) can be used.

The value of FLOSS lies in its unique security properties. One might think that it is easy for attackers to find weaknesses since the source code can be read by anyone, but according to Hoepman and Jacobs[20] this is put into perspective by the fact that at the same time it is harder for developers of the software to get away with bad project management or quality control than it is in closed source projects. If issues are found in open source software they can be fixed by anyone with the necessary programming knowledge while in closed source software only the producer has access to the code and is therefore the only one who can change it. This problem is further intensified by the issue that many bugs, but also security concerns, are either fixed after weeks or months or worse yet not at all [20].

Having code available to be read by anyone is similar to how cryptographic algorithms' security usually only relies on some form of secret like a key rather than the algorithm itself being secret. Former military systems also worked by restricting information to as few people as possible and thus used the "security through obscurity" principle and their ciphers were "not particularly difficult to decipher" [20].

When using FLOSS users can, if they so desire, evaluate the security of the software themselves or hire someone they trust to evaluate it for them. Such evaluation is crucial to reduce the risk of back doors in the software and for example in November 2003 an attempt to insert a back door into the Linux kernel was thwarted due to code review. In a closed source project it would be uncertain if the software developer is really trustworthy enough to not put in back doors and even if they are to be trusted the user cannot view the code and thus cannot verify the claim [20].

The quality of FLOSS code is not perfect since it is also written by normal people just like code for a closed source project and this is made worse by the fact that many open source projects happily take any help they can get and quality is often not controlled. However, since the code is available to be read, the user can take a look themselves and it is generally accepted that sloppy code is untrustworthy and should be avoided. Of course the user is free to take over the project and make it better or pay someone else to do so. Similarly if the original author decides to stop working on it the user can take matters into their own hands and even if a project is not properly maintained critical patches can still be applied. All of this is generally not possible for closed source projects without support by the owner [20].

While it is certainly possible to review source code of open source projects there is no guarantee that this is being done by other people yet it is easy for anyone to assume someone else does it and thus move the responsibility away from oneself. While Hoepman and Jacobs[20] often talk about the possibility to view the code they do not discuss how often this actually happens.

3 Basics of Backups

This chapter describes the reasons why it is important to make backups and what types of backups like manually triggered full backups or automatically created incremental backups exist. In addition it also explains how the method used to create a backup influences for example the storage requirements, how long a particular backup is kept and the recovery procedure.

3.1 Reasons for Making Backups

According to Garfinkel, Spafford, and Schwartz[14] there are multiple reasons why backups should be made. Firstly, they obviously allow the user to recover their data when the system breaks down and all data stored on it is lost, but they can also be used to determine when an intruder changed data and if so which data was changed and how. This can be very important when analysing an intrusion since files can be compared and if the attacker changed anything those changes might help to shed some light on what the attacker viewed, deleted or what they were looking for in general as well as determining if the attacker planted a back door to easily gain access again once the initial attack vector has been closed.

In the case of simple hardware failure recovering from a backup is faster than rebuilding from scratch and when an organisation's number of disks grows, the probability for drive failures also grows and sometimes disks even fail shortly after being put into service, meaning that having backups early is very important [14].

The German Federal Office for Information Security (BSI) [16] outlines that having backups also allows to easily correct user mistakes like accidentally deleting the incorrect file or directory or mistakes by system staff which might delete an active account rather than an old one. Backups also come in handy in more unexpected cases like software failures where an application crashes and in the process corrupts important data like a spreadsheet.

Backups can also be used to determine how files changed if archival information is necessary, but has not been created by using some kind of version control system. Garfinkel, Spafford, and Schwartz[14] say that such information can be invaluable if one ever has to reconstruct history for a court case. Last but not least, backups protect against theft of hardware, natural disasters like fire or a flood and other disasters like exploding gas leaks or coffee spills [14].

Backups therefore allow to increase the availability of offered services while also increasing reliability of stored data in the sense that data is less likely to be lost entirely. It may be unavailable for some time, but it can be restored rather than being lost forever.

3.2 Types of Backups

There are multiple types of backups depending on when or why they are created and how much data they contain. Backups can be created right after installing the system, new software or after making important changes, but they can also be created monthly, daily or even more often [14].

Level-Zero Backup A level-zero backup is created right after installing a system for the first time and before anyone uses it. It contains every file and program installed on the system including the operating system. This type can be invaluable after a break-in when a hole is not fixed and the intruder manages to get back in. Then the backup can be restored and the hole fixed rather than having to reinstall from scratch once again. [14].

Full Backup As the name implies a full backup is a backup of the complete system just like a level-zero backup, but rather than being created only once it is created regularly. Such backups allow for fast and easy recovery since they contain all files, but they are large and thus require lots of storage space, especially if kept for extensive periods of time [14].

Incremental Backup An incremental backup stores the changes made since a particular event like the application of a vendor patch or the last backup. Such backups generally need less storage space than a full backup if large amounts of data remain the same, but they rely on the previous backups and thus form a chain. Recovery requires the chain to be intact and recovery speed will become slower the longer the chain. A similar type is the differential backup, which also only records changes, although it always starts off from the last full backup rather than forming a chain [14].

Preston[31] claims that in Windows a differential backup may not always build upon the latest full backup, but rather act like an incremental backup due to the archive bit of files being reset when an incremental backup is created. The archive bit is used by Windows to track which files were changed since the last full backup. He stresses that it is important to ensure how the vendor of the backup software that is being used defines those terms because products and administrators do not agree on general definitions.

3.3 Storage Location

There are many different options when choosing a storage location for backup data, but each comes with their own advantages and challenges. Local backups allow for quick and easy access while a centralised storage server may be simpler to manage and cloud storage may offer better availability guarantees.

3.3.1 Local Storage

The probably most obvious storage location for a backup is an additional hard disk. This disk can be internal to the system in a RAID setup, it can be internal and used for non-RAID backups or it can be external. Other external storage media include tapes and optical disks.

Regardless which storage media is used, it is important to migrate old backup data if the media is changed. At some point it might be impossible to find devices to read old tapes just like it was for NASA and MIT [14].

Internal Disks An additional internal disk that is used to store backup data is convenient because it is always available for use, but if the device is stolen or destroyed by a fire it will also be gone and therefore additional backups are necessary to protect against these cases. The ease of use in the case of accidentally deleted or overwritten files makes it a good first level of protection. A simpler alternative is to allocate an additional partition on an existing drive and use it to store backups. This

can be done if an additional disk is too costly or there is no space to add it, but it does not protect against failure of the disk.

RAID A RAID system is usually used to ensure reliability by using multiple disks and redundantly storing the data on them. However, such a system does not protect against user errors because actions affect all devices instantly and old data is not kept.

The simplest form is called RAID 0 and is actually not a redundant array as the name would suggest since the data stored on the array is stored only once. The goal of RAID 0 is to be able to use multiple disks with a single file system without having to add dedicated support for multiple disks into the file system's code itself and also to improve performance because all disks can be queried simultaneously for parts of their data.

Patterson, Gibson, and Katz[30] define RAID 1 as the simplest form that actually provides redundancy. It works by putting the exact same data on multiple disks so that as long as one disk of the array is available the data can still be used normally. Once a disk fails it can be replaced to restore full redundancy.

There are also more complicated schemes like RAID 5 which allows for one disk to fail, regardless how many disks are in the array, or RAID 6 which allows for up to two failures [30].

However, simple disk failure is not the only failure mode of hard drives so even a RAID system cannot fully protect the data. Even if disks worked 100% correct until they fail a RAID system only duplicates the data and protects against hardware failure and it does not protect against operator mistakes or deliberate deletion of data.

Elerath and Pecht[12] show that it is also important to note that undiscovered data corruption due to latent defects can cause serious trouble. This is especially dangerous once a disk in the array fails and its data has to be restored from other disks. Missing or incorrect data on those other disks can happen due to various reasons like bad media, inherent bit-errors, high-fly writes and corroding media. The data stored on a RAID array has to be regularly checked for such defects by scrubbing the array. Scrubbing means that all data from all disks is read and verified and if corrupted data is detected it is corrected using different copies.

External Disks An external disk provides a good yet rather simple form of backup. This is especially true if the disk is disconnected when it is not in use because then an intruder cannot access and delete the backup as long as they are detected and dealt with before the disk is reconnected. It may be beneficial to ensure the disk heads are properly parked when the disk is disconnected because otherwise they can cause damage during transport.

Tapes Certain tapes provide switches to disable writing so that once the backup is created and the switch is flipped it can no longer be accidentally erased. When using tapes the storage location has to be carefully selected because they are sensitive to magnetism and heat. If stored for extended periods of time the magnetic fields on the tape itself can also affect each other and corrupt the stored data. [14].

Optical Disks Optical disks are rather cheap and there are variants that cannot be overwritten once data has been burnt on them. On the other hand they are sensitive to heat and chemicals [14].

3.3.2 Traditional Client Server Architectures

In a traditional client server architecture there is one central server that stores all client's backups. Clients can push their backup data or the server can pull it from the clients. These methods come with their own challenges and security implications and are discussed in detail in Section 4.2.6. Depending on the requirements the server can be a single machine with a couple of disks or it can be a cluster of multiple machines that present their pool of disks as one large storage device. As long as those machines are not distributed across multiple places this work refers to the cluster as well as a single server as a central storage system. Distributed systems are covered in Sections 3.3.3 and 3.3.4.

Using a dedicated system has multiple advantages over local storage:

- The server is reachable via a network connection and can thus, rather easily, be placed far away while still keeping ease of use the same. This protects against any local incidents in a physical sense.
- It stores backups for all clients and storage media maintenance can be managed in a more centralised fashion which reduces overhead that would otherwise be incurred per client.
- Storage space can be shared across clients and thus be used more efficiently, especially if duplicate files are deduplicated. Refer to Section 4.2.5 for more details.

3.3.3 Cloud Storage

Cloud storage can be used in a similar fashion to a normal client server architecture by using a storage only plan or by using virtual machines in the cloud. Ling and Datta[24] point out that cloud services usually provide rather high availability guarantees, but such services can still be compromised by attackers.

While Preston[31] mostly refers to physical media, like disks or tapes, when talking about storage media security, there are similar issues with a client server architecture and with cloud storage. Just because backup data is sent via a network link and then stored electronically that does not mean that there is no storage media involved in the process. Data will be stored on disks, tapes or other types of media at the remote location and that media is just as vulnerable to incidents as any other. Since the media is likely not directly owned by the person or organisation that created the backup it may vanish at any moment. Preston[31] gives the example of the electronic vaulting vendor themselves burning down while Ling and Datta[24] argues that they may simply go bankrupt or out of business in which case data stored on the service is likely lost.

Preston[31] also notes that when using any kind of on-line backup storage the network link might allow for incremental backups to be created in a reasonable time frame, but it may not be possible to restore all the data as quickly due to the sheer volume of data that can accumulate. Some vendors provide local recovery appliances for those situations.

Zhang et al.[44] caution that virtual machines in the cloud are prone to side channel attacks which could compromise security by allowing users of other virtual machines that are running on the same host system to access data used by the user's machine.

3.3.4 P2P

Lillibridge et al.[23] developed a backup technique that does not use any particular storage service, but rather uses multiple client computers connected to the internet and backs up data in a

decentralised Peer-to-Peer (P2P) fashion. Each system contributes some of its storage space and in return receives the same amount of space on a different system. Since computers are not on-line all the time multiple such pairings are set up for each client and if the number of partners is large enough this ensures high reliability. To prevent abuse of the system by for example uploading data and throwing away any data one receives Lillibridge et al.[23] regularly challenge each partner to prove that they still possess the data. If such a challenge fails or the partner is off-line more often than was agreed on it is abandoned and a new one is selected. Their system uses erasure codes as described in Section 4.2.5 to reduce the storage overhead and yet keep redundancy high. Clients can freely choose the parameters of their erasure code and thereby the expected reliability of their data. They also note that for the reliability to be correct it is important that partners are selected randomly from a large pool and that they are not located close to oneself. While it is possible to use their software in a corporate environment and for example let all computers in an office back up each other any incidents such as viruses, hardware defects or a fire are likely to affect multiple computers either because they probably run similar hardware and software or are located close to each other. For optimal reliability partners should be in different physical locations, run different operating systems and in general be as diverse as possible.

Toka, Amico, and Michiardi[39] developed a similar system, however, while they use P2P storage as the final storage location they initially use cloud storage services to increase the backup performance and only later distribute the backup data among peers. Their solution performs similarly to a client server architecture when creating a backup yet it greatly reduces the storage cost at the cloud provider since data is only kept there for short periods of time.

Untrusted Storage Using encryption or erasure codes is a natural choice when untrustworthy and potentially insecure storage is used. Examples for such storage include storing backup data on websites that are indexed by search engines, like the example given by Traeger et al.[40], storing it in the cloud (see Section 3.3.3) or in P2P storage systems [24, 39]. Traeger et al.[40] describe a backup system that uses free web hosting services to store encrypted data. In addition to the simple storage of data, search engines will add the data to their index and even if the data is no longer available at its original location their cached copy can still be retrieved. The original data is combined into a Tape ARchiver (TAR) archive, encrypted and then encoded into text using uuencode. The resulting text is put on a web page together with a backup identifier that usually contains the original directory name and a time stamp. Using the identifier a script can query multiple search engines and then extract the encoded data, decode and decrypt it and unpack the archive. The backup can also be restored without the script by performing the search and extraction of the encoded data manually.

3.3.5 Other Concepts

Traeger et al.[40] also created a similar backup solution that uses free email services by sending the backup archive in multiple emails via Simple Mail Transfer Protocol (SMTP) and by downloading it directly from the mail server via Post Office Protocol 3 (POP3). To support larger backups multiple accounts with different providers can be created that each store parts of the complete backup. One of the benefits of this solution is that the backup is uploaded directly to the storage rather than waiting for a search engine to crawl and index it, however, management is more complex since SMTP and POP3 are used and data may have to be split into multiple message so that it fits within the limitations of the provider. The solution using search engines is simpler since they just need an HTML file and it also provides better redundancy because many different search engines will find and cache the file and as such it is very difficult for an attacker to remove all copies.

3.4 Security Aspects

Backups do provide some protection against natural disasters, hardware and software failure as well as operator mistakes, but while such issues can cause considerable damage they are, by definition, not systematic attacks and as such their scope is at least partially predictable. Theft and electronic break-in, however, may be conducted intentionally as part of a systematic attack. Contrary to nature, people are able to adapt and thus they expect their victim to have backups and can also target them directly if they so desire. It is therefore important to consider backups as a part of a general security strategy and discuss their relationship with IT Security.

The probably most obvious security aspect of a backup is confidentially (definition see Section 2.1) because the backup contains all important data. If it is not properly protected and stolen it is easy for an attacker to gain further access, but depending on the type of data there may also be legal consequences due to privacy concerns. An example for highly private data that might be contained in a backup is the 2014 celebrity photo hack discussed by Fallon[13]. Using encryption can dramatically reduce the risk of someone being able to gain access to the data inside the backup, but it is important to properly manage the decryption keys because if the key is forgotten or not available (availability is defined in Section 2.4) it is impossible to decrypt and thus use the backups.

Which type of encryption is used depends on the protection requirements of the data. For normal protection requirements AES with a 128bit key is sufficient, while more important data may require a 256bit key [16].

It is also important that encryption is not the only line of defence against misuse because then everything rests on the decryption key being kept secret and on the encryption algorithm being secure forever since an attacker could keep an encrypted copy of a backup and decrypt it years or decades later [14].

Integrity needs to be verified regularly because even if a backup is available and kept confidential it is of no use if its content is corrupted due to hardware failure or broken software. Discussion about how integrity can be verified can be found in Section 3.6.

Even if the integrity and confidentiality are assured an attacker can cause considerable harm if they control the availability of the backups. Assume the attacker operates a media storage facility that stores all off-site backup media for an organisation. The backups are properly encrypted and - when used - their content is verified using digital signatures so integrity and confidentiality are provided. However, the attacker sets fire to the organisation's office which contains the clients and the normal on-site backup. When the operator tries to pick up the off-site media they discover that the storage location has been emptied and their media is no longer there. There are of course multiple reasons why off-site media can be unavailable most of them being the same threats that urge the creation of the backup in the first place, like natural disasters and theft, but also bankruptcy and cooperation of the storage facility staff or service provider if on-line storage is used [24].

As explained in Section 3.1 it is also important to have backups in the event of an intrusion even if the systems continue to work just fine. The intruder might have modified crucial executables or configuration files and finding such changes can be difficult if a search has to be started from scratch. One can gain even more insights if untouched log files are available and thus these should also be part of the backup and ideally they should be backed up in real time to another system so an electronic intruder cannot simply erase them once they gain sufficient access to the attacked system [14].

Some storage solutions are also more vulnerable to certain incidents than other. For example, local storage media may get damaged along with the system it is supposed to help protect if it is kept close to that system. This is particularly true for additional internal disks or RAID systems since

those are stored in the same enclosure as the original system. External disks, tapes or optical media can easily be removed from the respective drives and can be stored farther away in an attempt to contain damage caused by localised incidents. Section 4.2.3 discusses this issue in further detail.

Yet even external media, notably external disks, cannot always provide adequate protection against undetected intruders on the original system. Depending on how much access the intruder possesses they may be able to disable or alter the backup process and in the case of external disks they may be able to delete or alter the backup data stored on the disk once the disk is connected to the system to create the next backup. Tapes may provide a switch that sets them into a read-only mode which reduces the risk of an attacker being able to change the content, but new backups might still not be created properly.

3.4.1 Traditional Client Server Architecture

Obviously centralised storage also intensifies some problems that are less severe in an environment where each computer has its own set of backup media. Depending on where the central storage system is located different advantages and drawbacks may apply and need to be evaluated. If the storage system is in the same building as the company it will likely not be able to protect against large fires, earth quakes that destroy the building or intruders in the building that gain physical access to the systems. However, it will likely offer very good network performance at a reasonable cost since local networking is cheaper than a fast internet uplink. Having a fast networked backup can be very valuable when large backups need to be restored quickly because no external media has to be fetched since the storage system has all the data available all the time. This also applies to Hierarchical storage management (HSM) systems since those do not require operator interaction when files are migrated to and from cheaper storage even if tapes or optical media are used because these are located in a media library that supports exchanging the media via electronic requests. Backups made with simpler tape drives or optical drives obviously do not provide that feature so manual interaction is required whenever a backup is created or restored [31].

If the storage system is located in a different building like a data centre that provides colocation services or where pre-built servers can be rented from, then localised incidents like a fire in the company building are obviously not a threat. Electronic attackers on the other hand may very well be one because if the backup is uploaded from the client then the attacker can gain access to the server when the client is compromised. Their access may be restricted to only the particular client's backup, but this data could be deleted or altered. This issue is further explored in Section 4.2.6. Different threats that apply to external storage services include simple Denial of Service (DoS) and direct attacks on the storage system. DoS attacks are discussed in Section 3.5.3. Direct attacks on the storage system include exploiting bugs which can lead to attackers crashing the service or gaining privileges and for example deleting backup data [31].

Another issue when placing the storage system in a remote data centre yet using it only for one's own needs is that of using encryption. If backup data is stored in plain text an intruder on the server can access that data and potentially use it to attack the clients themselves and thus gain control over the entire network. To combat this, backup data stored on the system may be encrypted, however, if data is encrypted on the server then an intruder on the server may be able to easily obtain the key, if they even need it at all. If data is encrypted on the client one runs into the issue that encryption ideally creates a random data stream and random data is almost impossible to compress. This can be resolved by compressing data before it is encrypted, but when creating backups for many clients a more efficient form of compression, called deduplication, can be used. Deduplication detects duplicate files or parts of files and stores them only once thus saving very large amounts of space when for example 100 client systems all run the same operating system and the same applications.

Since encrypted data is essentially random no duplicates can be found and deduplication cannot be applied which means no space can be saved [31].

Of course the storage system in a client server architecture does not have to be under one's control and space on such a system can be rented. When renting storage space there are multiple possibilities such a renting only the space and using some kind of file transfer protocol to access it directly or renting a dedicated machine that has access to the storage space. With a dedicated machine there are more options available as to how a backup is transferred from the client to the storage system and when and how it is encrypted. A dedicated machine allows for pulling backup data from the clients rather than them pushing it to the server and it can also be used to encrypt data before it is stored on the storage system. For information about pulling and pushing data refer to Section 4.2.6.

3.5 Threat Models

While risk management as described in Section 2.6 also deals with threats the difference to creating a threat model is that in a threat model it is not important what the probability of occurrence is as long as the threat is possible [19]. Myagmar, Lee, and Yurcik[28] explain that threat modelling works by determining all assets that need to be protected, identifying their access points and then finding threats that target those access points. They further stress the fact that to identify assets one has to understand the exact system that is being protected. It is thus impossible to provide a general exhaustive list of threats because no two systems are equal.

Systems can be characterised by creating data flow diagrams which show how information can get into the system and how it is processed [28]. This is similar to the Data Lifecycle Threat Model which Hasan et al.[19] use to identify threats to a storage system.

3.5.1 Client Security

There are many threats that impact client security that can be contained by having a good backup concept. Section 2.1 defines IT Security as the requirement of IT systems to provide CIA. Ensuring that client computers are available for use and respond as can be excepted to input provided to them is therefore a question of ensuring their IT Security. Those goals are threatened by many factors such as hardware and software failure, theft and electronic break-in as well as operator error and natural disasters [31].

The specialty of client computers is that they are often operated by users that are not trained enough to properly handle secure operation of their workstation. Reis, Barth, and Pizano[33] note that in some organisations, client computers are used to browse the internet in addition to normal business operations and many such users are running old versions of their browser and thus vulnerable to attacks via malicious web sites. According to Garfinkel, Spafford, and Schwartz[14] regular training and positives incentives to raise security awareness are important for maintaining secure operations.

3.5.2 Server Security (Storage)

In contrast to client systems servers are usually maintained by experienced IT staff and also generally not used to access random websites because they normally have very few functions to perform. They are, however, rarely used by an operator directly in front of the machine and as such they are especially vulnerable to physical incidents [14].

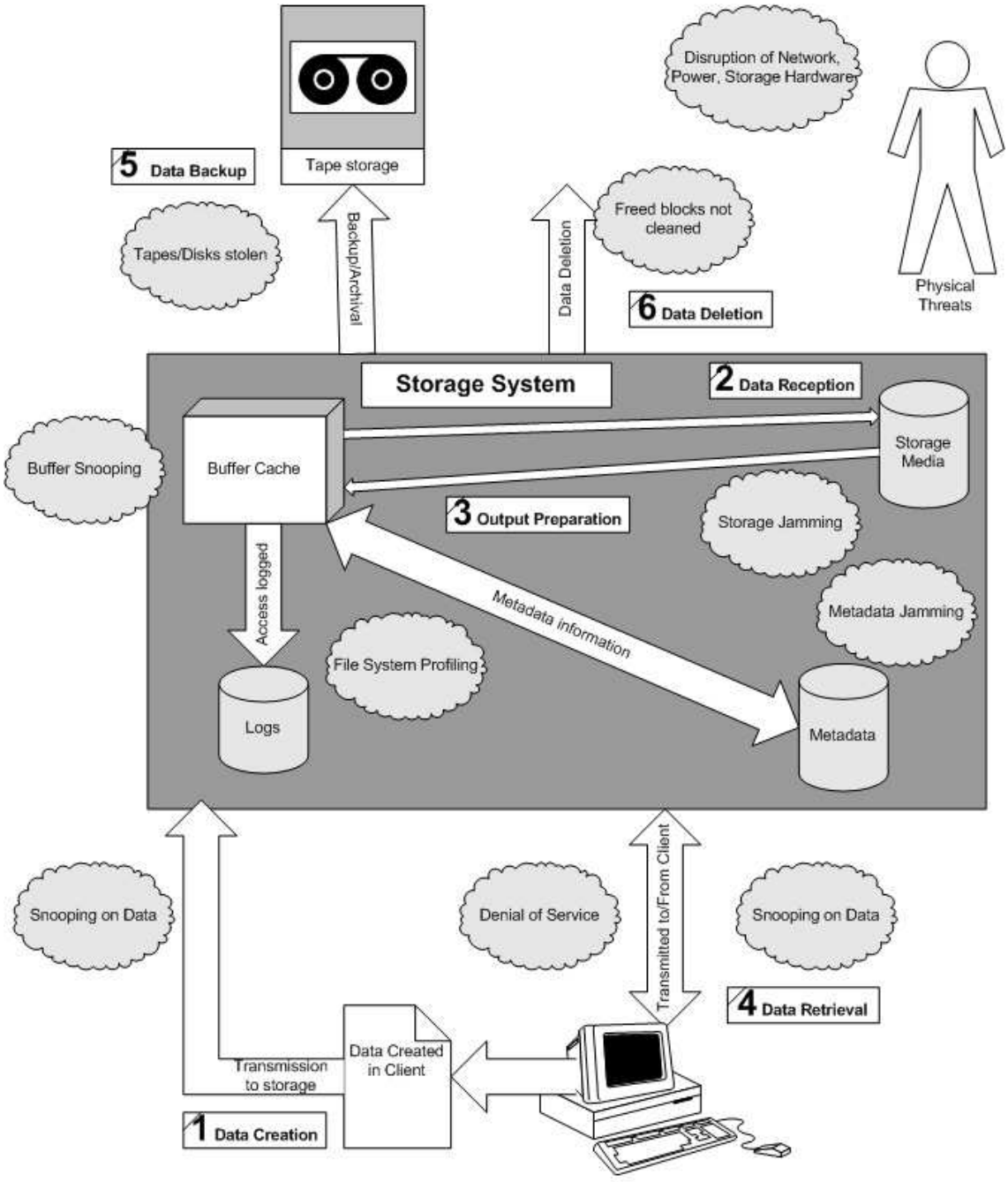

Figure 3.1: Attacks on a storage system identified by using the Data Lifecycle Threat Model. Source: Hasan et al.[19]

Hasan et al.[19] provide a discussion of attacks against storage systems they discovered using the Data Lifecycle Threat Model process. They organise the data flow into six stages from data creation to its deletion and then apply the CIAA process to each of those stages. The stages are data creation and reception by the server, output preparation and retrieval, data backup and deletion. Figure 3.1 provides an overview of the stages and for each stage additional assets that result from the use of the Information Lifecycle Management (ILM) practice are also shown. Hasan et al.[19] define ILM as a "practice to manage data from the moment it is created to the time it is no longer needed". Their implementation of ILM uses fast and expensive storage which is marked as “Buffer Cache” in the figure for files that are new or frequently accessed. Less important data is stored on slower and cheaper storage media marked “Storage Media”.

In the data creation stage (numbered 1 in Figure 3.1) the data is also implicitly transmitted to the server. They list attacks like sniffing and modification of data, disruption of communication and creation of data with stolen credentials. The next stage is data reception (2) which means that data arrives at the storage system, is written to the storage media and the activity log and buffer caches are allocated. An attacker may again sniff and modify the data written to the buffer cache, they may also exhaust available file handles and thereby block creation of new files on the storage media which makes it impossible to retrieve the data [19].

Later when data is retrieved, it first has to be prepared for output (3) by loading it into buffer caches. During this stage an attacker can monitor usage patterns by profiling the file system, they can hinder availability by for example attacking the power supply and they can use stolen credentials to retrieve selected data. This last attack also overlaps with the stage of actual data retrieval (4). Other attacks in the retrieval stage include sniffing and modification of transmitted data as well as theft of storage media [19].

In the last two stages data is backed up to tape (5) and finally deleted (6). During the backup stage attacks like theft of backup media, abuse of bugs in the backup software and attacks on the power supply or the network as well as masquerading as a trusted storage system that should also receive backup data are possible. During deletion it may be possible to snoop on deleted blocks since some regulations require proof of secure erasure. An attacker can also try to prevent deletion by modifying metadata, scramble data in the storage system to make it difficult or impossible to find or use a stolen identity to delete data prematurely [19].

In addition to the threats discussed above, Figure 3.1 contains some more that are not important to understand the concept and as such are not discussed here. Further information about these threats can be found in [19].

Depending on how the storage system operates, there are of course many more attack vectors and the list presented above is not exhaustive. Breaking down the system into smaller parts and then applying the CIAA process allows for better coverage than simply brain storming in general because there are many possibilities and it is easy to miss some of them when there is no additional structure that provides guidance.

3.5.3 Example Threats

This section discusses some selected examples of threats that need to be considered in threat models in an attempt to give a better understanding of what is possible. It is certainly incomplete and threat modelling and risk management which is described in Section 2.6 should be applied to determine those that apply in a specific situation. While Section 3.5 already discusses some threats to explain the CIAA process, this section focuses on more general categories of threats rather than on how they can be found in a system.

DoS DoS attacks try to - as the name suggests - prevent a service from being provided. This can be achieved by, for example, opening many connections to a server and thus taking up all system resources so that legitimated clients will be unable to connect. It can also occur if the server crashes or the network itself is flooded with so much traffic that nothing else can get through or if the network equipment is rendered inoperable by malware or direct attacks. The attack can affect a single machine or any larger unit that that machine is a part of like for example a rack, a data centre or an entire country.

Different forms of DoS attacks affect systems such as the distributed P2P storage systems described in Section 3.3.4. An attacker that possesses enough nodes in the network can essentially deny or provide incorrect service to particular nodes and thus hinder their ability to gain sufficient data distribution for reliable restoration access [23].

Similar to network based data transfer, physical transportation of media is also vulnerable to DoS in the form of for example traffic jams, cars being rendered unusable or a fire at the storage facility.

ARP Spoofing Chomsiri[9] explain Address Resolution Protocol (ARP) Spoofing attacks (sometimes also called ARP Poisoning) as a way to intercept traffic sent between hosts in a switched network. During normal operation, once a network switch knows which systems are located behind which ports it only sends traffic to the systems that are the intended recipients. This is accomplished by maintaining a Media Access Control (MAC) address to port lookup table and it means that an attacker cannot see traffic that is not directed at their machine. A way around this problem, for an attacker, is to send forged ARP packets that looks like they came from the gateway router to the host of which they wish to intercept traffic, but rather than sending the MAC address of the router they send their own. Hosts also maintain their own ARP lookup table and they will happily update it when a new ARP packet arrives so after the attacker sent their forged packet the host will send traffic for the gateway to the attackers system which can either forward, modify or simply drop it. This process can then be repeated for other machines as well until all desired traffic is routed through the attackers system.

Incorrect Authentication While it is obvious that clients should use login credentials to authenticate themselves so that nobody else can gain access to their backup storage it is equally important to ensure that one talks to the correct server [31]. If all the client does is send its credentials then an attacker can easily perform a Man-in-the-middle (MITM) attack on the connection and impersonate the server. They can then intercept those credentials and either play along and accept backup data without keeping it, they can store it so that subsequent requests that maybe try to verify the data succeed or they can be sneaky and forward it to the intended server. If login credentials are not regularly changed, the attacker can use them later when they gained access to the client system to try to delete the backup data.

Preston[31] notes that the common storage protocols, Common Internet File System (CIFS) and Network File System (NFS), for Network-attached Storage (NAS) systems are often used with very insecure forms of authentication like host-based authentication which simply verifies if the IP address that initiates the connection or the hostname to which the IP resolves are whitelisted. Such a system can easily be broken by spoofing the appropriate IP address, but even those schemes that use login credentials often send such credentials in plain text which can obviously easily be intercepted and reused. He further notes that Fibre Channel Storage Area Network (SAN) systems, which are an implementation of the traditional client server architecture described in Section 3.4.1, are also often affected by similar issues for example because the common way of handling authentication is via World Wide Name (WWN)-based zones. A WWN is equivalent to a MAC address and both can easily be spoofed because support for changing their value is built into the driver.

Similarly it is also important to perform authentication when transporting physical storage media like tapes or hard disks. If for example a courier comes to pick up the media someone can impersonate that courier, collect the media and then not take it to the actual storage facility [14].

Threats to Data Confidentiality On the other hand an attacker might also decide to simply steal the drive. Different measures, like encryption, are required to protect data confidentiality in case of theft of media either during transit or storage [14]. Any data that is being transmitted electronically should be protected using some form of secure communication like Secure Shell (SSH) or TLS which is the successor of Secure Sockets Layer (SSL). This not only protects the confidentiality, but the two technologies also ensure integrity and - if used properly - they also validate the identity of the server they talk to [31]. Systems that use plain text connections are prone to MITM attacks where the attacker records the conversions and for example extracts login credentials for later use, especially since those systems probably will also be missing any kind of data signing that can be used to expire credentials after a certain time.

Threats to Metadata Confidentiality It may be undesirable when third parties can determine how often and when backups are made or possibly more concerning when they not made. Especially in the case of a home or office computer, which is likely turned off when not used, the fact that backups are not made for some time when otherwise they are created; for example, daily can mean that the person who is normally using that computer is on vacation Garfinkel, Spafford, and Schwartz[14]. note that dormant accounts should be disabled because even if an employee only takes an extended leave the account can be broken into and more worrisome the incident may not be detected by the account owner because they are not around using it. Even worse operators that do notice unusual logins from foreign networks may dismiss suspicions when they know the owner is travelling abroad.

3.6 Verification of Backup Content

There are many reasons which can cause backups to be unusable, including disk failure, bad tapes, software errors and operator error. Garfinkel, Spafford, and Schwartz[14] provide an example of a process that involved an operator starting the backup process and then waiting for a couple of minutes. During that time, no output was produced, so the operator left and assumed the software was creating a backup. Later the software asked for the tape to be replaced, although the operator was already gone and when they came back they did not see the message because the system closed the session after an hour of inactivity. The problem only surfaced because an administrator asked if the operator needed any more tapes. If the backup concept of this organisation had used verification the issue would have been discovered sooner.

It is important to regularly verify that the content of the backups is not corrupted, that the backups are complete and it is possible to restore files from a backup. Depending on how backups are created verification can be as simple as comparing a list of checksums against one created from the content of the backup. Ateniese et al.[3] explain that in other cases it might be necessary to use Provable Data Possession (PDP) to ensure that an untrusted and unreliable client keeps its entrusted data. Thus verification of backups asserts integrity and availability of the backup.

The example of using checksums to verify integrity has already been discussed in Section 2.3. Using checksums is a good way to ensure that the contents of the files is correct and while it does satisfy the requirement for integrity it does not provide assurance that restoration of the backup is actually possible. To catch software or equipment failure, it is important to regularly restore

small amounts of data from the backup to ensure that the entire backup stack works properly. It is especially necessary to select random data, across all backups, for such tests so that old copies are tested as well as recent ones [14].

If local storage is used to store backup data it is also a good idea to verify the content using another drive or computer. Otherwise local issues like a tape writer with an alignment problem, which results in tapes being unreadable in any other tape reader, can cause serious trouble once backups are restored elsewhere [14].

3.7 Recovery Planning and Testing

While it is good if backups are available, they offer little help when the system is broken and restoration from a backup is the only choice yet nobody knows how to restore them. This can be even more of an issue when there is pressure to get the system back on-line as soon as possible. This stress can be prevented if a recovery plan is created that explains, in great detail, all steps necessary to recover the system from scratch using the backup. Having a detailed recovery plan also ensures that there is more than one persons that is able to perform recovery and thus a single point of failure is removed [31].

Preston[31] stresses that it is important that backups are tested by restoring single files, an entire volume, but most importantly testing a full recovery of a system. Missing files and incompatibilities between the process and newer software can only be discovered if recovery is regularly tested. He also states that is crucial to ensure that the backup can be restored with just the backup data and a bare minimum of information and it is equally important to test a recovery of the backup server because with some software one might end up in a catch-22 situation[1] and thus being unable to recover. This can happen rather easily if the backup copies of the backup server are encrypted with a key that is only stored on the backup server itself rather than being printed or memorised. Such a situation might happen because the key was previously used to encrypt off-site backups of the clients and later the server itself was added to the process, but nobody thought about the fact that restoration procedures would have to be different for this system. Furthermore, he argues how important it is to also apply such recovery tests to media stored off-site and thereby also testing if media can be retrieved.

[1] Catch-22 describes a paradoxical situation which cannot be solved due to conflicting requirements or conditions. For example in order to open a locked car one needs its key, however, the problem is that the only key is inside the car. Thus in order to being able to open the car door, it has to be open in the first place so that the key can be retrieved, but since the key is inside and the door is locked opening it is impossible.

4 Different Aspects of the Backup Creation Process

When looking for software that can be used to create backups, it is important to note that there is no such thing as simply creating a backup. There are many different requirements that can influence the choice of software, especially since not all of them can be satisfied by every product. Examples off such requirements are backup creation frequency, used bandwidth and storage space as well as the time needed to create or restore the backup. This section focuses on the theoretical issues while Section 5 discusses the features and limitations of selected examples of software products.

In the following, the backup process is described and different aspects of the backup creation process are discussed and - when appropriate - example implementations are provided.

4.1 Backup Process

A backup process describes how a backup is created, where its data is stored and how correctness is verified. Those decisions are influenced by the importance and time-sensitivity of the data that is to be backed up, but also by the amount of money that is available. Schneier[35] expounds that it is good to keep the process as simple as possible because the more complicated it is the more things can go wrong both on a software side and on the operator side. The following general structure is used when describing a backup process:

1. Storage Location
2. Backup Redundancy
3. Backup Content
4. Backup Creation
5. Verification and Testing
6. Backup Restore

Storage Location First it is important to determine which kind of storage is available or can be arranged. While Cloud Storage or a Client Server Architecture as described in Section 3.3 might be easy to set up and use, they they enable new threat vectors since they use a network connection to transfer the data. The backup process should therefore contain some form of off-line and preferably off-site backup so that even if all connected systems are compromised there are untouched backups available [14]. However, such backups can be combined with networked solutions by collecting backup data for each client on the central server. It is then possible to create the off-line copy from the backup stored on a server rather than from each individual client. The server also acts as an on-line copy and provides the benefits of such a setup.

Preston[31] says that it is imperative to properly label and organise all backup media so that when a certain volume is necessary for a restore it can be found with certainty and in a timely manner.

When deciding to use off-site storage services it is important to regularly test if random media can be retrieved. He also notes that when copies of media are created it is advisable to store the originals off-site and the copies on-site because when files need to be restore the on-site media will be used first. If the copy is kept on-site doing any restore operation also automatically verifies that the copy procedure works and if it does not flaws can be detected before the off-site media is necessary.

Backup Redundancy Backups can be corrupted, stolen or lost so depending on the data that is backed up it is advisable to keep some redundancy by keeping more than one backup, preferably in different physical locations, with different schedules and different retention times. Some organisations keep quarterly or yearly backups indefinitely, but such setups can result in legal issues because sometimes data has to be deleted forever [14].

Preston[31] describes a Tower of Hanoi backup plan that uses levelled backups in such a way that changed files are saved in more than one volume while also trying to reduce the total storage cost. A backup level in this case works similar to a differential backup as described in Section 3.2, but rather than always building upon the last full backup it builds on the last backup of a lower level. However, it does not build upon the most recent backup with respect to its creation time. Rather it builds on the one with the highest number that is smaller than that of backup being created.

Thus if a full backup, which Preston[31] also calls level 0, and a level 3 backup exist and then a level 2 backup is created it will contain all changes since the level 0 backup. It will ignore the level 3 backup because 3 is greater than 2. If a level 5 backup is created it will contain all changes since the level 3 backup, even though level 2 is more recent. When backups are created in this order (0, 3, 2, 5) and a file is changed after the creation of level 3 its content will be in both, the level 2 and level 5 backups because when the level 5 copy is created it builds on level 3 which does not contain the file even though the later created level 2 does.

Using such a scheme does, however, result in a non-linear usage of backup levels which on one hand may not be supported by the backup software in question and on the other hand is rather complicated to wrap one's head around. Preston[31] highlights that it is okay to use such a scheme if one is not confused by it, but he stresses that it is vital not to build an overly complicated system because the more complicated the system is the more likely it is to break. Furthermore, he argues that if a restore is necessary and nobody knows how the system really works the restore can easily go wrong. He summarises that if the system cannot be explained to a stranger in a couple of hours, it is likely too complex.

Backup Content There are two possibilities in terms of the decision concerning what to put in the backup.

- Back up the unique data like user files and system config files only. This approach tries to minimise the space necessary for the backup by keeping data that can be restored from installation media out of it.
- Back up everything including the operating system so that a restore can be done without the need for additional data sources.

Garfinkel, Spafford, and Schwartz[14] recommend the second approach because storage is cheap and restoration is a lot easier. They explain that installation media might get lost while the backup is available so if the first approach is chosen restoration might not be possible even though the backups are available. This means that keeping everything in one place is faster in general and easier to handle, especially under stress when the system needs to be restored quickly.

Backup Creation Different software comes with support for different storage methods and it is therefore important to keep in mind that certain software may not be able to store its data on tapes, disk or in the cloud and documentation should be read carefully. The software in question may also not support encryption of backups, the creation of incremental backups (definition in Section 3.2) or it may support different features for different storage solutions. It may also be necessary to use special software and possibly even special configuration files like decryption key files to restore a backup created by some products and keeping such restoration media available is hence crucial to the recovery procedure and such restrictions should be considered when selecting backup software.

If at all possible backups should be created automatically, preferably without user interaction, to reduce the possibility for operator errors. If automation is not possible the creation should be as simple as possible [14]. Section 4.2.1 discusses this issue in further detail.

Section 5 provides a discussion of selected examples of backup software that can be used for this step, although much more software is available. The different requirements and solutions are discussed in Sections 4.2 and 4.3.

Verification and Testing Creating backups is good, but without regular verification they cannot be relied upon to work once recovery is necessary. Verification and recovery planning and testing are discussed in detail in Sections 3.6 and 3.7.

Backup Restore It is important that systems can be restored from the backups when necessary. As explained in Section 3.7, this requires careful planning and regular testing to ensure that the restoration works as planned.

4.2 Functional Requirements and Implementation Examples

Glinz[17] defines functional requirements as describing specific functions and behaviour of the software in question. For backup software such functions include the backup trigger, encryption and deduplication support as well as support for different storage solutions like optical media or network file systems.

4.2.1 Backup Triggers

When to trigger a new backup is an important question since it directly determines how much data is lost in case the system crashes. If a computer is backed up only once a day then obviously the entire work done on a particular day can be lost if it crashes shortly before the next execution. Backup creation can be activated by multiple triggers like manual, time based or event based execution. Each of these come with their own sets of challenges as described below.

Manual

Functional Requirement The probably most obvious trigger is manual execution, which means that an operator runs the backup creation by starting the program. This may seem simple and reliable, although in reality it is a source of great danger. While it may be cheap it means that a human has to remember to start the process and sooner or later everyone makes a mistake [31]. When implementing any kind of manual control, not only a manual trigger, it is important to consider that humans easily make mistakes and can be rather forgetful. Preston[31] talks about an example where a publishing company lost thousands of images because they forgot to create

backups for a server from the moment they installed it until it broke a year later. McGoldrick and Shin[25] also explain that manual backups may be ignored by employees, even if policy dictates that they need to be executed, because they may be viewed as a time killer, work interruption or simply because they are too confusing for the user. This view is backed by Austin[4] who describes a situation in which a manufacturing plant upgraded to a new system that required 15 minutes of shut down time to perform a daily backup. Shortly before the system was installed, an additional requirement surfaced that said that it had to support continuous uptime without any pause for the backup process during peak production times. Multiple solutions were found and the development team decided that overhauling the system so that on-line backups are supported would be the best way to go, but due to the approaching deadline they implemented an on/off switch for the backup and as long as it would be turned on at least once a week the system would continue functioning properly. They anticipated that this may result in catastrophic failure because, especially during peak production times, the personnel might forget to turn the backup back on and after a few months of operation the system failed in exactly this way. Thus, while it may be possible to put the burden of performing or controlling backups on the users themselves this can easily lead to errors and users forgetting to run the backup job. Errors seem even more likely when pressure is high and in such situations there is even more potential for loss of data or availability because there is more being done in the first place.

There are also obviously different levels of manual execution such as running a backup suite that uses a configuration file to create backups of multiple machines or running a command manually for each machine. If the software is running on the client, it may also involve the user of that machine having to run the software locally and thus each user is responsible for their own backups, thus making it possible for them to forget about creating their backups or find excuses just like described above.

Preston[31] notes that doing any kind of manual interaction with the backup process is prone to errors. He explains that it is important to create backups of all data and to avoid creating any kind of special configuration like excluding certain directories on certain systems because when the system is later used in a different way and nobody remembers to check the exclusion list, valuable information might not be backed up.

Preston[31] also stresses that manually maintaining a list of for example databases or file systems that should be included is dangerous and says that the backup software should detect and back up new ones automatically. He even recommends that software should look for systems that can be backed up rather than someone having to remember to add the system to a list. This ties in to the example above where the server has not been added to the backup schedule and thus no backups were created. If such automation is not possible Preston[31] recommends that the software notifies an operator so that action can be taken rather than relying on someone remembering all the things to do when a new system is installed or changed significantly.

Time Based

Functional Requirement Since manually triggering backup creation is error prone, it is a good idea to automatically create backups after specific time intervals like a week or a day. This ensures that backups are created regularly, but it also adds complexity and it is important to monitor if backups are actually created or if for some reason the automatic job is failing or not run at all. It is crucial that the process creates log files so its actions can be traced in the future, but it is necessary that these logs are monitored so problems can be detected early on [31].

Preston[31] notes that some people will ask the question of when to create backups when setting up an automatic process that runs for example once a day. He explains that the best time is usually

the middle of the night because in a normal office there are generally less, if any, people working during the night and as such files are most stable at that time. This is important because if someone, for example, uses an accounting application while the backup is running, this application might write its data to multiple files. When one of those files is being backed up and the user subsequently makes and saves some changes, it may be possible that the next file that is being backed up already contains the changes yet the previous one did not. Finally, when the files are restored the accounting application will be confused because the contents do not match each other. Some backup software supports creating a snapshot of the file system and thereby prevents inconsistencies between files in the backup, although even when such features exist it is good to pick a time that provides a clean state because if a file changes throughout the day it may be confusing for users which version they obtain when a backup is restored because they may not remember the exact time at which they made a change.

Another reason why it is good to created backups during lean hours is because it means that the system that is being backed up is under less load and as such backups can be created faster. If backups are transferred over a network it is also likely that the network is less loaded and thus available capacity can be used more efficiently [31].

Another issue Preston[31] discusses is that he is often asked whether backups should really be created every night including weekends where nobody is working. He stresses that it is important not to make any special exceptions for any part of the backup process and that includes no exceptions for weekends. He explains that when backups are created only on work days it may very well be possible for someone to finish very important work on the weekend because a deadline is approaching. If the backup is subsequently only run on workdays in the evening and the server breaks down on Monday morning, all the work done over that weekend will be lost. He further argues that the backup software should be able to efficiently detect what was changed and if it can do that then there is no reason to exclude days where no work is done from the automatic backup creation because if nothing was changed the backup will complete very quickly.

An aspect that may be worth considering is that the backup process might run code that is not usually run during normal operation and might touch important parts of the system. This can cause unexpected damage especially if operators are unavailable. For example, when using file system snapshots to ensure consistency a bug in the snapshot code - introduced by an operating system update - may not occur during normal operation because snapshots are not used. However, since the backup process uses them, it can trigger the bug and possibly crash the operating system. It may prove beneficial to plan for such incidents and schedule the backup creation at times where system administrators are available and can respond to the incident like in the morning or evening rather than for example in the middle of the night.

Event Based

Functional Requirement Waiting for the clock to reach a specific time is just a special kind of event. There are many different events that can be used instead, such as the logout of a user or data being written to a file.

Example Implementations Zhongmeng and Hangtian[45] describe a system that backs up data at the time of writing which means that the window during which changes could be lost is practically gone. Changes are monitored by hooking into the file system driver, intercepting write requests and triggering the creation of an incremental backup for the file in question. This process can also leverage additional information like the offset and length of the write request to increase efficiency and reduce redundant information in the backup copy.

4.2.2 Full and Incremental/Differential Backup

Functional Requirement As defined in Section 3.2 a full backup is one that contains all data from a system while a differential backup contains only changes since the last full backup and an incremental backup contains changes since the last backup of any type. Full backups generally allow for fast and easy restoration since only one backup has to be restored, although they also tend to be large and take quite some time to create. Incremental backups are faster, but they form a chain, which makes restoration slower.

Example Implementations A combination of both is used by the rsnapshot tool, which is described further in Section 5.3. It uses rsync to make a copy of the source file system and then uses hardlinks to create a snapshot of that copy, thus making each snapshot appear like a full backup, despite the fact that only changed files are transferred and stored similarly to an incremental backup.

A hardlink is a file that points to another file's data and only makes it available via a different path. From an operating system's perspective, a hardlink is nothing special since files themselves are already hardlinks that point to a portion of data somewhere else. However, in the context of rsync and rsnapshot, a hardlink is like a copy of a file because these tools create additional hardlinks for each backup and when the original file is deleted the data remains referenced and it is kept by the operating system. Only when the last hardlink to a portion of data is deleted does the operating system also remove that data. Changes to the content of such a file will affect all hardlinks equally because they all reference the same content on disk; thus, when hardlinks are used for backup purposes it is important that the tool supports this and rather than changing the backup file it deletes and recreates it so that the data portion is no longer shared [31].

Similarly, one can create a repository of deduplicated data and then maintain an index that maps paths to chunks like Rahumed et al.[32] do in FadeVersion which is discussed in Section 4.2.5. Other deduplication systems work similar and in a sense this is a reimplementation of what a file system does with hardlinks, although file systems are general purpose tools and when many hardlinks are used operations on those file systems can become quite slow, especially for example deleting a snapshot that comprises hardlinks. When such a system is implemented for the specific purpose of creating a backup it can likely be optimised better for certain use cases like for example deleting an entire backup set. When re-implementing the idea, it also becomes possible to store metadata at a different location than actual file data as is done in FadeVersion and in the system described by Storer et al.[36], which is discussed in Section 4.2.6.

4.2.3 Storage Location Diversity

Functional Requirement Backups are only helpful if they can be used if the original system is broken. Similar to Lillibridge et al.[23] in Section 3.3.4, Preston[31] brings up the example of a server catching fire and taking anything in its close vicinity with it. If the backup server and the redundancy server are directly beside the main server and that server burns they will also be damaged and then the original data as well as the backup data are both gone. A similar tragedy could happened due to flooding or an earth quake that destroys the building and the only way to reduce the impact of such events is to keep backups in different locations. Maintaining backups in diverse locations helps to keep availability, as defined in Section 2.4, high. The German Federal Office for Information Security (BSI) [15] also shares these views regarding building redundant data centres and discusses that redundant systems should not be located in the same fire compartment within a building and - more generally - they should be sufficiently far apart such that a local disaster does not affect both of them. Following these guidelines they recommend a minimal distance of 5 km between redundant data centres.

While the examples above may be somewhat extreme there are also more subtle ones like the catch-22 situation discussed in Section 3.7. Furthermore, it is important that information about off-site backups is also maintained in multiple places because if the document that contains this information is in a shelf next to the burning server it may also get damaged and be unreadable. There should be at least some form of off-site backup for everything, including data and documentation, to be able to recover from the worst local disaster [31].

Example Implementations Abu-Libdeh, Princehouse, and Weatherspoon[1] discuss the impact of using a single cloud storage service and explain that relying only on a single vendor easily leads to vendor lock-in because even if a provider raises prices or another has lower ones it is often very costly to move from one provider to the next due to traffic costs. They explain that moving costs can easily trump the cost raise thus leading to what they call "data inertia". They also argue that even though cloud storage providers usually abide by strict Service-Level Agreements (SLAs) which claim impressive availability guarantees, failures still occur and sometimes cloud providers also lose data [1]. While such data loss may be small relative to the total data the provider holds, it can be massive for a few customers in isolation.

Obviously data can be duplicated across multiple providers, but this adds incredible overhead that can be very costly. A much more efficient approach is to stripe data in a similar fashion to RAID 5 by using erasure coding which Abu-Libdeh, Princehouse, and Weatherspoon[1] call Redundant Array of Cloud Storage (RACS). Erasure coding is discussed in further detail in Section 4.2.5, but in short it allows to lose a configurable amount of data while still being able to read it. For example, it can allow for up to two storage providers to be unavailable, yet data can still be retrieved and read from three other providers and the total overhead is thus only 2/5 [1].

When using erasure coding to achieve redundancy [1] calculate that when striping data across nine providers while allowing for 1 failure the total monthly cost overhead is around 11 percent. However, they also note that such a scheme does not take full advantage of single providers that for example have lower storage costs than others because not all data is stored with that provider. On the other hand since data is striped redundancy obviously allows for one provider to lose data and the total cost of abandoning a single provider is much smaller than when only one provider is used because less data needs to be migrated [1].

While using multiple cloud storage providers, rather than relying on a single one, is generally a good idea it is not sufficient in case of backups because if the only backup is stored with this system and a software failure corrupts the data that corruption will affect all redundant copies. This is similar to a local RAID storage. A secondary backup that is not updated at the same time is therefore still important and preferably it should be as different as possible so that incidents are less likely. For example, it should not use the same storage because if an attacker gains access to the storage credentials they could delete all backup data. It is much better if this secondary backup uses simple off-line storage like tapes. This idea follows the idea of Preston[31], who often stresses that it is important to keep off-site and off-line backups and to keep the backups simple because the more complex they become the more things can break and - for example - sometimes old file formats like the dump format may not be readable by more modern tools. This last problem is further explored in Section 4.2.5

4.2.4 Source Data Consistency

Functional Requirement As discussed in Section 4.2.1, it is crucial to ensure consistency between different files in one backup set so that when files are changed while the backup process is running either the old or new state are saved, but not a mixture of both. This issue is especially

pressing when making backups of databases since those generally work with many files and it is often not possible to simply shut the database down to make a backup [31].

Example Implementations With some products, it is possible to put the database in a state where its files are not changed and can be backed up without actually shutting it down and with others there is software available that can create a consistent copy even while data is being changed. While such solutions may work for creating a consistent backup of the database's content, they do not ensure that other files stored on the system are consistent with the database [31].

According to Bhattacharya et al.[6] hybrid systems store some data in a database and other data in a normal file system. For example a web shop may store product information like the name and price in a database, but images of the product in the file system so they can be easily served by a web server. When creating a backup of such a system it is important to ensure that the database backup matches the state of the file system or otherwise either images may be missing for some products or product information may be missing for an image. Either situation may not be handled by the software because it likely should not occur during normal operation.

A possible solution to this problem is to store all data including for example images in the database so that any process that can guarantee consistency inside the database is sufficient to create a backup. Bhattacharya et al.[6] explain that this approach has multiple issues. On the one hand, storing data in a Large Object (LOB) incurs a substantial read/write performance penalty compared to storing the same data in a regular file or on a networked file system. On the other hand, it is generally difficult to access data stored in LOBs since it is necessary to use the database's Structured Query Language (SQL) Application Program Interface (API) to access them rather than normal file system operation functions. There are wrappers that support storing data in a database, but present it as a normal file system. However, such systems do not generally support hierarchical storage management solutions that store data according to its access patterns and thus help utilising fast and expensive storage media more efficiently by moving less used files to slower and cheaper media.

A more practical solution is to bring the database into a consistent state by issuing a write lock and flushing all outstanding data to disk. The write lock means that any database query that tries to change data will block until the lock is released again. Queries that only read data will work just fine and return normally. Once the lock is acquired, a snapshot of the file system is created and thus the file system and the data files of the database are both in a consistent state. Obviously, the write lock is lifted after the snapshot has been created, whereby normal operations can resume and since the snapshot will not change, a backup can be created from its data without having to fear inconsistency issues, even if the backup takes a couple of hours. Preston[31] highlights that an important consideration when using snapshots is that any data written to or changed on the source file system forces the operating system to keep the old data around so that the snapshot continues to exist without modification. This obviously means that additional disk space is used and if lots of data is modified, this additional space requirement can become quite large and should be accounted for.

4.2.5 Backup Data Storage Requirements

While there are many possible storage solutions which have already been discussed in Section 3.3, this section focuses on requirements that may need to be fulfilled by the chosen storage solution.

Volume Organisation

Functional Requirement The most important issue concerning data storage is that of being able to find a certain backup set when a restore is necessary. This requires that volumes are organised in a way that allows for the location of each individual volume to be found at any given point time, as well as ensuring that the volume is actually present at the stored location.

Example Implementations As noted in Section 3.3 Preston[31] recommends to label and organise all storage media and regularly test if media can be retrieved. He also explains that when off-site media vaulting is used to store storage media it is a good idea to keep a database that tracks the location of each media. The vaulting vendor should also keep their own list and the lists should be compared frequently so that lost media can be detected before it is necessary. He also explains that when tracking media locations it is advisable to track individual media rather than containers because when only containers are tracked and one volume of a container is fetched, it may not be tracked by the vaulting vendor because they would not return the container but they would rather fetch the requested volume and leave the container in storage.

Transportation of Backup Media/Data

Functional Requirement Off-site backups are vital to backup availability as discussed in Section 4.2.3. This requires that backup data is stored in a form that allows it to be transported between physical locations.

Example Implementations Transportation can happen either by physically moving storage media or by using an electronic network. Transporting storage media from and to a media vaulting vendor is briefly discussed in Section 3.3 and further explored in Section 4.2.7, alongside the issues that arise when sending backup data via a network link.

Expandability

Functional Requirement When deciding on any kind of storage solution, it is important to think about how large one single backup can be as well as how large the total amounts of backups can become and how the system can scale to store more backup data in the future.

Preston[31] brings up the example of a backup system that used 10GB tapes while the largest client system was 8GB in size. When the clients later got replaced by machines with 20GB of local storage the backup system was unable to handle the fact that it had to store a backup of one client across multiple tapes. This issue can be tackled by either making the storage media large enough or by using smarter software, but most importantly it should be considered early enough so that not too much, if anything at all, has to be changed when the clients grow in size.

Archival

Functional Requirement Backups may be needed again after multiple years, for example, in case a machine gets decommissioned and later data from that machine has to be restored or in a court case. This means that old backups are required to remain readable for extended periods of time and that they are required to be migrated to newer storage solutions when those are changed.

Example Implementations When storing backup media for archival purposes for multiple years, a number of issues need to be considered. The most obvious one is that tapes will degrade and information stored on them may no longer be readable. As noted in Section 3.3.1, this may be due to magnetism from one layer of the tape affecting other layers, although it may also be due to dirt on either the tape or the drive head. Additionally, the data itself may be the issue if it was written with a drive that had a misaligned head and so drives with properly aligned heads are unable to read the data [31].

Similar issues can occur when using different storage formats. Preston[31] gives an example of having to restore a system that had been decommissioned 10 years earlier and when they wanted to restore the tapes they first that to find a suitable tape drive with sufficiently low density. Once this was taken care of, they discovered that the backup was created using the dump format, however a restore failed because the format used by the 10 year old copy of the dump program was different from that of the current version. Porting the old code to a current system proved very difficult, but luckily there was a second backup set that did not use the dump format but was rather saved with TAR which is an arguably simpler format that did not change as much and could still be read using modern tools.

When using tapes there are yet more possibilities for problems. Tape drives perform differently when used with different block sizes and reading data with a different block size than what was used to write it likely results in Input/Output (I/O) errors. There are also drives that support transparent compression of data to store more data on one tape. The used compression algorithms may not be supported by other drives because they are optional, but worse, vendors may implement special algorithms that perform better than the standard ones. When such compression is used it may prove impossible to read the tapes in the future when the vendor has gone out of business and the drive is no longer working properly. In such cases it pays off to plan ahead and use common compression algorithms. Finally different tapes may look similar and sometimes they are not labelled regarding which kind of drive they should be used with. Drives made by different manufacturers may also be incompatible with each other [31].

When using software that saves backups in custom formats it is advisable to regularly test the backups as explained in Section 3.7 and it is even more important than when using software that saves in the TAR format or a similarly common one because with custom formats there are most likely less programs available that can read them. In many cases, the software that created the backup is probably the only one that can read it again and if the vendor changes the format in later versions it might prove difficult to read old data similar to the issue with the dump format described above. There can also be problems because the custom format may not be properly designed or the vendor does not publish information about its restrictions like a maximum file size or a maximum number of files that can be stored in one archive. Data that exceeds this limit may be corrupted, the entire archive may be unusable or restoration may be impossible [31].

Finally Preston[31] notes that backups should not be used as a general archive because searching for a particular file in all backup copies can take a really long time while specialised archive software will do a way better job. Also backups are snapshots of systems at particular points in time and if files are created and deleted within a backup interval they will not be in the backup and thus cannot be retrieved. Archive software will keep such files and allow for them to be retrieved if they are necessary; for example, in a law case.

Deduplication

Functional Requirement To reduce storage space requirements files that are the same should only be saved once. The process of finding duplicate data and replacing duplicates with links to

the original copy is called deduplication. When applying this technique to complete files the result is file-level deduplication, but sometimes larger files differ only partially which requires smarter algorithms. According to Sun et al.[38], a better way to detect partially equal files is to apply block-level deduplication which divides the data into fixed size or variable length blocks and detects those with equal content.

Example Implementations The Fingerprint-Based Backup Method (FBBM) developed by Yang et al.[43] does not only detect identical files, but rather splits the files into variable-sized chunks ranging from 2KB to 128KB and deduplicates those separately. The splitting is done in a way that detects anchors in the file rather than using fixed block lengths so even if a file is changed in the middle the chunks before and after the change will remain the same and can be deduplicated properly. Identical chunks are detected based upon their SHA1 hash and it is assumed that no collision will occur.

Ling and Datta[24] created a system that they call InterCloud RAIDer. It stores deduplicated, non-systematic erasure coded backups in multiple cloud services to provide fault-tolerance against individual providers. Erasure coding is described later in this section. Similar to FBBM, deduplication is performed with variable-sized chunks to overcome the boundary shifting problem of fixed size chunks. The boundary shifting problem describes the issue that when a file is sliced into fixed sized chunks and a single byte is removed, all chunks following the change will have different content due to their fixed size. By using an algorithm that detects certain features in the data it is possible to have relatively stable anchors that can be used as block boundaries and as such data following a change will still contain the same anchors and continue to be deduplicated. Anchors around the change may differ, but most of the blocks of the changed file will remain the same.

When using block-level deduplication it is crucial to carefully choose what should be the average block size. A smaller block size increases deduplication efficiency because small changes will affect less blocks, but it also increases the number of total blocks and thus the system's memory and processing overhead because each block has to be compared to existing blocks to determine if it is a duplicate and at least in InterCloud RAIDer the list of existing blocks is kept in RAM for performance reasons. A larger block size pretty much has the opposite effect. It reduces computation and index overhead, yet it also reduces efficiency because small changes affect larger blocks and as such more storage space is wasted [24].

Min, Yoon, and Won[27] provide discussion and comparison data for possible performance improvements when using deduplication. They propose the notion of context-aware chunking which uses a file's extension to determine which chunking method should be applied. Variable-sized chunking is more expensive than fixed size chunking and they explain that while variable-sized blocks provide better space efficiency due to eliminating the boundary shifting problem they cannot cope very well with media files, as well as compressed and encrypted data. These types of data change their entire form even if the original file is only modified slightly and therefore it is unlikely to find duplicate blocks even when using variable-sized chunks and doing so may not be worth the effort. Text files, documents and source code usually get updated partially which allows deduplication to work better. However, they also note that such files tend to be small and applying variable-sized chunking may not help efficiency due to the cost of computing the chunk boundaries.

Context-aware chunking determines the type of file that it has to deduplicate and conditionally applies fixed sized or variable-sized chunking. Data that is a media file, compressed or encrypted content is split into fixed size chunks because it is categorised as immutable while other data is seen as mutable and variable-sized chunking is applied [27].

Min, Yoon, and Won[27] also propose a new variable size chunking algorithm that they call Incremental Modulo-K. It is a variant of the more commonly used Rabin's algorithm which

extensively uses modulo operations and while it is already very efficient it is possible to increase that efficiency by replacing some of the modulo operations with faster ones such as SHIFT and XOR. They measure a 10 to 15 percent increase in chunking performance, however, actual performance is wildly affected by the chosen target pattern. The target pattern determines the boundary between two chunks and a longer pattern results in larger chunks since it will match less often. A 13 bit target pattern results in a mean chunk size of around 10KByte while an 11 bit pattern produces 4Kbyte chunks. With a 10 bit pattern they are able to process 250MB/s of Linux source code while with a 12 bit pattern the performance drops to 122MB/s. It is important to keep in mind that a longer pattern results in less chunks and therefore the lookup table is smaller and lookups will be faster. When using an 11 bit pattern rather than a 13 bit one the deduplication ratio improves by a mere 0.66 percent while the number of fingerprints and their management overhead increase by 250 percent and they note that the overall backup speed becomes faster when larger chunks are used [27].

Erasure Coding

Functional Requirement Deduplication intentionally reduces the amount of duplicate data and as such if data is lost it is more likely that there is no other copy around. If chunks are shared between multiple files and one chunk is lost it can obviously even bring multiple files in a broken state which means that it is crucial to ensure the integrity of the stored data [27]. Erasure coding restores some of that lost redundancy in a controlled manner, if required, to satisfy availability requirements.

Example Implementations Creating redundancy can be done by using RAID variants as described in Section 3.3.1 or with erasure coding which is a more general form of the idea used in RAID 5 and which is used in InterCloud RAIDer [24]. Erasure coding creates redundancy without having to resort to full duplicate copies and it allows for some of the encoded data to be lost while still being able to reconstruct the original input data. An erasure code can for example create ten encoded data streams and only require any six of those to be able to reconstruct the data. A systematic erasure code creates encoded data pieces where each piece on its own is sufficient to retrieve part of the original input while a non-systematic erasure code like the one used by InterCloud RAIDer encodes data is such a way that multiple pieces are required to gain any information about their contents. Non-systematic codes thus not only provide redundancy, but they also ensure confidentiality as long as no single party possesses enough pieces [24].

Assured Deletion

Functional Requirement When using untrusted storage another issue that comes up is that of deleting data because if data is kept forever sensitive information may be exposed in a future data breach [32]. Some laws even require data to be deleted after a certain amount of time [31].

Example Implementations Rahumed et al.[32] provide a cloud backup system called FadeVersion that allows for assured deletion of data in the cloud even if the cloud provider keeps copies of the data after a client issued a deletion request. It provides deduplication and uses layered encryption to allow for any backup to be deleted without affecting other copies. Layered encryption means that each object is encrypted with a random key and that key is then encrypted with a different key for each version of the file. The second key, which they call a control key, is kept

in a key escrow system rather than in the cloud because to provide assured deletion they rely on the encryption system being secure in a sense that it is infeasible to revert the encryption without the key. They explain that keeping a key secure and - more importantly - being able to completely erase that key is easier than assuredly deleting the backup data itself because the key does not have to be stored in the cloud since it is much smaller.

Rahumed et al.[32] claim that when a particular control key is deleted, it is impossible to decrypt the object keys stored in the cloud and thus the backup is assuredly deleted. However, this is misleading because as explained in Section 3.4 encryption should not be assumed to remain unbreakable forever.

4.2.6 Server Pull and Client Push

Functional Requirement When backups are created and stored on the same system there is only one place from where to initiate the backup, but when they are transferred via a network link either the server or the client can start it and the client can push the data to the server while the server can also pull it from the client. Both of these methods have some more or less important issues that may be resolved by a hybrid solution, which is subsequently described In the following, server will always refer to the storage system and client will always refer to the system that is being backed up even if the storage system initiates the connection and is thus, technically, a client.

Pull

When the server pulls data it needs to be able to talk to the client and initiate the connection and thus it has to possess some kind of authentication credentials. Since it can open connections, an attacker that has control over the server may be able to use those credentials to further attack the clients. This is especially true if the backup data that is pulled from the client is not encrypted and the attacker can read it to determine passwords, additional services or other weaknesses in the client systems.

Using pull backups allows the server to determine when backups should be created and it can rather easily balance the load when there are many clients, by simply limiting the number of backups that are created in parallel. The server can also stop making connections when it believes there is a threat in the network and since all it has to do is make outgoing connections to backup clients it can be configured to undertake its work without any service listening for incoming connections.

However, such a setup requires that the clients provide a listening service which creates problems similar to those that were discovered with the active mode of File Transfer Protocol (FTP). Mainly, it requires that the firewall allows incoming connections to clients which means that they can be attacked more easily [14]. Bellovin[5] points out that this is especially troublesome when combined with the issue that keeping many client machines secure is more complicated than doing the same for a single or a few server systems.

However, using pull backups reduces management overhead because all active parts are located on the server while the clients are passive since they only accept incoming connections. If the backup schedule needs to be changed, backups should be kept for longer or the underlying backup software changes, such adjustments can be made on the server, rather than having to modify configuration data on each client. If a client is not available for unknown reasons or the backup fails, the server can notify an operator about the error.

If backup data is not encrypted on the client, the server can apply deduplication across all clients without much additional work [36], however, storing non-encrypted data for many clients in a central place can be a nightmare if an attacker gains access to the backup server.

Another issue that can arise - especially when a pull backup scheme is used with non-server computers - is that the client may not be on-line when the server wants to create a backup. This is different from the case discussed above where a client is not available for unknown reasons because, for example, office computers might be expected to be turned off when nobody works with them. A possible solution to these situations is to simply try to create backups very often or to monitor the network to detect when clients are available. One could also restrict the time window during which backups are created to working hours; however, as explained in Section 4.2.1, this can lead to unexpectedly large loss of important data if someone decides to work outside the normal hours and the machine crashes before the next backup window [31]. Similarly, a misconfigured firewall can cause the backup server to be unable to connect to a client and the server may interpret this as the client being off-line while the user of the client system may be able to connect to other systems just fine and thus they may not notice the issue themselves.

Push

Push backups work the other way round, whereby the client uploads its data to the server which needs to listen for incoming connections and the client obviously needs some authentication credentials so it can open the connection. Contrary to pull backups, uploading data simplifies the creation of firewalls, which Garfinkel, Spafford, and Schwartz[14] consider a major advantage when choosing between passive and active FTP.

In addition to the authentication credentials, the client needs write access to a directory on the server and it can thus potentially upload arbitrary data and try to exploit bugs to gain additional privileges. Software like duplicity and rdiff-backup also assume that they can change and delete files on the server because they manage deduplication and versioning on their own. Since the software can do this, an attacker can use the same credentials that are used to create the backup to delete existing backup data before deleting the system itself. Such actions may be limited to one particular client's backups if privileges are not escalated, but losing backups against an attacker can still cause considerable damage if there are no additional backups that are protected against direct access. Of course this additional backup may also be implemented by, for example, creating snapshots of data on the server or by using chroots, but then a privilege escalation attack still poses a great threat. The chroot function restricts the file system a process can interact with to a certain directory and thus limits the amount of damage that can be done, however, it can be easily circumvented if administrative privileges are obtained. Moving files from the writeable directory to another that is either not writeable or not accessible at all will not work with software like duplicity and rdiff-backup that expects to be able to modify its storage location to provide some of its features. Nonetheless these issues may be less troubling than having to keep many clients secured in a pull backup scenario.

Another issue with push backups is that the backup is configured on the client and any changes in schedule, software or general configuration need to be changed on each client. This increases management overhead and makes it easier to miss performing adjustments on particular clients. This may be addressed by implementing a hybrid solution as discussed in Section 4.2.6. Since the client initiates the backup process, a configuration error, like setting the backup to only execute once a year rather than once a day or failing to configure the cron daemon to start automatically and thus never executing the backup, is more easy to miss. The cron daemon executes scheduled tasks, but if it is not running its configuration will not be used and no task will be executed. Even if the backup script would notify the operator about failures, such a feature will not help if the script

is not run and it is likely that the configuration error will not be noticed unless there is a second mechanism in place that, for example, verifies that something was written to the backup storage space of each client within the last couple of days.

It is also more complicated to control server and network load because the client decides when a backup is created. If many clients all upload their data at approximately the same time, either the network or the server will be under very high load, making the entire process slow and due to scheduling conflicts it will be slower than it would be if uploads were to happen sequentially. Multiple concurrent streams of data being written to disk essentially create random I/O which means the disk has to seek more often and during seeking no data can be read or written so effectively throughput decreases. Yet when no backups are created neither the network nor the storage will be used and thus available resources will be used inefficiently.

When encrypting backup data on the client, using push backups means that the server does not have to be particularly trustworthy and it especially means that an attacker with full access to the server no longer possesses the possibility of gaining an incredible amount of information about the clients. The best they can do is to delete backups of clients, but they cannot read them and thus cannot use the information inside to simplify attacks on the clients. However, encryption makes it difficult to deduplicate data on the server and thus data stored by clients will always differ and deduplication efficiency will be very low when used in an environment like an office with many different clients where most of them run similar software, but use different keys [38].

Storer et al.[36] describe a system where data is deduplicated on the client and each chunk is encrypted with a hash based upon its content. The mapping, which chunk belongs to which file, is encrypted with a client specific key, but the chunk itself is encrypted in a way that allows other clients that possess the same chunk to also encrypt it and produce the same output. Thus the encrypted chunk can be stored only once on the server. Such a system allows for secure and space-efficient storage of backup data, although it requires that all clients create and encrypt chunks in the same way and the authors note that deletion and garbage collection do not work well since nobody knows which backup sets need a given chunk because that information is encrypted with a different key per client.

Push-Pull Hybrid

For certain situations it may prove beneficial to use a combined solution of both methods. Arguably, keeping a single or a few servers up to date and secure is easier than doing the same for many client machines and it can be desirable to not allow incoming connections on clients. However, if software is unable to listen for incoming traffic on a client then a pull scheme is not possible which means that it is rather difficult to efficiently utilise network and storage performance because the server cannot open the connection when it has resources available. In a hybrid solution the connection can be opened by the client, but rather than using it upload backup data the connection is kept open until the server decides that it wants to receive the data and only then the client starts its transmission. It also allows the client to get configuration data from the server which simplifies management. Thus the only system that needs to allow incoming connections is the storage server and clients make the initial contact. This also allows the client to easily detect if there are problems establishing the connection and alert the user that backups cannot be created.

4.2.7 Transport Security

Functional Requirement When data is being moved - either virtually between systems or physically between storage locations - it is exposed to multiple threats. Section 3.5.3 provides a

few selected examples of threats, most of which apply to transportation of data. Different from the last section client and server should now again be understood with their original meaning where a client initiates the connection and the server accepts it. This also means that most of the examples discussed below will be applicable to both, push and pull backups, regardless of which system starts the connection.

4.3 Non-functional Requirements

Glinz[17] provides discussion on the definition of non-functional requirements and notes that there is no real consensus on what they are. For the purpose of this thesis, a non-functional requirement is defined as a quality or property of the system like speed, efficiency, usability or the fact whether or not it is open source. Advantages of using open source software have already been discussed in Section 2.7 while usability will be evaluated when discussing examples in Section 5.

4.3.1 Speed

Non-functional Requirement It is important to determine how fast the backup has to be created because if the creation takes longer than the backup interval, in a time-based system, then backup processes will pile up and slow each other down even more. Additionally, it is always desirable to create backups as fast as possible to reduce the window during which data modification can affect them.

Example Implementations Speed is also a factor when using snapshots because for the duration of the backup the file system needs to write new data to an alternative location and keep the original data around thus requiring more storage space until the snapshot is removed [31]. Consequently, a faster backup process allows for a lower storage overhead, but making a backup faster for example by increasing its priority without consideration can lead to performance problems of the system during the backup creation. Such problems are discussed in Section 4.3.2.

On the other hand, there are ways to increase the general backup speed that may not affect system performance as much as raising the priority. Deduplication allows to transfer less data while ensuring everything is backed up. In this case, incremental backups can be considered a form of deduplication as their primary aim is to reduce used storage space as well as making backup creation faster, although normal deduplication also increases performance because data that is already stored does not have to be transferred again [31].

An aspect that may be worth considering is to ensure that the backup creation always takes a similar amount of time to complete, whereby it is possible to plan for it efficiently. When using a scheme with full and incremental backups, this is unlikely to be the case because the point of incremental backups is that they are faster than their full counterparts, however, since they form a chain they cannot be stacked forever and a full backup is required once in a while. The full backup will then take considerably longer than the incremental backup and so the creation time can vary greatly. This can be a problem when one wants to create backups of a laptop, for example, at the end of a work day. In this case, it may prove beneficial to ensure that the full backup is created on specific days so that the user knows this in advance and the longer time can be accounted for.

Depending on the bottleneck, different solutions can improve the backup process speed. If the network is slow because the backup is for example uploaded to a remote server via a home internet connection then incremental backups can save a great deal of time. However, the first backup will still be slow, although it can be made faster by employing compression and deduplication. In

a sense deduplication is also a form of compression because compression tries to find duplicate parts of data in a file and replace them with a shorter identifier which is essentially the same a deduplication.

Preston[31] says that if the backup server is connected via a Local Area Network (LAN) then it may also prove beneficial to set up a dedicated network for backup purposes. He proposes this so that the backup process cannot only use the full network link's capacity, but also that at the same time the main network, that is, for example, used to serve client requests, is left alone and performance does not degrade due to excessive traffic [31]. In a real-world scenario, such a setup is, however, very expensive and a much better solution is to employ traffic shaping to reduce the maximum bandwidth that each kind of network traffic can use. If a single network link in total is insufficient, multiple links can be combined and normal as well as backup traffic can share this combination, thereby maximising efficiency.

To increase the speed of a backup server that uses tapes, Preston[31] explains that tapes have an optimal speed at which they can process data best. If this speed cannot be reached because data is coming in too slow, the tape has to rewind more often which causes the entire process to slow down greatly. This limitation can be overcome by either buffering backup data on a local hard drive so that the tape can then be streamed at maximum speed or by interleaving multiple backup streams. While interleaving streams does improve performance during backup creation it will, however, also mean that data is interleaved on the tape and thus when restoring a particular backup that data has to be skipped and skipping data slows down the effective transfer speed because the tape will already turn at maximum speed.

Xia et al.[42] describe the Recovery Point Objective (RPO) as "the maximum time period in which data could be lost due to a system failure". A lower RPO requires that backups are created more often, but such a requirement also means that creating backups takes increasingly more system resources away from the main purpose of the system and therefore decreases performance and/or availability. A common approach to this issue is to use a mixed schedule of full and partial backups where partial backups can be either differential or incremental [42].

As discussed in Section 4.2.1, it is possible to back up data at the time of writing and so the backup is always continuously created in the background. Using such a system means that the RPO is very low and thus only marginal amounts of data are lost in case of a crash. However, if the backup system is not fast enough to process all incoming data in real time, it may lag behind. Possibly worse, if a write request to the file system only completes once the backup is complete, a slow backup system will cause everything else to be slow due to waiting times, even if other systems have sufficient resources available.

4.3.2 Impact of the Backup Process on the System

Non-functional Requirement Creating backups accesses a considerable amount of data - if not all of it - and thus can come with a high performance impact, especially if the system is being backed up live while providing its normal services. It may be important to control this performance impact so that normal operation is not interrupted.

Example Implementations Xia et al.[42] created models to describe what impact the backup process has on the live system. Depending on whether the data is copied on-line or off-line, the system's resources are either shared with other processes - thus reducing maximum performance - or not available at all. There are also multiple types of backups that result not only in different restore time but also differences in creation time and necessary system resources. These types are described in detail in Section 3.2

Xia et al.[42] not only focus on the performance impact of backups on a system, but also on the availability of its services. In their example, they use a file storage system that allows users to access local data while also allowing them to change it. Without periodic backups those changes will obviously get lost, but each backup creation takes up some of the system's CPU, network and I/O resources. They serve files using the Apache web server and create backups using the rsync file copy tool. Based upon this setup, they create models that can be used to determine "the file server availability, the rejection rate and ratio of user requests, and the loss rate and ratio of system data".

In their setup, performing a daily off-line full backup results in significantly reduced file server availability, as can be expected since the system is shut down quite often. Creating the same full backup on-line, rather than off-line, hardly impacts the availability yet other metrics remain similar. They note that prioritising a backup process that uses only on-line full backups causes heavy increase in request rejection when compared to one that runs without priority, yet it hardly improves other metrics. Finally, they note that a mixed strategy that employs full and partial backups is more effective than one that creates only full copies and they also show that prioritising partial backups hardly improves desired metrics yet increases request rejection similar to the full backup case.

5 Selected Examples of Software for Backups

In this section, multiple examples of backup software are discussed, as well as their features, strengths and weaknesses. While comprehensive systems like Bacula and BackupPC try to simplify management of backups for many machines, they add quite a lot of complexity which Schneier[35] considers to be the enemy of security and as such they are not discussed further in this work.

5.1 Tarballs

Probably one of the most reliable and simple tools to use for backup purposes is TAR. Preston[31] explains that almost everyone knows how to read a tar file and if they do not, it is easy to show them. He also often highlights that the format itself is rather simple and hardly changes which makes it perfect for archiving and backup purposes. Even backups that are 10 years old can still be read using modern tools which is a feature that not all software possesses. TAR is also much more portable between different platforms than for example dump or cpio [31].

Preston[31] explains that all TAR does is that it takes a directory or file and creates an archive that contains it. This simplicity allows it to be used as a part of a self-written backup script that combines multiple similarly simple tools and it is also used as a building block in many more complex open source software programs. By itself TAR does not support compression or encryption, however, the resulting file can be processed by other tools to implement this kind of functionality. TAR also allows to write its data to standard output (stdout) which can be used as an input for other software like gzip, openssl and gpg and thus data can be post-processed without the need for a temporary file on disk [14].

Example Creation of a tar archive that is compressed and encrypted can be achieved by a command similar to this [14, 31]:

```
# tar -cf - /home | gzip | openssl enc -des3 -salt > backup.tar.gz.des
```

It should be noted that openssl will prompt the user to enter a symmetric key so this command cannot simply be executed automatically by a cronjob.

Unsupported Features While TAR is a powerful and simple tool, it also lacks some features that may be required in certain environments. For example, TAR only generates an archive and since it aims to support tapes and Unix pipes - as shown in the example above - it does not support deduplication of data. Preston[31] also notes that not all versions of TAR support incremental backups or preserving permissions and ownership of files and if such features should be supported, one has to use the GNU version.

If the created files are encrypted or compressed, then deduplication by other tools will not be very efficient, as discussed in Section 4.2.5. It is also important to note that unless incremental backups are used, each archive will be a full backup and thus keeping multiple backups will require a lot of storage space.

5.2 Duplicity, Duply

Duplicity can be used to create encrypted, incremental tar archives. It allows storing its data on different types of storage systems ranging from local to cloud storage, although, it can only upload backup data in a push fashion. The duplicity website [11] notes that tar archives do not support seeking once they are compressed or encrypted so to find a specific file in an archive, the entire archive has to be decrypted and decompressed. The documentation on the site further explains that incremental backups use the rdiff tool to record differences between files rather than including a new copy of the entire file in each backup set. While the first backup set is an ordinary tar archive that contains all files and can simply be extracted even without using the duplicity tool, an incremental archive is more complicated to use when the tool is unavailable. However, the website states that even an incremental backup can be restored manually by applying the differences with the dedicated rdiff tool.

Since backups are encrypted on the client and the push principle is used the advantages and issues discussed in Section 4.2.6 apply. Mainly, the client's credentials can potentially be used to attack the server because they provide at least some form of data upload and removal permissions, but since backups are encrypted on the client an attacker on the server cannot gain any additional insights into the client's configuration because they cannot read the backup data. This greatly reduces the impact of attacks on the backup server and makes it a less interesting target and is in direct contrast to the tools discussed in Section 5.3 and 5.4.

Limitations Duplicity uses incremental backups as well as archives rather than some kind of object storage and thus suffers from the chain problem discussed in Section 4.2.2. The user has to create full backups rather than incremental ones at regular intervals because otherwise old backups cannot be deleted and restoration will become slower the more incrementals are kept. While duplicity provides an option to automatically create a full backup after a number of days, it cannot work around the issue of extra storage space that is necessary to keep multiple full backups. Depending on how much data changes this can lead to higher storage requirements when compared to a solution like rsnapshot which is discussed in Section 5.3. The duplicity website [11] also states that the tool can require quite some temporary space, on the client, when restoring a backup.

Duply Duply is a front-end that tries to simplify usage of duplicity by keeping data like the backup location, login credentials and keys in a dedicated directory.

5.3 Rsnapshot

Contrary to duplicity, rsnapshot does not use tarballs for storage, but rather it uses rsync to copy files from the original system onto the backup system in a pull fashion. It uses hardlinks as a form of incremental backup as discussed in Section 4.2.2. Each backup looks like a full backup when browsing the directory, however, files that did not change from the older backup are hardlinked to the old data thus saving disk space. Note that only complete files can be hardlinked and as such files that are only changed slightly, or extended like a log file, will not be stored this way [31]. Especially with log files or mailbox files each backup is likely to contain an entirely new copy of the file and Preston[31] notes that in such situations using rsnapshot is the incorrect way to go and alternative software should be preferred.

One of the benefits of this storage scheme is that each backup can stand on its own in the sense that each can be individually deleted. Rsnapshot supports multiple backup levels like for example

hourly, daily and monthly. Backups from each level are numbered, yet any copy can be deleted without affecting others because the operating system transparently handles hardlinks and only truly deletes data once all hardlinks have been removed. Using existing file system features in this way also allows for access without any special software, since the backup data is really just a normal directory in a file system.

Performance One of the drawbacks of rsnapshot when it is used for extended periods of time is that it creates an enormous number of files. A single backup comprises as many files as the source system contains, which can easily be hundreds or millions, and each snapshot created by rsnapshot contains a full set of all of them. Deleting such a snapshot can take many minutes and even though rsnapshot tries to delete as little as possible by rotating existing snapshot to their new position and only deleting the last one, it cannot work around the problem that there are millions of files that the file system needs to track.

Another problem when using that many files is that creating a copy of the backup becomes increasing slower the more systems are to be backed up. Software that creates large tarballs that store data for many files may easily end up with a number of files well below 1,000 yet rsnapshot and any backup software that simply copies files to the destination system is likely to create well above that number. However, creating tertiary copies is important because a single backup cannot adequately protect data against different kinds of threats as discussed in Sections 4.2.3 and 4.2.5.

Other Problems Using rsync to create a copy of the original system's data obviously means that the backup is not encrypted and - to the best of my knowledge - rsnapshot does not support any kind of encryption. This can be mitigated by storing the backup data on encrypted storage for example by combining Logical Volume Manager (LVM) and Linux' dm-crypt to create an encrypted block device and then creating a file system on top of it. Confidentiality is subsequently ensured as long as the block device is not decrypted. Since rsnapshot stores a copy of the original files on the backup storage's file system this storage system must support all features used by the source. Preston[31] highlights that permissions and features like ACLs and file system extended attributes are especially affect by this issue. He discusses that a possible workaround is to create a script that dumps all this information in a file that will then be included in the backup.

5.4 zfs-backup.sh

I have created zfs-backup.sh due to memory problems caused by rsnapshot creating millions of files on my file system. The main idea follows that of rsnapshot in that it uses rsync to copy the original data to the backup server, however, rather than using hardlinks to create snapshots it uses ZFS' snapshot feature which is much quicker. Strobl[37] explains that rather than copying each inode at creation time, like rsnapshot, ZFS, which models the file system as a tree, simply keeps the old tree when it is being changed. Creating and deleting a single snapshot now only takes a couple of seconds rather than many minutes.

It still suffers from the problem that a single backup contains many files because the source system contains that many and it also does not support encryption because files are direct copies. Additionally, it also requires the target file system to support all the features used on the source system because the target aims to be an exact copy including permissions, timestamps, ACLs and extended attributes. Nonetheless, like rsnapshot, it relies on the file system and general operating system tools and no special software is required to access backup data which reduces the risk of being unable to later read a backup as is often argued by Preston[31] and discussed in Section 4.2.5.

Security Since zfs-backup.sh is based upon the pull principle the server is able to make connections to clients and it possesses sufficient permissions to read any file on the client's file systems. As explained in Section 4.2.6 it is much harder to keep many client systems secure than it is to secure a small number of servers. Transfer of data is secured by using the ssh backend of rsync which performs mutual authentication and employs encryption to ensure confidentially. While the transfer is encrypted, most of the security on the server side relies on the operating system's handling of file permissions. The script itself does not use any kind of encryption of stored data but rather expects the server's file system to encrypt data before it is stored on disk. This means that an attacker who is able to gain root privileges on the server can easily look into all backup data and determine possible weaknesses in clients to further attack the network.

5.5 Comparision

Table 5.1 compares the different examples discussed in Section 5.

Table 5.1: Comparison between selected backup software

Feature	Tarballs	Duplicity	Rsnapshot	zfs-backup.sh
Software availabity	Preinstalled	Extra install	Extra install	Extra install
Encryption	Archive creation	Archive creation	Storage file system	Storage file system
Incremental Backup	GNU version	Yes, requires regular full backup	Yes, always	Yes, always
Compression	Yes	Yes	Storage file system	Storage file system
Deduplication	No	Yes	Hardlinks	Hardlinks
Restore of single file	Slow, needs to read whole archive	Fast, many archives with index	Fast, direct copy of all files	Fast, direct copy of all files
Amount of files on storage	Small	Small	Very high, many hardlinks	High, file system snapshots of one full copy
Deletion of old backups	Fast	Fast	Slow, many hardlinks	Fast
Connection direction	Pull and Push	Push	Pull	Pull

6 Backup Concept for a Small Network

This chapter describes a backup concept for a case example of a small network which is described first. Subsequently, the implementation is examined and some numbers regarding storage space and backup speed are provided, as well as other data about a test of the concept in practice.

6.1 Description of the Case Example Network

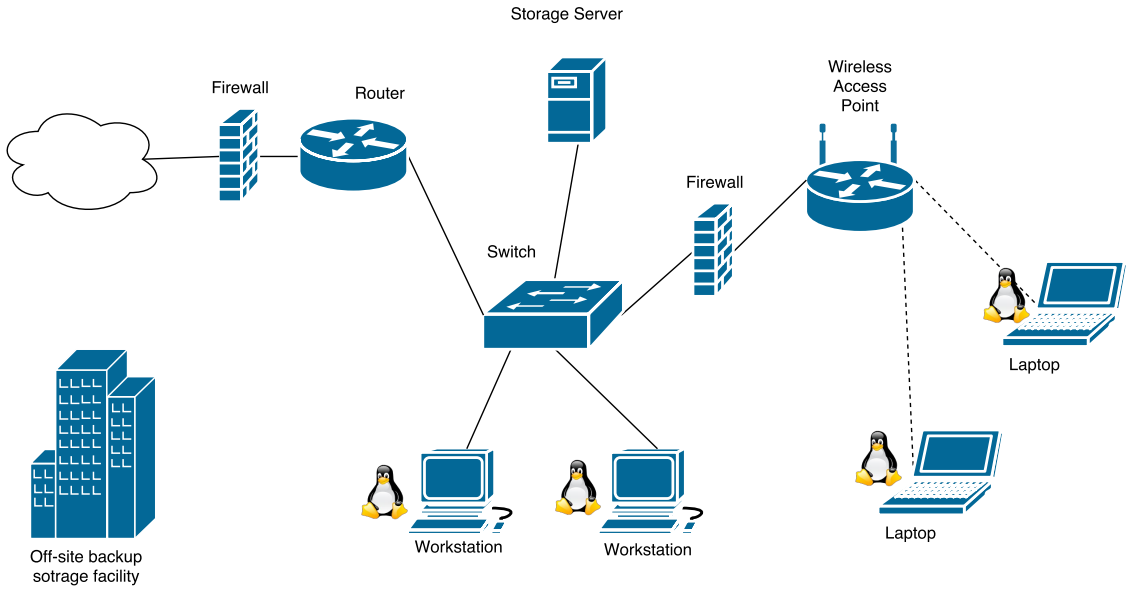

Figure 6.1: Network graph of the example network

The example network comprises two workstations, two laptops and one storage server all in the same LAN. It is assumed that the storage server provides NAS services to the network and thus does not need to be acquired for the sole purpose of providing backup storage. Figure 6.1 shows the configuration of this network, although the devices labelled Firewall, Router, Switch and Wireless Access Point may be contained in one single enclosure especially in home or small business environments. Because this work generally focus on FLOSS, all systems in this example run some form of Linux.

The systems in this example network are assumed to be used in a software engineering setting. This means that typical programs used on the systems include web browsers and email clients, text editors and software development tools like compilers, as well as a data base for development purposes. The systems use Solid State Disk (SSD) drives with a size of 240GB, of which 120GB are used. The operating system has many additional development tools and libraries installed and uses 20GB of disk space, while logs and database files use 10GB. The user of each system uses 80GB of data for various projects, test setups, reference files and application data. The remaining 10GB are used for temporary or easily reproducible files - like distribution package files - which are excluded from the backup. As mentioned in Section 4.2.1, excluding files can cause problems when different exclusion lists exist for different systems. This problem does not apply here because

all systems use the same, very general, exclusion list that only excludes very specific and carefully selected files.

During a normal day of usage, files with a total size of 10GB are changed, however many changes only affect small parts of the files like appending to log files or additional entries in the browser history. The real number of changes is 2GB per day. Operating system updates only occur every two weeks and change a total of 5GB per update.

The NAS only needs a minimal operating system with 5GB in size to provide network services. Additionally, it stores 115GB of user data which is served via the network. This data only receives 1GB of changes per day.

3TB hard disks are used because they provide sufficient storage space while keeping costs and rebuilds times of the RAID low. They could store five full backups of all five machines even if compression or deduplication were not used. For simplicity, the same type of disks is used in the NAS as well as for the off-site storage media. Tape is not used for backup storage because the drives are rather expensive and due to the reasons mentioned in Section 3.6 at least two drives are required to properly verify written tapes and hard disks provides sufficient storage capacity.

6.2 Threat Model

The most important question concerns the reasons why backups should be made and against which threats they should provide protection because these threats determine what the system must support. It is assumed that in a typical small office the main issues are hardware failure as well as data loss due to operator mistake and malware. On the other hand, physical attacks on hardware like theft and targeted attacks against backup copies are considered very unlikely. However, these assumptions will probably not hold for high profile business environments where a criminal may try to attack the backups, although the risk assessment for such situations is left to those who actually implement a concept in such an environment. This discussion focuses on possible attacks and - as explained in Section 2.6 - likelihood has to be determined on a case by case basis.

Attacks on the server have already been discussed in Section 3.5.2 but some of the problems examined there do not apply in this example because the server does not use ILM procedures to store data. Tapes are also not used in the example because tape drives are rather expensive [31] so instead the server uses a simple storage consisting of multiple hard disks in a RAID setup. The reasons for using a RAID setup are explained later in this section.

Components To determine threats to the system it is broken down into smaller components as described in Section 3.5.2. Figure 6.1 shows the basic configuration of the network and each of these components' security can be threatened by various means.

The firewall may be incorrectly configured which mistakenly allows unauthorised access to the storage system or client computers. Depending on which backup scheme and which software is used this could result in attackers being able to impersonate either a server or client system and perform backup creation or restoration to either obtain confidential data from either system or restore forged data to a client. This is prevented by using the authentication system that is built into SSH. It verifies the identity of the server as well as the client when initiating a connection.

The router is pictured in its most basic form in the diagram because the firewall and switch are separate devices which means all it does is directing traffic between networks. The router may fail to route or it may route traffic incorrectly, however, both threats do not impact the backup process because all communication regarding it happens inside the network and never reaches the router.

The switch is more interesting because it connects the clients to the storage server and this connection can be used to intercept traffic between them. An attacker who has a device that is connected to the switch can use ARP Spoofing techniques to intercept traffic sent over the switch. Refer to Section 3.5.3 for details about ARP Spoofing attacks. The switch can also be overloaded with traffic resulting in a DoS situation which means that network communication is no longer possible. This may affect the complete switch or it may only affect certain ports which are connected to specific systems like for example the port connected to the storage system.

The wireless access point is even more interesting than the switch because it only requires physical proximity rather than a physical connection. Attackers might be located in a neighbouring apartment or outside the building on the street yet they can still establish a wireless connection if the signal is strong enough. Threats to wireless network security can be of various nature, like, for example, the network may not employ encryption. This allows attackers to passively intercept the traffic and extract backup data. Other attacks include weak passwords, which will allow an attacker easy access to the network, even when strong encryption and authentication schemes are used. A wireless network's performance can easily be impeded by employing signal jammers, which send noise on the same frequency used by the wireless network to transmit data. Finally, the WiFi access point may be impersonated by an attacker by setting up their own access point with a stronger signal. This works because clients generally connect to the access point with the best signal strength. The access point may also run outdated firmware which contains a security vulnerability that allows an attacker to take over complete control of the device.

A workstation is even easier to attack and provides far greater number of entry points for attackers, but it may also be more vulnerable to other threats because the users is more often than not rather inexperienced. Hof[21] explains that inexperienced users may be more likely to ignore error messages or any kind of error indicator because they may not full understand what the message means and they may not particularly care because for them a computer is just a tool to achieve certain goals like creating a tax report or finding some information on-line. Having to deal with an error message thus means that the need to stop working on whatever it is they were doing and think about what the right action is regarding the error. However, taking correct action is only possible if the user possess sufficient information, but this is difficult to achieve and thus asking the user should be avoided if it cannot be guaranteed that they can make an informed decision. Workstation computers may also be attacked electronically without direct user action for example by exploiting vulnerabilities in browser software with advertisements or drive-by downloads.

The issues raised for workstation computers also apply to laptops, however, due to their portable nature laptops are vulnerable to a few more problems. Mainly laptops can move around not only inside a building and a network, but also between different locations and networks. When the system is used for example in a university location or in a hotel it may also be connected to the public WiFi which may allow attackers easier access than a personal network at home or in the office. A public network is also likely used by more people and thus the number of potential targets is higher which means attackers who do not target any particular user but rather cast a wide net are more likely to attack such networks. Finally, since laptops are more mobile than workstations, they can also become lost or stolen more easily.

Section 3.5.2 already describes attacks on storage systems, however, attackers are not the only threats that have to be considered. Other threats include hardware failure of for example disks and power supplies, but also operator mistakes and natural disasters. These are the same threats that are considered in Section 3.1 as the reasons why backups are necessary in the first place and they obviously apply to any IT system regardless of its purpose.

6.3 Concept Description

When setting up a complex system there is a larger margin for error and as Kernighan and Plauger[22] put it “Everyone knows that debugging is twice as hard as writing a program in the first place. So if you’re as clever as you can be when you write it, how will you ever debug it?“. Under the assumption that this also applies to setting up software and because Schneier[35] explains that complexity harms security, this example concept is limited to rather simple software like tarballs, rsnapshot and duplicity of which duplicity has been selected because it satisfies the requirements described in the rest of this section. Complicated concepts like the Tower of Hanoi backup plan - which is described in Section 4.1 - are also not applied for the same reason.

The following discussion evaluates the requirements defined in Section 4 and determines how they need to be addressed in the context of this example. Table 6.1 provides a summary of the requirements and their solutions.

6.3.1 Backup Process

Backup Storage Locations One of the first questions that needs to be dealt with is where to store the backup data. Section 3.3 discusses different types of storage locations and for this example the network already contains a storage server which will be used. While using cloud storage or other electronic off-site storage is also an option they are not used in this example because internet connections in homes or small offices tend to only provide limited upload bandwidth [31], which can result in long backup times when lots of data is modified by for example a system update.

On-Site Server Storage On-site storage can easily be attached using Gigabit Ethernet (GbE) which provides good bandwidth in both directions. If managed properly, local storage is sufficiently reliable because as discussed in Section 4.2.3 even cloud storage services with high availability guarantees can fail or lose data. To increase reliability and availability of the on-site NAS it uses a RAID setup to protect against single disk hardware failure. Which exact RAID level is used depends on the actual storage space requirements, but for most small businesses and homes, like in this example, a simple RAID 1 array with two disks should provide sufficient space. As long as disk health is monitored and bad disks are replaced swiftly the risk of data loss due to disk failure is assumed to be sufficiently small for normal operation. In case of errors, the backups can be used to restore the system. While other types of hardware failure like power supply failures are harder to protect against, they also pose less of a threat to the data stored on disks. Modern file systems are rather resistant against crashes and tend to at most lose changes to files that were being written to at the time of the crash. As long as backups can be verified and replacement hardware can be obtained in a timely fashion, the impact of hardware failure is rather limited and negligible in this example.

To ensure IT Security and thus data confidentiality, backup data is encrypted. Encryption should happen as early as possible so that when data moves between storage media it does not have to be decrypted and encrypted again. To protect metadata confidentiality, like for example which backups are on a certain piece of storage media, storage media may still employ encryption of their own. However, this is only a bonus requirement because the example only contains a small numbers of systems and distributing backup data across multiple storage media is avoided due to cost and efficiency reasons which means all backups will be on the same storage media.

Off-Site Backup Of course backups should not be solely kept on the on-site storage server because as explained in Section 4.2.3 everything can fail. Therefore, to reduce risks associated with hardware or software failure of the on-site storage server as well as to protect against localised

incidents like a fire in the building a secondary off-site backup needs to be maintained. For cost reasons this secondary backup also comprises hard disks since tapes require a rather expensive drive and optical media does not provide sufficient capacity to store multiple backups of a couple of systems. To ensure availability it will also be set up in a RAID 1 fashion so that even when one disk breaks during storage or transport the backup can be restored.

Since on-line storage services have been ruled out due to network capacity reasons the off-site disks can also not be updated via a network link and thus need to be regularly brought on-site for updates. To minimise potential impact of a localised incident during such an update the off-site backup comprises two separate RAID 1 arrays which at no time are both located on-site. One off-site backup set (one RAID array) is kept on-site and one off-site which allows for the on-site set to be updated and then rotated to the off-site storage facility where it is exchanged for the other set. The other set is then also updated and stored on-site, however, it is not connected to the storage server so that for example a power overload cannot damage it as easily. This means that there is a total of three RAID 1 arrays of which one is permanently on-line and located in the storage server and two are regularly connected to receive updates while at any time at least one of them is located off-site.

In this example the off-site storage facility is a bank safe which at the time of writing costs 76.00 Euro per year. If such a service is not available an alternative is to store the off-site backup copy with a friend or at ones workplace if possible.

Backup Redundancy Redundancy is achieved by storing multiple backup chains and by keeping backups in multiple locations as described above.

Backup Content Any data that is not very explicitly determined to be easily replaceable like for example cached or otherwise temporary data is included in the backup. In particular, this includes the operating system for easy and fast recovery procedures as explained in Section 4.1 Depending on what the machine is used for and what kinds of applications are installed this can easily result in a rather large amount of data that needs to be backed up thus incremental backups are required. While this means that backups more difficult to restore if duplicity is unavailable, this only affects the incremental backups and not the full ones as explained in Section 5.2. Since backups are regularly verified and manual restoration without duplicity is possible, the benefit of lower storage requirements outweighs the drawbacks.

Preferably backups are kept for as long as storage space allows, but at least two months whereby even if problems are not found right away, good backups are available. The software needs to be sufficiently efficient so that keeping backup data for this time is possible with the available storage space.

Backup Creation As mentioned throughout this chapter, backups are created using the duplicity software. The different aspects of the creation process are discussed in Section 6.3.2.

Verification and Testing Backups can be verified by executing a single duplicity command. Restoration testing can also be performed with a single restore command that uses an empty destination directory rather than the root directory.

Backup Recovery Recovery of a backup is similar to restoration testing, but the destination is the root directory. Only the duplicity software, the ssh key and the symmetric encryption password are required for this operation.

6.3.2 Different Aspects of the Backup Creation Process

Trigger (Section 4.2.1) This concept uses a manual trigger, which means it is triggered by the user, and it is assumed that the user runs the backup script once a day if the system had been used that day. Future iterations may switch to a time based trigger as described in the outlook in Section 7. To prevent backups from not being created for an extended period, for example because the user forgets about them, the server monitors the data directories.

Full and Incremental/Differential Backup (Section 4.2.2) Duplicity supports full and incremental backups and handles the complexity associated with incremental ones. Thus it provides the benefit of saving space while maintaining usability.

Storage Location Diversity (Section 4.2.3) Storage location diversity is achieved by maintaining backups on the local NAS and in a bank vault as explained in Section 6.3.1.

Source Data Consistency (Section 4.2.4) It is important that source data is consistent in the backup so that when the backup is restored all application will work as expected and will not be running with inconsistent data. This is especially important for the databases that are running on the workstations as explained in Section 4.2.4. Duplicity does not by itself support creating backups of unstable sources, but this is not too big an issue on workstation computers because it can also be achieved by simply not using the system while the backup is running. See the outlook in Section 7 on how to better address this issue in a future iteration.

Backup Data Storage Requirements (Section 4.2.5) Since the number of off-site volumes is low, simply numbering the two sets is sufficient. Expandability is not expected to be an issue since all storage is set up in a RAID that can be expanded transparently. Availability of old backups data is achieved by moving data between disks when switching to newer or larger ones. Deduplication is already provided by the duplicity software. Assured deletion is not required because no untrusted third-party storage solution is used.

Server Pull and Client Push (Section 4.2.6) As discussed in Section 4.2.6, the push scheme is better suited for a network with workstations and thus this scheme is used and duplicity has been chosen because it supports client push.

Transport Security (Section 4.2.7) Confidentiality during transport of the on-site backup can be satisfied by transferring data via encrypted connections like for example by using SSH. Ensuring confidentiality during transport of the physical media is more complicated, however, it can be solved by using Linux' dm-crypt or Linux Unified Key Setup (LUKS) functionality to encrypt all content of the hard disks with a symmetric key. This key should be memorised and it should be rather long to provide sufficient strength.

As explained in Section 3.4, encryption should not be the only line of defence. The backup data is only transported in a local network which is protected by a firewall and physical protection of the office building. The physical media, which is stored in the bank vault, is transported by a trusted courier. Encryption is thus an additional layer in the event that the others are compromised.

Speed (Section 4.3.1) Due to the way in which source data consistency is handled the backup creation speed is more important than it would be otherwise because the system cannot be used

while the backup is running. This means it is important for the process to be somewhat predictable, especially regarding the creation of full backups which can take considerably longer than incremental ones (see Section 6.4. This is handled by only creating full backups once a week on Monday and if no full backups are created for 4 weeks then creation of a full backup is done at the next run regardless which day it is.

Impact of the Backup Process on the System (Section 4.3.2) As explained above, to ensure data consistency the system may not be used during backup creation. This is possible - in this example - because the system only has to be usable during business hours, although it may be unavailable otherwise. Especially, it is allowed to be unusable during the night or during the lunch break, which in this example is assumed to last 45 minutes. As shown in Section 6.4 duplicity supports this requirement.

Table 6.1: Summary of the requirements and their solution

Requirement	Solution
On-Site Storage	Local NAS
Off-Site Backup	Hard disk in bank vault
Backup Redundancy	Provided by on-site and off-site backups
Backup Content	Everything except carefully selected, temporary files
Backup Creation	Custom script using duplicity
Verification and Testing	Provided by duplicity
Backup Recovery	Provided by duplicity
Trigger	Manual with server-side monitoring
Full and Incremental Backups	Provided by duplicity
Storage Location Diversity	Provided by on-site and off-site backups
Source Data Consistency	System can not be used during backup creation
Volume Organisation	Simple numbering sufficient
Expandability	RAID can be expanded transparently
Availability	Old data is migrated to new disks
Deduplication	Provided by duplicity
Assured deletion	Not required
Pull/Push	Push is more secure and the only solution supported by duplicity
Transport Security	Connections secure with SSH; physical media secured with LUKS
Speed	Duration predictable; fast enough for lunch break
Impact on the System	System cannot be used during backup creation

6.4 Review and Testing

The script which implements this backup process can be found in Appendix A.1. During testing the following two issues emerged:

- Sometimes removable media or network shares are mounted on the system when the backup is being created and such file systems need to be detected and excluded. While excluding data from the backup is not recommended for the reasons explained in Section 4.2.1, it may still be desirable in cases like this one. Not excluding network shares would result in the content of those shares being backed up once by the server that provides the network share and once by each system that mounts it. This would result in a huge number of backups for any given network share and thus waste lots of space. However, additional dedicated file

systems for data of the system itself should not be excluded which means that the duplicity option `--exclude-other-filesystems` cannot be used. Instead an exclusion list has to be passed, but keeping such a list up to date is error prone. To reduce potential for mistakes, the script checks if all active mount points are either of a whitelisted file system type or part of the exclusion list.

- While duplicity supports verification of backups against the source file system this feature is only partially helpful because the source file system is not static. Some files will still change even if all applications are closed to minimise modifications to the source file system. Files that change include the shell history which records the command used to run the backup and the system log files which record that root privileges have been used to run it. Such false-positives need to be looked for manually in the resulting verification output. This can be improved in a future iteration as explained in Section 7. However, integrity of the backup data itself can be verified because this type of verification does not compare backup data to the source file system, but rather uses checksums.

The workstation used for testing runs an Arch Linux system on a SuperMicro X10SAE mainboard combined with an Intel Xeon E3-1230v3, 32GB ECC RAM and a Samsung 840 EVO 256GB SSD. The block device for the file system is encrypted using LUKS with the aes-xts-plain64 cipher specification and a 512bit key. The storage server also runs Arch Linux with a ZFS on Linux (ZOL) RAID 10 over 4 3TB 7200rpm hard drives and is connected with GbE to the workstation. The tests below have all been run three times and each time the fastest has been selected to reduce the impact of outside factors.

Creating a full backup of approximately 48.5GB of data takes 34 minutes and 3 seconds which means source data is backed up at a speed of approximately 24.3MiB/second and the space used on the storage server is only 28.4GB which is approximately 58 percent of the original data. While this is a rather long time it is fast enough to be run during a lunch break and thus is not too much of an issue even if the system cannot be used during the backup creation. Creating the first incremental backup takes just under 5 minutes and after having used the web browser (Firefox and Chromium) and email client (Claws-mail) for a few minutes the total size of changed files is already at 731MB, yet duplicity manages to store all the changes in just 7MB on the storage system.

Verification of a full and one incremental backup including verification of the data against the file system takes 25 minutes and 15 seconds. However, as explained above, some files change between runs and thus create undesired warnings. Verification of just the backup volumes - without also comparing the backup data to the source data - takes 22 minutes and 45 seconds. Since both operations are only run occasionally rather than every time the backup is created and since at least the second can run in the background during normal operation, the amount of time taken is acceptable although times will rise the more backups are created.

Due to space restrictions, restoration testing uses a dedicated Samsung 840 EVO 120GB SSD with the same encryption settings as the main storage device. Restoring the backup created previously with one full and one incremental backup run takes 24 minutes and 12 seconds.

The cron job which alerts when no backups have been created for some time works as expected.

7 Conclusion and Outlook

This bachelor's thesis explains the reasons for making backups, describes the different types of backups like full, incremental and differential backups and discusses how a backup process is structured. Since backups contain lots of important data in a central location, their security is highly important so issues of ensuring confidentiality, integrity and availability are explored in combination with processes that help to determine threat models for actual systems and concepts. Backups are only useful if they can be used to restore a broken system, which requires that verification, recovery planning and testing are also examined.

Additionally, some important aspects of the backup creation process are discussed, including source data consistency, which means that not only the source file system needs to be backed up in a way that ensures that it is not modified during the backup creation and are thus is consistent within itself, but also different services or different file systems are consistent with each other. Other aspects are data storage requirements, which may require data deduplication, yet also require to keep some redundancy to increase availability in case of simple disk failures. Differences between having a server pull backup data from clients and having the clients push their data to a server are also examined. This discussion shows that there are important security considerations for both cases and backup concepts need to adapt accordingly.

Finally, some selected examples of software for backups are examined and a backup concept for a small network in a small business setting is developed utilising the aspects, techniques and processes explained before. The example network is described, requirements for the backup concept are evaluated and a threat model is determined. Finally, a backup software that fits with these requirements is selected and implementation and testing of the concept shows that it works as expected.

This work thus provides and overview over the basics of backups as well as different aspects of a backup concept. It also shows how this information can be used to develop a backup concept in practice and hopefully, this work can serve as an entry point into the world of backups and help inexperienced users to make the right decisions.

Outlook The time of restoration of a backup - as given in Section 6.4 - is remarkably similar to that of verification, which likely means that both actions share the same bottleneck. During restore duplicity uses 100 percent of one CPU core and the SSD does not appear to be particularly heavily utilised. This likely means that the process is not limited by storage or network performance, but rather by CPU performance and lack of multithreading, however, the exact bottleneck is not explored further in this thesis and is left for future research.

Section 6.4 explains that verification of backup data against the source file system is not reliably possible in an automatic fashion and requires manual inspection. Another potential annoyance is that the requirements allow that the system is unusable during backup creation and that backups may be created manually. A possible solution to these problems is to use file system snapshots as a source for the backup creation rather than using the live file system, however, implementing this may not be trivial because for example the ext4 file system itself does not support snapshots and some systems use multiple file systems, which need to be detected and handled correctly. Possible solutions include LVM and alternative file systems like B-tree file system (BTRFS) and Z File

System (ZFS), although a discussion of the benefits and shortcomings of these and other solutions lies beyond the scope of this work.

While the backup employs encryption to protect confidentiality, it does not take measures to ensure that backup data cannot be modified by a third party that possesses the symmetric key. Adding protection against such incidents may be accomplished for example by storing information about verified backups at a secure location or by using digital signatures. The issue of how to securely store information about verified backups or key files that are necessary for digital signatures can be explored in future research.

Bibliography

References

[1] Hussam Abu-Libdeh, Lonnie Princehouse, and Hakim Weatherspoon. „RACS: a case for cloud storage diversity“. In: *Proceedings of the 1st ACM symposium on Cloud computing*. ACM. 2010, pp. 229–240.

[2] James M Anderson. „Why we need a new definition of information security“. In: *Computers & Security* 22.4 (2003), pp. 308–313.

[3] Giuseppe Ateniese et al. „Scalable and efficient provable data possession“. In: *Proceedings of the 4th international conference on Security and privacy in communication netowrks*. ACM. 2008, p. 9.

[4] Robert D Austin. „The effects of time pressure on quality in software development: An agency model“. In: *Information Systems Research* 12.2 (2001), pp. 195–207.

[5] Steven M Bellovin. *Firewall-Friendly FTP*. RFC 1579. RFC Editor, 1994. URL: http://www.rfc-editor.org/rfc/rfc1579.txt.

[6] Suparna Bhattacharya et al. „Coordinating backup/recovery and data consistency between database and file systems“. In: *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM. 2002, pp. 500–511.

[7] Donghoon Chang et al. *Rig: A simple, secure and flexible design for Password Hashing Version 2.0*. 2014.

[8] George Chang et al. „A novel approach to automated, secure, reliable, & distributed backup of mer tactical data on clouds“. In: *Aerospace Conference, 2012 IEEE*. IEEE. 2012, pp. 1–7.

[9] Thawatchai Chomsiri. „Sniffing packets on LAN without ARP spoofing“. In: *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*. Vol. 2. IEEE. 2008, pp. 472–477.

[10] Tim Dierks and Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. RFC Editor, 2008. URL: http://www.rfc-editor.org/rfc/rfc5246.txt.

[12] Jon G Elerath and Michael Pecht. „Enhanced reliability modeling of raid storage systems“. In: *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*. IEEE. 2007, pp. 175–184.

[13] Rebecca Fallon. „Celebgate: Two Methodological Approaches to the 2014 Celebrity Photo Hacks“. In: *Internet Science*. Springer, 2015, pp. 49–60.

[14] Simson Garfinkel, Gene Spafford, and Alan Schwartz. *Practical UNIX and Internet security*. " O'Reilly Media, Inc.", 2003.

[17] Martin Glinz. „On non-functional requirements“. In: *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*. IEEE. 2007, pp. 21–26.

[18] Barbara Guttman and Edward A Roback. *An introduction to computer security: the NIST handbook*. DIANE Publishing, 1995.

[19] Ragib Hasan et al. „Toward a threat model for storage systems“. In: *Proceedings of the 2005 ACM workshop on Storage security and survivability*. ACM. 2005, pp. 94–102.

[20] Jaap-Henk Hoepman and Bart Jacobs. „Increased security through open source“. In: *Communications of the ACM* 50.1 (2007), pp. 79–83.

[21] Hans-Joachim Hof. „Towards Enhanced Usability of IT Security Mechanisms - How to Design Usable IT Security Mechanisms Using the Example of Email Encryption“. In: *International Journal On Advances in Security* 6.1&2 (2013), pp. 78–87. URL: http://arxiv.org/abs/1506.06987.

[22] Brian W Kernighan and Phillip James Plauger. *The elements of programming style*. McGraw-Hill, 1978.

[23] Mark Lillibridge et al. „A cooperative internet backup scheme“. In: *Proceedings of the annual conference on USENIX Annual Technical Conference*. USENIX Association. 2003.

[24] Chih Wei Ling and Anwitaman Datta. „InterCloud RAIDer: A do-it-yourself multi-cloud private data backup system“. In: *Distributed Computing and Networking*. Springer, 2014, pp. 453–468.

[25] Ray McGoldrick and Richard Shin. „The Weakest Link in Corporate Backup Processes“. In: *Document News* 23.4 (2005), p. 2.

[26] Daniel Mellado and David G Rosado. „An overview of current information systems security challenges and innovations J. UCS Special Issue“. In: *Journal of Universal Computer Science* 18.12 (2012), pp. 1598–1607.

[27] Jaehong Min, Daeyoung Yoon, and Youjip Won. „Efficient deduplication techniques for modern backup operation“. In: *Computers, IEEE Transactions on* 60.6 (2011), pp. 824–840.

[28] Suvda Myagmar, Adam J Lee, and William Yurcik. „Threat modeling as a basis for security requirements“. In: *Symposium on requirements engineering for information security (SREIS)*. Vol. 2005. 2005, pp. 1–8.

[29] Donn B. Parker. „Toward a New Framework for Information Security?“ In: *Computer Security Handbook*. John Wiley & Sons, Inc., 2012.

[30] David A Patterson, Garth Gibson, and Randy H Katz. *A case for redundant arrays of inexpensive disks (RAID)*. Vol. 17. 3. ACM, 1988.

[31] Curtis Preston. *Backup & recovery: inexpensive backup solutions for open systems*. O'Reilly Media, Inc., 2007.

[32] Arthur Rahumed et al. „A secure cloud backup system with assured deletion and version control“. In: *Parallel Processing Workshops (ICPPW), 2011 40th International Conference on*. IEEE. 2011, pp. 160–167.

[33] Charles Reis, Adam Barth, and Carlos Pizano. „Browser security: lessons from google chrome“. In: *Queue* 7.5 (2009), p. 3.

[34] Yu Sasaki and Kazumaro Aoki. „Finding preimages in full MD5 faster than exhaustive search“. In: *Advances in Cryptology-EUROCRYPT 2009*. Springer, 2009, pp. 134–152.

[35] Bruce Schneier. *Secrets and lies: digital security in a networked world*. John Wiley & Sons, 2011.

[36] Mark W Storer et al. „Secure data deduplication“. In: *Proceedings of the 4th ACM international workshop on Storage security and survivability*. ACM. 2008, pp. 1–10.

[37] Roman Strobl. „Zfs: Revolution in file systems“. In: *Sun Tech Days* 2009 (2008), p. 2008.

[38] Guo-Zi Sun et al. „Data backup and recovery based on data de-duplication“. In: *Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on*. Vol. 2. IEEE. 2010, pp. 379–382.

[39] László Toka, Matteo Dell' Amico, and Pietro Michiardi. „Online data backup: A peer-assisted approach“. In: *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*. IEEE. 2010, pp. 1–10.

[40] Avishay Traeger et al. „Using free web storage for data backup“. In: *Proceedings of the second ACM workshop on Storage security and survivability*. ACM. 2006, pp. 73–78.

[41] Sean Turner and Lily Chen. *Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms*. RFC 6151. RFC Editor, 2011. URL: http://www.rfc-editor.org/rfc/rfc6151.txt.

[42] Ruofan Xia et al. „Performance and Availability Modeling of ITSystems with Data Backup and Restore“. In: *Dependable and Secure Computing, IEEE Transactions on*. Vol. 11. 4. IEEE, 2014, pp. 375–389.

[43] Tianming Yang et al. „FBBM: A New Backup Method with Data De-duplication Capability“. In: *Multimedia and Ubiquitous Engineering, 2008. MUE 2008. International Conference on*. IEEE. 2008, pp. 30–35.

[44] Yinqian Zhang et al. „Homealone: Co-residency detection in the cloud via side-channel analysis“. In: *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE. 2011, pp. 313–328.

[45] Zhao Zhongmeng and Yao Hangtian. „A Data Backup Method Based on File System Filter Driver“. In: *Software Engineering (WCSE), 2010 Second World Congress on*. Vol. 2. IEEE. 2010, pp. 283–286.

Online References

[11] *Duplicity website*. URL: http://duplicity.nongnu.org/index.html (visited on 10/11/2015).

[15] German Federal Office for Information Security (BSI). *Hinweise zur räumlichen Entfernung zwischen redundanten Rechenzentren*. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Hilfsmittel/Doku/RZ-Abstand.pdf?__blob=publicationFile (visited on 10/26/2015).

[16] German Federal Office for Information Security (BSI). *IT-Grundschutz-Katalog*. URL: https://gsb.download.bva.bund.de/BSI/ITGSK/IT-Grundschutz-Kataloge_2014_EL14_DE.pdf (visited on 02/20/2016).

A Appendix

This appendix contains the backup script that has been used to obtain the test data in Section 6.4.

A.1 backup.sh

```
#!/bin/bash
#
# This is a simple backup script using duplicity.
#
# Important steps:
#  - define a host "backup" in root's .ssh/config
#  - read through the configuration section below and adjust to your needs

###
# Configuration starts here
###

# backup storage location of the server
backupdir="$HOSTNAME-backup/full-backup-thesis"

# Time for which to keep backups
retentionTime="120D"

# Symmetric encryption passphrase for the backup data
PASSPHRASE="CHANGEME set some random or fancy password here CHANGEME"

# An arbitrary exclude/include list. The backup will contain everything
# starting with / except for things excluded here. Mountpoints should be
# excluded by using the $excludeMountpoints setting below.
# The first line that matches wins. For details on the format refer to the
# duplicity manpage.
IFS='' read -r -d '' excludeList <<EOF || true
+ /home/flo/.local/share/Steam/steamapps/common/Counter-Strike Global Offensive/csgo/cfg
- /home/flo/arch/iso/work/*
- /home/flo/arch/iso/out/*
- /home/flo/tmp/*
- /home/*/.local/share/Steam/steamapps/common/*/*
- /home/*/.cache/*
- /home/*/.claws-mail/imapcache
- /root/.cache/*
- /home/arch.fs
- /var/cache/pacman/pkg/*
EOF

# List of mountpoints that should be excluded from the backup. Entries have to
# end with a slash. The content of this list is appended to $excludeList
excludeMountpoints=(
/tmp/
/sys/
/dev/
/proc/
/run/
/mnt/mistral/flo/
/mnt/ostro/torrent/
/mnt/levant/nfs/
/mnt/caju/flo/
/mnt/mistral/srv/
/mnt/chroots/arch/
/media/
)
```

```
56
57  # mountpoints of these types do not need to be excluded in $excludeMountpoints
58  fsWhitelist=(ext4 btrfs)
59
60  ###
61  # Configuration ends here
62  ###
63
64  set -e
65
66  main() {
67          if [[ $UID != 0 ]]; then
68                  echo "Error: Root permissions required" >&2
69                  exit 1
70          fi
71
72          if ! type duplicity &>/dev/null; then
73                  echo "Error: duplicity is not installed" >&2
74                  exit 1
75          fi
76
77          TMPDIR="$(mktemp -d "/tmp/${0##*/}.XXXXXX")"
78          trap "rm -rf '${TMPDIR}'" EXIT TERM
79
80          # ensure duplicity keeps its cache at a central location
81          export HOME=/root
82
83          printf "Starting backup. Do NOT use the system until this process is complete!"
     ↪  >&2
84
85          exclude_mountpoints
86          echo "$excludeList" > "$TMPDIR/exclude-list"
87
88          # save some data that's useful for restores
89          local backupDataDir=/root/backup-data/
90          mkdir -p "$backupDataDir"
91
92          # Partitioning information is useful to recreate partitions with suitable
93          # sizes for the data to be restored and avoids trial and error.
94          fdisk -l > "$backupDataDir/fdisk"
95          df -a > "$backupDataDir/df"
96          findmnt -l > "$backupDataDir/findmnt"
97          if type vgdisplay &>/dev/null; then
98                  vgdisplay > "$backupDataDir/vgdisplay"
99                  pvdisplay > "$backupDataDir/pvdisplay"
100                 lvdisplay > "$backupDataDir/lvdisplay"
101         fi
102
103         backup / "sftp://backup/$backupdir/" --exclude-filelist "$TMPDIR/exclude-list"
104         ssh backup "touch $backupdir/last-backup-timestamp"
105
106         printf "Backup complete. You can now use the system again." >&2
107 }
108
109 backup() {
110         local src=$1
111         local dest=$2
112         shift 2
113         local -a options=()
114
115         # try to only run full backups on monday (weekday 1)
116         if [[ $(date +%u ) == '1' ]]; then
117                 # Use 2D to only create one full backup per day. Otherwise multiple
118                 # runs on the same day might create multiple full backups.
119                 options+=(--full-if-older-than 2D)
120         else
121                 # force a full backup after 4 weeks
122                 options+=(--full-if-older-than 28D)
123         fi
124
125         export PASSPHRASE
126         duplicity \
```

```
127                 --log-file /var/log/backup.log \
128                 -v5 \
129                 --numeric-owner \
130                 --volsize 250 \
131                 --allow-source-mismatch \
132                 --asynchronous-upload \
133                 "${options[@]}" "$@" "$src" "$dest"
134
135         duplicity --log-file /var/log/backup.log --force remove-older-than
    ↪   "$retentionTime" "$dest"
136         #duplicity -v5 --compare-data --log-file /var/log/backup.log "$@" verify "$dest"
    ↪   "$src"
137         export PASSPHRASE=""
138 }
139
140 ### support functions below ###
141
142 ##
143 # Check if an array contains a certain element
144 #
145 #  usage : in_array( $needle, $haystack )
146 # return : 0 - found
147 #          1 - not found
148 ##
149 in_array() {
150     local needle=$1; shift
151     local item
152     for item in "$@"; do
153         [[ $item = "$needle" ]] && return 0 # Found
154     done
155     return 1 # Not Found
156 }
157
158 # same as in_array except 0 is returned if any item in haystack starts with needle
159 in_array_startswith() {
160     local needle=$1; shift
161     local item
162     for item in "$@"; do
163         [[ "$needle" == "$item"* ]] && return 0 # Found
164     done
165     return 1 # Not Found
166 }
167
168 # Add excluded mountpoints to the exclusion list and check that non-whitelisted
169 # mountpoints are excluded.
170 exclude_mountpoints() {
171         local error=0
172
173         for fs in "${excludeMountpoints[@]}"; do
174                 if [[ $fs != */ ]]; then
175                         error=1
176                         echo "Error: excludeMountpoints entry doesn't end with /: $fs"
    ↪   >&2
177                 fi
178                 excludeList+="- $fs*"$'\n'
179         done
180
181         while read line; do
182                 local mountpoint=$(echo "$line" | cut -d\  -f2 | sed 's#\040# #g;')
183                 local type=$(echo "$line" | cut -d\  -f3)
184
185                 if ! in_array "$type" "${fsWhitelist[@]}"; then
186                         if ! in_array_startswith "$mountpoint/"
    ↪   "${excludeMountpoints[@]}"; then
187                                 error=1
188                                 echo "Warning: mountpoint not excluded: $mountpoint" >&2
189                         fi
190                 fi
191         done </etc/mtab
192
193         if ((error)); then
194                 exit 1
```

```
195             fi
196     }
197
198     main "$@"
```