# All.Net Analyst Report and Newsletter

***Welcome to our Analyst Report and Newsletter***

## *Build Better Backups*

About a year ago, “Build Backups Better”[1] was published. Since then, I have redone our infrastructure and rethought our backup strategy and tactics. I thought I would provide a suitable update, especially since its coming up in the end of the year when annual backups are hopefully being done.

On this rare occasion, there is so much being covered, I have decided to provide a table of contents. This is how I eventually end up writing books instead of short articles...

# Table of Contents



1 https://all.net/Analyst/2025-01.pdf

  

## *Our organizational objectives*

A backup approach is largely a function of protection objectives in the context of a control architecture.[2] In our case, we find the critical objectives to be integrity and availability, which is to say that we care less about use control, accountability, transparency, custody, and confidentiality for most of our content. Our content's utility lies in your ability to gain access to it at any time and from any place in the form we chose to make it available. It also has internal utility in making us more efficient and effective at doing our work than competitors trying to do similar things. Assuring these are; reasonable mechanisms for change control and access facilitation; more serious mechanisms for the control scheme, perimeters, and functional units; all organized based on a trust architecture[3] and operated according to a cognitive scheme.[4]

As a fundamental decision, we want our backup and recovery systems to be architecturally protected from any outside force intentionally injecting code or content or inducing unauthorized behaviors. Physical protection separating backup systems from other systems and people is adequate according to our internal requirements.

## *Protection objectives*

As long as the interpretation mechanisms on the backup systems are implemented in a proper limited function manner and they don't interpret information from the systems under backup in a general purpose fashion, the backup system itself can be safe from corruption by the systems it backs up and restores to. However:

- **Leakage:** The transport mechanisms; secure shell, secure socket layer, or whatever transport mechanisms are used; can be subverted at the target end and produce things like data leakage, for example, by restoring from backups that had information no longer available on the target system and leaking it to the outside world.
- **Denial of services** can be attained to a limited extent by feeding too much information to the backup system. To counter this, things like space allocation can be controlled to limit the ability to disable the entire backup system by running it out of resources for storage or communication.
- **Corruption** cannot really be pushed into the backup system because it retains copies of every version of historical information backed up through the archival mechanisms and because it only operates on the software embedded in it and is not effected by incoming data in terms of its operational mechanisms. Restoration of corrupted versions after they are backed up are, of course, properly restored as corrupted.
- **Accountability** is maintained by the backup system to the extent the information it pulls reflects accountability information. The backup system also maintains independent records fo what it did for accountability purposes and can transmit it as recovery information to authorized backup targets.
- **Transparency** is maintained by the provisioning of status information and other information provisioned for automatic recall on demand or periodically. The configuration provides the desired level of transparency, and the backup system can be implemented to include whatever information is desired.
- **Custody**: The backup system maintains custody of everything it backs up and custody is assured through the lack of external influences, the redundant copies, and how they are physically controlled by the entity implementing the system. Additional controls are provided by the archival mechanisms of the unified backup mechanisms which track and access files by cryptographic checksums and file lengths.
- **Use control** is controlled by the lack of ability to cause any use other than those programmed into the backup system and the lack of channels for inbound initiated communication.

Obviously, there has to be an administrative process by which the system is controlled, to the extent such controls are allowed. And there is physical access because the systems are physical systems.

2 https://all.net/Arch/index.html
3 https://all.net/Analyst/2018-07.pdf "The basis for trust"
4 https://all.net/Analyst/2018-10.pdf "The Cyber Security Brain"

  

## *Our network structure*

The structure of our network, from most distant to the castle keep, is provided in this table:

| Area | Description |
|---|---|
| Cloud | Servers supporting external users are the face of the companies and provides all externally usable services. Firewalled at Cloud Provider, load adaptable (not balancing) and cold standby (30 minutes) |
| Internet | The IT world not our Cloud presence or internal systems |
| Providers | Redundant providers for bits over the wire and in the air, also provide services like television. |
| Yellow | Internal only including guest facility access, printers, and other externally controlled computing devices, including externally updated WiFi – NAT to Internet |
| Green | Internal trusted area which allows only systems configured in-house with Linux operating environments – NAT to Yellow – NAT to Internet – Primary, Secondary (hot standby and surveillance) user systems, Cold standby user systems (1-10 minutes to op) |
| Red | Backups and repository NAT to Green – temporary only physical connections for Setup,. Control, Verification, Maintenance, and Recovery |
| Black | Physically separate never connected to the Internet or anything connected to it. Internal R&D only |

***Our network structure***

It should go without saying that your network structural approach will be critical to your backup and restoration approach. That's why I say it here… because things that should go without saying are the assumptions used to defeat systems.[5]

## *What Backs up What, When, and How*

**Cloud:** Backups of Cloud data from external users and audit data is done by pulls from Primary (ad-hoc) and Backup (periodic automatic more than hourly synchronization without deletion) servers. The automatic version assures maximum data loss is minimal (usually not more than 15 minutes) but also that old versions remain unaltered and recoverable in case of corruption. Non-user data (content, software, controls,etc.) is driven by internal systems only, so that it is backed up and restored from the sources (in Green). Thus no attack on the Cloud can corrupt or make unavailable (to us) anything but user data and logs, and these can be restored within about 15 minutes of loss. The cloud systems with user data also do versioning so that every version of user data saved by a user is stored in a time and date stamped file. With the no deletion of old data upon backup, intentional (or accidental) destruction or alteration of old content on the server does not corrupt the backups. Cloud cold standby is periodically updated from the Backups during short maintenance periods and if it is invoked (which has happened once in the last 15 years for a few hours), so it cannot be corrupted during normal operation (because it is turned off) except by the cloud service provider (CSP) screwing up. And of course if it is corrupted, the restoration should fix it (unless they really screwed up), and another one can be created in a few hours from scratch if necessary.

**Yellow:** No internal user systems are on Yellow. Printers and similar things are not backed up. We pretty much don't care about them other than getting printed output, and since confidentiality is not a substantial issue, if someone should get into them or their remote update mechanisms should corrupt them we still largely don't care. Same with surveillance… nothing on that network is able to surveil or access anything on the Green network, which is what we care about from a corporate perspective. Assume we have also physically separated them… except for the NATs from Green to Yellow. Also note that to prevent mistakes, all connections other than WiFi are over colored cabling of the same color as the network name. So all Yellow resources have yellow cables attached to them and NAT gateways have the two colors of the two networks plugged in, with the 'WAN' side of Yellow to White (external) being white and the LAN side being yellow. A mis-wiring immediately causes a

5 https://all.net/Analyst/2021-03C.pdf "How to defeat any system"

  

network failure on any attempt to connect to the Internet, and the addressing scheme also isolates the networks so that the NAT will not recognize the traffic on either side unless it has the right addressing. Yellow is similar to classic demilitarized zones (DMZs) discussed in typical firewall documentation, but does not proxy inbound.

**Green:** Internal only users on this network, with no inbound connectivity except from the Red Backup network. Primary backs up to Secondary on an ad-hoc basis (we don't want it automatic because when we make and test changes, we want the last working versions on the Secondary systems available for restore, AND we want the secondary systems fully operational as hot standby and as change control (reversion) measure. Except, of course, as change control, it works the opposite of standard change control[6]. The Primary is in use and changes are made on the Primary then tested there and only propagated to the Secondary (hot standby) once they pass the tests. So we take the more risky but rapid approach to development and operations while still assuring reasonable recovery capabilities. Also, both Primary and Secondary are backed up from the Red backup systems so, at worst, only limited loss is going to happen. Cold standby system(s) are not powered on in normal operation, but can be turned on and fully operational and up to date in a few minutes. We do this by our storage scheme. In essence, each system has a nearly identical configuration, but no user data. The user data is always on a separate disk that is physically independent of the system using it. Unplug the disk from a system, plug it into another system, power up the system, and you are back up. For the higher surety crowd, a Faraday cage containing these unpowered systems may even be effective against electromagnetic pulse (EMP) attacks.[7]

**Red:** In this network, no programs are run from external content. Once configured and operating, the content backed up from wherever it comes from is placed in locations separate from the operating software on the Red network systems. Red systems can reach out through the NAT to pull content from other systems, but the NAT prevents inbound connections from Green, and all other connections are made through multiple NATs (Green to Yellow to Internet) that similarly limit initiation from the less trusted network. A physical connection to control the Red systems is available and plugged into a cold standby system (like other cold standby Green systems with no user disk attached) for maintenance as/if desired. This is where the backup processes are operated:

- Cloud user data is typically backed up (changes only) at least hourly (15 minutes is the goal, but the world does not always cooperate).
- Yellow network systems are not backed up. They contain no user data and are considered utilities.
- Green network systems are backed up daily – first the secondary, then the primary
- Backups from other networks are internally backed up (disk to disk, changes only) daily, weekly, and monthly (on separate disks).
    - Green systems are backed up so they can be restored using a cold standby in a few minutes by physically taking a backup disk to the hot standby and plugging it in then powering it on.
- The Unified historical archive discussed in "Build Backups Better" gets a copy of the current backups after they finish the monthly backup process. Over the next however long it takes (often a few days to a week depending on specifics) the Unified system is updated and consolidated into a repository that can be used for historical access and as a last resort if all else fails.
- The historical archive is then made available read-only on the backup system.
- Meanwhile, the monthly backups are copied to separate disks which are connected once a month (one at a time), get copies for the fire safe, and are disconnected.

**Black:** This network is physically separated from the world with no connections to other networks. It is for internal R&D and testing only, is not backed up by the backup system (although it may do internal backups and disks may be attached to the Red network for transport to the Backup system as/if desired).

## *Restoration*

Restoration is the purpose of backups systems, and operates in a more ad-hoc fashion. It is rare indeed for a restoration from the Red network to be needed because of the cold standby Secondary system being so

6 https://all.net/books/virus/SCVirusBook.pdf Section 3.3.2 (p58-59) F. Cohen, "A Short Course on Computer Viruses, 1990, ISBN# 1-878109-01-4

7 See https://en.wikipedia.org/wiki/Electromagnetic_pulse for some more details.

  

effective. Also, Primary and Secondary Green systems can perform restoration to Cloud infrastructure, making the need for restoration from Red systems unnecessary. While Red can connect to the Internet for this purpose, in practice, restoration from Red to Green is the only thing ever done. Green to Cloud is then used for restoration of Cloud systems. There are two paths to restoration; Physical from Red or Requested from Green.

- Physically attaching a cold standby without user disk to the Red network, a backup administrator can find what has to be restored and restore it by pushing a copy to a Green system.
- Requested restorations can be invoked from Green systems by placing a request where the Backup system can find it and act on it. More on this later...

The Primary, once it is certain it has what is needed, can then push to the associated Green Secondary (hot standby) and both will be automatically backed up as current. Backup disks can also be used in cold standby systems for fast restore to previous backed up state or older stored states. But restores use another path.

## *The NATs*

The NATs are what allows this to be safely controlled and are a critical component of the protection against remote corruption of Green or Red systems. There are several NATs involved, and they are separate and different to increase surety of integrity of backed up content with rapid restoration. Recall NATs are from Internet to Yellow, Yellow to Green, and Green to Red:

- Internet to Yellow has an ISP provided NAT or a purchased NAT that connects to the ISP. It is considered untrusted in the sense that an external actor from the ISP can bypass it, and these tend to allow IPv6 to pass, which is inherently problematic for effective separation. After that NAT, there is an internal NAT that provides WiFi for guests and non-Green users. That is a separate technology from a different source, currently from about 15 years old, that blocks IPv6 and similar problematic protocols.
- Yellow to Green runs a different NAT device, a Linux box internally created and configured as a firewall with NAT gateway. It does not do automatic updates or communicate with the Internet other than to pass packets between zones. Internal firewall rules also prevent anyone who would get access to it from getting any packets out other than from and to NAT interfaces.
- Green to Red runs a commercially available NAT gateway that internally runs a different and more restricted open source Linux operating environment. It does not allow IPv6, and does not go out to the Internet for updates or other things.

There are also other internal separations for subzone network segmentation as/if desired, but the Backup mechanisms for Red is able to access each subzone in the Green network that requires backups.

The use of independent NAT gateways is one of the strongest mechanisms for separation and traffic control, if it is properly configured and operates independently of other systems and external forces once configured. It allows only one side to initiate traffic, and as long as the mechanisms that operate don't allow external infiltration or subversion, it is effective. These are also quite inexpensive (less than $50 in quantity 1).

## *Controlled configurations*

A critical part of the success of this scheme is controlled configurations. That includes Primary and Secondary systems, but in the end, these are user systems, and can and/or will eventually be subverted for a period of time until regenerated (which we do perhaps once a year unless something seems off, in which case we will switch the Primary and Secondary, do a rebuild of the old Primary from physically separated boot media, restore or verify user content, and get the old Primary operating as a hot standby for the new one in about an hour). Depending on particulars, we may then make the old Primary the Primary again and do a restore of the Secondary. By proper process, we limit even a persistent virus or threat actor, once detected, from maintaining a permanent presence. By doing this periodically (or sporadically and unpredictably), even an undetected advanced persistent threat actors have to find a way back in now and then.

As a minor note on our configurations, our computers are laptops (with built-in uninterruptible power from the battery) and we keep user file systems on a separate mountable disk (usually a USB drive, typically 4 Gigabytes either spinning or solid state). This is particularly useful for reconstituting systems very quickly. We have cold standby systems (usually the previous Primary and Secondary computers) already loaded with standard

  

baseline software from their last used state or different older versions of platforms loaded from scratch and configured with our standard configuration process. For rapid recovery, pull the user disk from the problem machine, plug it into a cold standby, power up the standby, attach network cables, and you are up and running in 1-5 minutes. In practice, we have only every needed the hot standby except for testing.

## *Status information and 'remote control' of the backup system*

Getting status information on backups and requesting recovery can all be done from systems under backup, but the mechanisms are a bit different from the way normal computer interactions tend to work. The mechanism is the use of files on target systems (a.k.a. systems under backup) as the messaging mechanism. Because no inbound traffic to the backup system can be initiated by the target systems, the backup system has to initiate the action. Files designated for the purpose and interpreted by the backup system are then used for signaling.

- **Status reports on systems under backup:** These are produced by the backup system automatically and transmitted to target systems by providing 'restored' files containing the relevant status information. These are produced and transmitted periodically, depending on the desired information to be provided to each system. They are typically provided to the target systems and administrator systems.
- **Requests to the backup system:** These are placed in identified files on the target systems, backed up to the backup system, interpreted by mechanisms designed to only accept authorized syntax and perform known acceptable operations, and operated on by the backup system.
- **Polling only:** The backup system in this discussion has the fundamental principal of being immune from external influence or code injection. As such, having external systems able to interrupt or initiate any interaction with the backup system is prevented. This means polling target systems for any control input allowed by the backup system from target systems. A polling schedule is created in the backup system to check for requests and other controls from backup targets.

Care is required in the scheduling of polling to avoid slowdowns. For example, a process that checks each of 10 targets every minute will probably not be disruptive of target systems, each doing one secure shell operation per minute in response to the polling. However, opening a secure shell session pulling files from the backup system every 6 seconds may slow backup operations because of other activities on the backup system.

## *Automated recovery*

As an example of remote control, to restore files or directories from the backup system:

- The target system might place a file called "Restore" in a known location, with that file containing path names of the files and/or directories to be restored and the time span for restoration, seeking the most recent versions within an identified range of times.
- The backup system, upon retrieving the request within whatever time frames are selected for pulling such information, can look in the various repositories for relevant directories and files to be restored.
- Found versions can then be restored to a restoration directory in the target system and the user can then use them as desired.
- The backup system can also report the status of such requests as they are processed by updating a status reports so that, for example, each request is identified in a working list along with its last updated status, and upon completion, moved into a completion log, also updated on the target system if desired.

For full system recovery, we can take one of the backup disks (they are also USB disks mounted as user file systems, essentially identical to the user file systems on the Primary and Secondary machines), turn on a cold standby system, plug in a physical disk from the repository, and be up and running in a few minutes.

## *No centralization required*

This backup and restoration mechanism can operate at essentially any scale and cover any set of systems under backup. It can be used redundantly and in an overlapping configuration so there can be multiple backups covering any system under backup for any desired level of redundancy, physical distribution, and other desired properties. Backed up content and file names can be encrypted at the file or directory level if desired so that

  

service providers can provide backup services without access to the keys to reveal the content. These systems can run in completely isolated environments for local operation without dependency on the Internet or outside mechanisms. And they can be operated in any operating environment desired, so redundancy of hardware, software, transport, storage, and location can be applied to the extent desired. They can also be scaled to contain multiple computers at any given location, and to share the load for larger repositories.

## *Retention and disposition*

The mechanisms identified in “Build Backups Better” dealt with disposition at a technical level in the sense of deletion from the repository by overwriting the content and providing disposal records. But as a backup and recovery mechanism, some care must be taken:

- Recall (or be reminded) that the repository stores content by the cryptographic checksums and sizes of the content. As such, when you dispose of a file by its precise content, it is gone for all backups of it everywhere, once it propagates through the various storage devices.
- If you should foolishly delete some critical file or program that is used elsewhere, it will not only no longer be stored in the backup repository, it will not be placed into the repository again, but rather just marked as another instance of the disposed of content. It will still be recorded in the intermediate areas, like the weekly and monthly copies you just backed up, and the removed copies placed in a fire safe and so forth, but only till they are overwritten in subsequent backup processes.

An undo mechanism for this process is not available or anticipated. But a possible path to preservation would be to alter the file by one byte somewhere, in which case the new versions will be propagated into the backup repository going forward.

Note that if there are redundant backup mechanisms and repositories, disposition in one such repository may not be reflected across other repositories, and even if it is, the times associated with these operations may be different on different repositories. Similarly, if repositories take or give controls or content in different locations on their target systems, different actions may apply, producing inconsistencies between repositories. However, each should still accurately reflect the situations it has encountered. For example, a restore from a time frame might produce different results for different times backups were done in different repositories.

## *Key management*

Another substantial issue in most such systems is key management. The typical approach we use is to use secure shell (ssh) as a transport mechanism since it only depends on a single private and public key for each communicating repository. The public key is distributed to the targets who can implement it or not by giving the key appropriate access or not on a user by user or system by system basis. The private key is generated from and kept in the repository, and each repository can have its own. As such, no cryptographic infrastructure is required to support the secure connection, and repositories cannot act as if they were other repositories. Using sftp on the recipient side prevents the repository from executing commands on the target system, and limits read and write access to authorized locations. Thus the target can protect itself from a rogue repository even if the target chooses to embed the key in its own systems to allow access. Other systems, such as SSL can be used in infrastructure or non-infrastructure mode, each with their own properties depending on particulars.

## *Temporary outages and other failure modes*

Generally, backup systems operate using periodic processes, and if one period fails for whatever reason, the next attempt will invoke a retry. Of course each try can embed additional retries if desired. Because the mechanisms we use are effectively mirrors in time of remote systems, a partial copy may leave internal inconsistencies in the stored copy of the file system where different files are from different backup times. This is generally in the nature of all such systems, and even within a completely successful backup, the time it takes from when it starts to when it ends may end up with changes in the target system reflecting inconsistencies between the start and end time of the backup. You can of course shut down other uses of the target system between backups, but that reduces the value of the system as a whole.

  

### Backup system outages

The backup system itself can, of course, go down or have communication failures, leading to missed or partial backups. Because the present methodology is a mirror of the relevant parts of the target file system, this usually produces limited side effects, and because of the scheduling of updates and the repository mechanism underlying the definitive history system, restoration to an adequately consistent state is usually obtainable, even if it involves a substantial roll-back and restoration.

We tend to use ‘rsync’ through ‘ssh’ for target backups, and it tends to operate reliably including handling many failure modes relatively gracefully and providing log information usable to detect if things are going awry. The production of logs for the target systems provides this sort of information, and those systems and their users can look at these logs to detect problems and mitigate them. As an example:

- The log is not updated for more than the expected time frame indicating an outage, or the logs indicate failures of the backups along with details of the failure messages.
- A simple response would be to explicitly request another backup at a convenient time for the target system, and await the response.
- Eventually, if enough goes wrong, the repository operator can be contacted, or for owned repositories, a control system or administrative process can be attached to identify and fix the problem.

### Bugs in the transport mechanisms

One example found in a debug of our system at installation time was a different version of ‘rsync’ failing to properly back up an ‘sqlite’ file. Once found, ‘rsync’ was updated on the appropriate system, and operation proceeded properly. While copies of some files on the target system was not updated on the backup server in a timely fashion, once the problem was fixed, everything proceeded, and the changed files in the interim were then reflected in the backups and ultimately interred into the repository.

### Problems with mirroring targets in the backups

Some of the stranger failure modes include might be more problematic. For example:

- rsync deletes files that no longer exist in the same path name in the target system and makes new copies of the same files if they were moved elsewhere, under the new names.
  - This can be stopped by an option to not delete files in the backup system, but then when you move a large directory, the backup system will end up inconsistent with and full of extra copies of what was ever in the target system.
  - If you use the delete option, as the target file system is spanned, if the prior location of moved files are spanned first, the resulting deletions on the backup system will happen before the subsequent location copies are in the backup system.
  - If the subsequent location is spanned first, there will be duplicates in the backup file system until the deletion is encountered. If there are failures between these times, the backup will reflect an inconsistent state with the target system until the next backup.
  - If there is an error that somehow stops the backup from completing, the error might persist across backup versions leaving an inconsistent state for some time until the error mechanism is corrected.
- But perhaps more concerning is a target system that moves files around often as part of its automated processes.
  - It might be that some files are never properly accounted for in the backup system because the rate of change in the target system is too high for the rate of spanning the system in the backup process.

  

## Change tracking and large file changes

There is a solution to these sorts of issues, of course, that being a file system that tracks file system operations, with the backup mechanism replaying the file system changes instead of simply copying files and deleting others.

- Journaling file systems keep track of changes in a journal that doesn't get changes deleted until the results are reflected in the physical media. The same sort of mechanism is used in distributed databases for keeping transactions pending until accurately reflected in all or enough copies.[8]
- Virtual file systems allow for replay up to a threshold of history, and ReplayFS is an example file system that can be used for this today.[9]
- The Filesystem in Userspace (FUSE) file system[10] is a widely used approach that allows, among other things, remote mounting of file systems or portions of file systems, representing databases and other data structures as file systems, and is part of the approach for accessing archival content in the "Build Backups Better" approach. Because this file system is built on top of other file systems or any other mechanism that can emulate file system calls, it can be used to track each file system call and do anything desired with it, including creating a set of sequential changes that can be replayed elsewhere.

This approach has an added advantage in that, for large files containing databases and similar sorts of mechanisms that change in only a small portion of their content, it can automate a dramatic reduction in the quantity of content transmitted for backups. If you only use the file system as the basis for backups, a single byte change in a 500Gbyte file requires the entire file to be copied to the backup system to reflect the change. But if you have another mechanism for dealing with change on a more incremental basis, you may only have to transmit a few bytes of location information and one byte of content information to reflect the change in the backup copy.

## Backup scheduling

Another critical issue has to do with backup scheduling. Since a single server may be backing up multiple systems on a time schedule, if one backup takes a long time, it will increase the load on the backup server, slowing the next backup, and if the total set of resources for the scheduled backups get oo high, the overall backup system will grind to a halt. There are five obvious ways out of this:

- Schedule enough time between backups to allow for completion based on available resources on the backup system and networks between systems. This should be done anyway as part of the standard defense against denial of services. Unfortunately, these times are not as predictable as desired, and for significant volumes of changing content over time, this can get onerous.
- Create a backup queue and invoke the next element when there are enough resources. In some cases, backups may be during undesired periods of heavy load, causing potential other problems for other systems. And the queue may grow without bound if there are too little available resources to keep up.
- Do better error handling and stops and starts of backups to keep them within bounds of available time slots, so slow things get interrupted and restart during free time, if any.
- Add resources adequate to all the backups required in the worst case. This can be expensive and represents the tragedy of the commons with shared infrastructure.
- Prepare backups on the target systems, so that things like rsync are done on the target leaving only a zipped tar file for the backup and restoration processes. Most of the time in incremental backup is in searching for changes. Instead of scheduling the backup to do rsync remotely, do rsync on the target systems and back up the results when ready, perhaps with a polling system.

8 https://en.wikipedia.org/wiki/Journaling_file_system provides some details with examples
9 https://www.usenix.org/legacy/event/fast05/tech/full_papers/joukov/joukov.pdf N. Joukov, T. Wong, and E. Zadok "Accurate and Efficient Replaying of File System Traces", FAST '05: 4th USENIX Conference on File and Storage Technologies. Also available at https://www.filesystems.org/docs/replayfs/index.html
10 https://en.wikipedia.org/wiki/Filesystem_in_Userspace

  

### Maturity of the process

Fixing any specific problem is easy enough, but individual failures reflect failures in the processes used to operate the backup mechanisms. As such, the process should be updated to eliminate or otherwise mitigate future instances of each identified problem, and to retroactively correct the problem in cold and hot standby systems. Of course that is an issue of maturity of the overall entity and backup systems within the entity, and the risk management decisions associated with the level of perfection desired in the backup systems. We generally run at the Manged maturity level, and we advise everyone to reach that level in their backup and restoration systems.

## *What is being protected from what?*

The trust architecture of the system as described assumes a trusted archival mechanism, and that means that the target systems must trust the backup system more than the other way around. Balancing this trust has a lot to do with the entity and how it thinks about trust. As a fundamental, the backups system as a whole has to be trusted to do whatever its job is (the system includes all the redundancy you build into is, so that any failed system may not violate the trust architecture). Here are a few variations you might consider:

- Targets prepare whatever is to be backed up on their own, encrypting and compressing the content, and requesting backup and restoration as/when desired.
  - Problems include the repository not having access to the content and thus not meeting the archival requirements for things like data retention and disposition strategies and searches, etc.
  - The targets that don't act to schedule and properly operate their schedules commonly end up not being backed up for long periods of time, defeating the purpose of the backup system.
- Access controls on the target systems can limit the backup system as to what it can access for backups. This can be done 'in place' by trying to manage protection settings, for example putting things to be backed up in read-only mode for group and make them part of the group for backups.
  - As it turns out, this becomes a management nightmare. Getting all those settings right and compatible with all of the software that uses its own setting management processes causes lots of failures within target systems that are hard to detect and often harder to mitigate.
- Have special areas for copies of things to be backed up, making them accessible to the backup system without exposing other content.
  - This increases disk space, perhaps substantially, and creates a problem of synchronization between the actual content and the backup area copies.
  - In practice, users almost never consistently place the right things in the right places, and putting the burden on them is problematic.
- Have a push backup server for the users and a pull repository for the backup system so that the targets of backup push their backup content to the push server and the repository pulls from the push server.
  - This makes the whole system a little bit more complex and puts the impetus on the target systems to decide and act to back things up. This can fail because the targets don't operate properly, because they are, among other things, attacked.
  - The big advantages of this approach are that the primary systems can choose operational necessity over backup timeliness, and the load on the primary systems is under their control. We use this particular approach for several reasons:
    - The Primary systems are isolated from entry from the repository (or anywhere else), so other systems do not need keys to access them, and they need not even have remote entry capability, thus further hardening them from attack.
    - The Primary systems have some fragility under various conditions, and adding the remote control automated load has been observed to cause crashes.[11]

11 Overwatch: Careful what you watch for... you may see it. At https://all.net/Analyst/2025-10B.pdf provides

  

- The Primary system users are trusted in our trust model for doing backups, not automatically or instantly, but often enough, and using the authorized methods. In our case, the loss is their loss, so they avoid it wherever reasonable, and they have proven reliable at doing the backups.
- The logs on the various systems track process, and automation can detect inadequate behavior by the Primary system users. Non-compliance is addressed administratively.

All of this added complexity, while good for limiting what can go wrong from malicious actors, also causes lots of things to not go right for legitimate users. At some point, tradeoffs have to be made between operational efficiency, trust limitations, and risk. Uncertainty associated with complexity eventually exceeds that of failures.

## *Handling other repositories*

Mechanisms like Git[12] used in GitHub and other similar repositories can themselves be backed up and restored and have the integrity and other protective mechanisms of the system described here applied to them. This backup system can be used on any target system with compatible underlying tools (e.g., rsync, secure shell, and file system mechanisms). Backing up a Git repository, whether public or private, is like backing up anything else from the perspective of the backup system. Provide access via the network, place the proper public keys, configure the system to allow operation, and off it goes.

> Repositories of repositories can be implemented as well, of course, with a reduction in surety for the isolation repositories that are targets of other repositories.

As it turns out, if there is a lot to back up from a lot of different systems, the backup infrastructure will be larger and more complex than the systems it supports. And as it turns out, this is not all that unusual or a bad thing.

## *Backup here and restore there and other such changes*

The reason for backups is generally to be able to restore things. The systems where things were backed up are not always the same systems as the restoration targets. Depending on the addressing scheme, this can be relatively easy, but it also presents some risks (uncertainties).

- If you use DNS for name to number and use names for backup and restore, then all you have to do is change the name of they restoration target in the DNS, provide the proper public key and restoration directory, and the restoration should work. But if someone else changes the DNS entries, all of the content might come from or go to the wrong places.
- If you use IP addresses, like we do in our internal scheme, you can set the IP address of the target for restoration and place the proper keys and directory for restoration, and it will work. But be careful if you use DHCP or a similar protocol for providing IP addresses, the addresses may change and you will get the wrong content backed up and restored from the wrong targets.
- You can also have an administrator manually do a restoration (for example if you don't have a Secondary system in place or a cold standby with the proper keys), and bypass the other controls by directly typing on the console of the backup system(s). Of course the administrator will potentially be able to do bad things as well.
- And of course, you can setup the backup system to allow fo pre-defined restorations to Secondary systems rather than using them as hot standbys for the Primaries.

At small scale this works just fine, but at large scale, it becomes a management headache, which is why we think large scale should be attained by having many small systems distributed throughout infrastructure.

## *Use for deployment, distribution, provisioning, change management, etc.*

A backup system should really be just that. A backup system… intended to backup and restore other systems. However, folks have a tendency to leverage any tool that works well for other functions. Rather than let you imagine how this might work and go abuse the system by bypassing controls, I thought I would outline it here.

details of an incident of this sort cause by remote monitoring and backup systems and incompatibilities.

12 https://en.wikipedia.org/wiki/Git

  

The restoration capability of a backup system can be used to produce standard configurations and content on systems. For example:

- **Deployments and provisioning:** Prototypes can be reproduced across multiple machines through multiple restorations. This can be used for controlled distribution and other applications.
  - Changes tracked by mechanisms like rsync can be used to track and produce changes, restoration can then be used to reproduce an updated operating environment, and individual changes restored back into the updated standard.
  - Because the mechanism can act as a reproduction machine, it can form a viral computing environment wherein programs are reproduced by the backup and restoration mechanism.
  - Distribution of new software or content can be facilitated by the backup system doing backups of new content then restoring it to distributed machines, selectively or en mass.
  - Standard provisioning components can be distributed after system invitation or network boot, and these can be distributed by the restoration mechanism to provision new systems.
- **Change control:** The backup system can be used to intervene between research and development, testing, and deployments systems by acting as the control between them. For example,
  - The R&D group can place items for testing into an area backed up to the backup system and automatically restored to the testing system ingest location.
  - Testers can load from restoration with no ability to transmit back to R&D, and because the backup retains copies, the testing environment cannot alter the golden units sent to testing.
  - When approval is granted for production, the backup system can get the approval from the testing environment and deploy the updates it holds to the production environment.

  The backup system can also be used for reversion to back out changes to previous states, including the separation of data restoration from operating environment restoration by proper control of the repository.
- **Rebalancing the load:** The same mechanisms can be used to restore partitioned portions of content to different systems for processing to do things like non-instantaneous load balancing between machines.

These are all feasible with this system because of its automated nature and independence from control by systems under backup.

## *Keeping backup and restoration clean*

A backups system of this sort is also quite useful for countering the proliferation of malware through the backup system. It has long been known and published that backups act as repositories for the return of computer viruses and other malware.[13] Augmenting the present backup mechanism so that checking for known computer viruses and other malware is performed in the backup process just after loading into the backup system allows infected or otherwise undesired content to be marked as such in the backup system prior to long-term archival and restoration. Quarantine in the backups system for subsequent restoration for forensic purposes is feasible, as well as destruction before committing to the archives. But a more interesting facet of the process is that in the restoration process, updated checks for malicious content can be put in place so that even undetected malicious content entering and stored within the backup system can be blocked from subsequent restoration processes.

- **Backup:** Check for undesired content based on current methodologies.
- **Restore:** Check against undesired content using newly current methodologies.
- **Periodic:** Check the archives for then known methodologies and relate them to previous backups and restorations to track down undesired content over time and possibly remove it.

Of course in the backup system, you can also do other functions if you have appropriate mechanisms. For example, you can do code reviews using updated methods, regression testing, and similar functions, depending on the desire to make the backup mechanisms part of the more common practices of the entity.

13 https://all.net/books/virus/SCVirusBook.pdf F. Cohen, “A Short Course on Computer Viruses” pp19-20, 1990

  

### *Across multiple sites through infrastructure*

Via virtual private networks (VPNs), as an example, these backup and restoration mechanisms can operate maintaining integrity of the mechanism and content by using the encrypted channel as a tunnel through the VPN so that the VPN mechanism has no control over the content or ability to meaningfully alter it. The VPN can operate on the outside of the NAT protecting the backup system and as far as the backup system is concerned, the computers it is connecting to are just another computer on the (e.g., Green) network. The VPN mechanism that bypasses other controls just provides a virtual connection to the target systems.

### *Fragmenting content in the backup system*

In “Build Backups Better”, the use of cryptographic hashes combined with file-size allowed for file-based backup, restoration, disposition, and checking. But another approach to this is to have another layer of content associated, for example, with records within databases, emails within mail boxes, and other content below the file level within file-based content. It is an interesting notion that by fragmenting the content and identifying components of the composites that form the files and file systems, a great deal of space and time could be saved and far better provenance and attribution could be attained. Here are a few examples:

- **Libraries** embedded in compiled code could be segmented out. Since they are highly repetitious across many files, they could be replaced by a reference just as they were in the source code, and library usage and tracking would allow updated libraries to be tracked and adapted in a DBOM-like mechanism.[14] [15]
- **Code segments** are readily identified with exact copies of functions and programs within larger programs. A large portion of code in many of today’s systems are built up from existing code from other sources, and these may be readily identified.
- **Linguistic sequences** are quite common and compression based on sequences of words can produce both enormous compression and the ability to perform linguistic analysis. In addition to things like unacceptable word usage and violation of copyright, tracking of memes across communications sequences and segments of text copied and pasted or included in replies to emails and other messaging technologies support compression, provenance, and related maters. Language graphs could be used for this technology, and this is compatible with the current technologies underlying generative AI and search methodologies. This makes access and use of the backup system approach the utility of an archive while supporting compression as well as these other features.

Clearly, as we examine content in more depth there are benefits in terms of compression, comprehension, search, disposition, complying with legal requirements, and in many other areas. While this may not be the historical basis of backups, their increased usability for other purposes may well be worth exploring for archival solutions that do more than just back things up and allow for their restoration.

### *Enhanced surety for the mechanisms*

By configuring redundant protective mechanisms, the backup system can increase the surety of limiting access and enforcing outbound only session initiation. For example:

- The backup system can apply local firewall rules to limit the ability to communicate in unauthorized ways or from unauthorized sources even if the NAT mechanism it sits behind is somehow defeated.
- Multiple NAT mechanism of different manufacture and redundant channels can be made available against denial of services or other similar attacks.

14 https://www.openbom.com/blog/digital-bom-vs-traditional-bom-which-is-better-for-modern-manufacturers https://dbom-project.readthedocs.io/en/2.0.0-alpha-1/ https://dbom.io/ https://www.linuxfoundation.org/press/press-release/new-linux-foundation-effort-to-focus-on-data-confidence-fabrics-to-scale-digital-transformation-initiatives

15 The DBOM concept and initial implementations were initiated by Chris Blask in the 2015-2017 time frame and have spread to become part of the larger arena of assuring provenance in a wide range of systems including the narrative space, the software space, and as an evolution of the classic Bill of Materials used for centuries.

  

- Internal controls limiting the availability of target systems to the backup mechanism can be addressed by having multiple backup mechanisms operating in the independent subzones, or they can be operated through a control zone if there is one.
- Mechanisms can operate on virtual machines within the backup system machine for internal separation, or multiple servers can be used within a physical backup system for internal redundancy or performance.
- Multiple backup servers and their content can be backed up and checked against each other by a further NAT'ed backup server to detect differences between them, with results of the analysis passed back to those servers and/or other control machines to alert administrators of problems.
    - This can be operated by redundant independent operational groups so they check on each other, defeating insider threats for defense against N out of M corrupt systems or people.

Of course, the list is endless, but hopefully this gives you an idea of how these mechanisms can be leveraged for higher surety against specific sorts of attack mechanisms… at a cost

## *Structural efficiency and cost Issues*

It turns out that in almost all complex networked systems with lots of content on the edges and a smaller number of centralized (even if distributed) resources, it's critical to reduce the load starting at the endpoints. The earlier in the process and the closer to the outer edge you can reduce the resource requirements, the better it is for the system as a whole.

- By limiting what you back up to the things that are important to your entity and not backing up anything else, you save storage, communications bandwidth, repository load, intervening infrastructure load, and the need for more and more expensive resources everywhere along the way.
- By compressing content at the edges, you save everything in between, at the cost of decompressing in the servers. As it turns out, this is usually a net savings even ignoring the bandwidth issues, because storage bandwidth is often slower than decompression times. So it takes less space and goes faster.
- By only transporting changes to the backup server, the vast majority of resources are reduced in the vast majority of cases. The exceptions are things like large datasets with random internal changes that do not have transaction logs and checkpoints.

**Cost issues**

All of this obviously is adaptable constrained by cost. If and to the extent these mechanisms are automated and built into standard distribution by commercial entities, the cost of such systems can be made very low. We anticipate that open source versions may be made available soon…

The cost of the backup systems I use consists of 3 components:

- **Time spent:** The time spent in configuring the backup system and getting it running was considerable the first time I did it. But then I was doing the development and testing along the way, writing scripts, testing it manually, going through change control, and so forth. The time spent building a redundant backup server is less than an hour from scratch (with the software already developed). The time configuring the hardware and NAT gateway, etc. plugging it in, finding the shelf space, adding the network wiring, and so forth is the same as any other system in our infrastructure. The system is largely automated, so the user and administrative time is slightly less than it was before we put this system in place.
- **Hardware:** The systems I use are all systems that were no longer in use because they aged out of use as a Primary or Secondary system for lack of performance. The backup systems don't have substantial user interface activities, do not run the sorts of programs run by users, and 10-year-old systems are still adequate to the tasks involved. The added cost of handling multiple disks on these systems is less than $100 for a powered USB 8-port interface. The additional NAT gateways cost less than $50 each, and we currently only use 1 per backup server. And all the disks we now use were being used as Primary, Secondary, or backup disks previously. We just re-purpose them all instead of disposing of them.
- **Operations:** Operationally, the backup systems sit in a closet and are plugged into power strips and network cables. They consume power, but we have solar power adequate to the facility so the cost

  

differential is negligible. These older systems are a bit less efficient than newer ones, but they are laptops not using their displays except during configuration, and the disks generally only operate when they are being accessed. One backup system with 8 external disks takes about 2 ft of shelf space. The effect on other operations is negligible because once configured there is really nothing to do but watch for alerts or the lack of operational information. A reboot is rapid, simple to do, and not urgent enough to warrant any special care, and a rebuild consists of plugging in a cold standby and shifting the USB interface to the disks.

The cost of such backup systems might reasonably be compared to two different costs:

- **The cost of a less capable backup solution:** In my experience, it's just as hard and expensive to make a bad backup system as a good one. Perhaps more expensive if it is ad-hoc.
- **The cost of backup system failures:** This is the painful part. We have had failures in our backup systems from time to time over the last 50 years of my experience, and the lost development time and wasted time from what was lost and had to be remade is in the $1K+ per incident range.

## *But what should we (you) do in our (your) case?*

This report outlines my strategies and tactics for our environments and provides some alternatives for others to choose, but obviously, one size does not fit all. It's a reasonable question as to what you might reasonably and prudently do in your situation. To try and help out, I have outlined some of the issues and decisions, and more generally will refer you to our standards of practice for more details on our published decisions in this regard.[16]

Decisions start with the overarching understanding of your entity and its operations. The goal of a protection programs is to assure the utility of the mechanisms, and as such, you need to start with understanding the mechanisms and how technology supports their utility. In my case I started my description with the control architecture because I already know my businesses well. To help you I would have to start by understanding who you are and what success and failure look like for you. Control architecture decisions are one part of this.

A big part of understanding yourself is recognizing your size, skill, consequences, and maturity:

- **Size**: A micro-business, like a single small town shoe store, is obviously very different from a major retail chain with physical presence in 1500 locations across the world. In my case, I run a few micro-businesses, which has broad implications related to all manner of decisions, including especially trust-related decisions. If you are in a micro- or small- business, and your skill level is low, you should not try this on your own. If you are a larger entity and lack the skill, you should seriously consider paying for the expertise required to do this sort of thing.
- **Skill**: Skill level, in terms of cyber-security and other related matters to the issue at hand (backups), makes a lot of difference. If you don't have the skills to configure systems in these sorts of ways, you will have to do something else, like outsourcing some portion of it to others, buying a solution, or trying a different approach. My skill level in this arena is extreme, so I can and do regularly configure and customize systems of all sorts to meet my needs. Large entities tend to also have more skills available to them, but most small and micro businesses will not have the skills for this sort of thing. Most medium and large companies and enterprises will have the skills to do this and more.
- **Consequences**: The consequences drive everything. In my companies, confidentiality is almost a non-issue for backups, since almost all of the non-proprietary content is available from other sources as a commodity. The proprietary stuff, like patents, once published as patents, is public, and most of my writings are made available on our Web sites. There is essentially no regulatory requirement for confidentiality for almost anything we posses, and where there is, we do something special for confidentiality. Our identified consequences drive our risk management in our quest to limit future outcomes to those we desire, and this has to be assessed on a case by case basis. It's not just some number, it's about uncertainty about the future and what you are willing to tolerate as the owner(s) of the business or a representative of the owner(s). Our outcomes are most effected by the parameters identified earlier, and we make decisions on that basis.

16 https://all.net/SoP/SecDec/index.html This is not an up-to-date version. Updated versions are used internally.

  

- **Maturity**: We operate at Managed maturity. That is, we use repeatable processes that produce predictable outcomes, measure the processes and outcomes, and adapt the processes over time to meet changing situations and stay within our desired futures. If you operate at less than Managed maturity for backups and you have consequences of failure associated with content or capabilities of computer systems that is sufficient to do material harm to your company, in my generic view you are probably not operating in a reasonable and prudent manner. Of course there are exceptions to this, on a case by case basis. You have to be running at least in the Repeatable maturity level in order to successfully operate this scheme because it calls for repeated processes and if you do not get the result of the ability to restore what you have backed up on a consistent basis, then it's probably not worth doing the backups in the first place.

It turns out that in our analysis, these are more determinative than most other factors in decision-making for cyber-security programs…

**One more thing**. Anyone doing backups should **test them** in the sense of trying to do actual restorations in order to verify that they are working in the sense that when restored, the business can meaningfully use them for the utility they are intended to provide. It is a very good bet that if you don't test restoration to operation at least once per year, when you need the backups to restore properly, they will fail to do so.

## *A few examples of variations based on these parameters*

Typical entities have typical profiles, and different approaches are appropriate for them on this basis. Here are a few examples for some of the more common cases we encounter.

- **Regular family at home:** Most regular families living in a house have a few devices, perhaps some smart phones and a few computers, plus an Internet connection and some appliances. For you, our approach is overkill. You should make backups of your data (not the whole operating environment) on a USB drive or a cloud service provider or both. For sensitive information, you should have one computer only connected to the Internet when doing the sensitive things. A Chromebook is usually good for this.
- **Micro-business with little expertise**: Our approach is overkill. Use a NAT gateway to the Internet, segment your internal network to keep worker information (HR, finance, etc) limited to those authorized, and do backups to separate media kept elsewhere, but in your possession and control, and perhaps a copy in a bank's safety deposit box rotated monthly. You should be running at least at Repeatable maturity and have tested recovery from backups at least once a year.
- **Family offices and small investment firms:** These sorts of situations tend to call for a very strong solution manageable by few people and at relatively low cost. The reason strength is desired in this situation is that there tends to be a lot of value under control of a small number of people. They tend to run standard commercial systems that are easily penetrated and exploited, and they are quite often targeted by ransomware and more targeted threat actors. Strong backups provide the means for recovery and, in some cases, limitation of damages. They would sensibly use a deployment much like the one I use internally. It is low cost, simple to maintain and operate, and can be located in a small number of strategic locations (an office and a few homes) for a relatively high surety system.
- **Small and Medium businesses with medium or lower consequences**: Our approach is quite reasonable for most of these companies. Obviously there will be lots of tweaks, but the structure is basically right. If you have inadequate expertise, you should pay someone who knows how to do these things well to do them well, and you should get an expert to help architect it for your specific needs. It will save you a lot of problems later, for example, when you get a spyware attack or an insider decides to disrupt the company when they are fired. You should be running largely at Managed maturity, and this backup solution is easily operable at that level of maturity.
- **Entities with multiple locations and common content and/or operating environments:** These are ideal candidates for the distributed version of the backup and restoration processes. The multiple offices assure business continuity and disaster recovery in terms of content and distributed capabilities, those who travel site to site can use the backup and recovery mechanisms via a VPN, and those on the road can use VPN mechanisms for the same functions, local updates, and to address similar content, software, and related distribution challenges.

  

- **Companies with cloud services in limited use:** This is one of the ways we operate. Redundant cloud services providers with content reproduced and made available across their infrastructures. They are kept appropriately up to date in non-real-time. I would not trust the cloud service provider to have adequate backup and recovery, even if most of them do. And if you ever have to leave one and go to another, your backups will likely be required for business continuity.
- **Larger entities or higher consequence situations**: This is where it gets tricky. You should almost certainly have or bring in high expertise folks for the architecture and to independently verify backup and recover is working as desired. You should practice recovery in a limited way at least quarterly (although you likely practice much more frequently because of incidents requiring it), and have at least an annual thorough disaster recovery and business continuity exercise. You should run at Managed maturity for sure, and perhaps seek optimization beyond that. You should have internal expertise adequate to the task and bring in outside specialists to keep your internal team sharp and make sure your assessment is really as good as you think it is. At this level, you should also likely have a variety of specifically differentiated backup and recovery mechanisms for different aspects your entity, because there will be substantially different situations for different content and circumstances. A unified strategy with varying tactics is called for. For you, the methods identified here and in our previous articles on the subject should help identify some issues you may have missed, and perhaps be useful for review purposes.
- **Smaller software development shops and similar companies:** Most of these sorts of entities we encounter these days use Git repositories for managing version control, but typically fall under the spell of the central repository and get smashed when that tactic fails. For example, a recent infestation in the supply chain of Github corrupted thousands of repositories. A distributed backup and recovery mechanism provides a higher surety mechanism for assuring that substantial code loss is avoided and corruption, even of the Git or similar repository itself, does not destroy the content or its usability when reversion becomes necessary through backup systems.
- **Cloud storage issues:** Many cloud service providers have recently started to remove and destroy less recently used content. In many cases, this is done without notice. Similarly, lost account information or deaths in a family commonly result in loss of the cloud-based content, and this often includes pictures, personal journals, and other things shared within the family as part of its history. By creating one or more repositories that backup the cloud content from family members, families can relatively inexpensively assure the availability and integrity of family digital history.
- **Extreme consequences**: You should be far more custom than the overview I provided here. This will likely represent an ongoing process that examines and adapts at least quarterly, the number of backup locations may reach into the tens or hundreds, and so forth. For example a financial institution operating in 100+ countries: lots of regulations, real-time 24x7 processing, requirements to keep some content in country, extreme consequences for some failures, and so forth. Definitely lots of custom required here.

## *My current setup after experimenting for as bit*

Not that you should do what I do, but…

- I currently have a single backup system in **Red** with plenty of cold standby systems ready to go if I need them. It connects to about 8 external disks, has scheduled backups, pulls from a jump server rather than Primary or Secondary systems directly, and I have fire safes in 2 locations for holding monthly and aperiodic backup disks.
- When I leave the facility for an extended period (days or weeks) I generally take the most recent backup disks to the fire safes for increased surety of recovery of the most recent content in case of a fire or other incident that could be very harmful. I keep the backup system running 24x7 and find out about outages by my every 10 second watch programs, which I look at frequently because they are in front of me all the time when I am in the office.[17]
- I update the long-term repository every once in a while, at least once a year, but I have just gone through my disposition process on about 12 10+ year old systems, extracted the hard drives before sending them to recycling, and have not yet integrated all the content from these disks into the archives.

17 https://all.net/Analyst/2025-10B.pdf Overwatch: Careful what you watch for... you may see it.

  

- Not all users do their jobs all the time, so I try to urge them to do so or even do it for them myself every once in a while… for their own good… don't ask them about it.
- I am eternally tuning the mechanisms for performance, reliability, and so forth, and likely in the next year I will write another big article on backups and recovery.
- I am hoping to create a family archive as well, and two of my children started a business looking at doing this sort of thing for other families, so I figure I might get enough support with them to distribute that archives across homes around the world…
- I am doing the research required to create a fragmented version of the archives to both reduce size and enhance the capabilities of the archival content for other purposes. I also hope that this will support other research into using generative AI for responses to queries into segments of my published writing, code base, and elsewhere.
- I have not yet produced a full FUSE file system version of the long-term archives to allow ready reference to historical content by date and time or for date and time ranges, or to allow a multi-view of the content from perspectives pother than file systems from different systems at different archival times.

I view the backup and recover space and the archival space as increasingly part and parcel of the same thing.

- The ability to keep operations going dispute the inevitable failures of components with as little time and effort expended as reasonable is a top priority.
- The ability to make use of the historical content and reuse of content and code, reference when what was done easily, and rapidly apply the digital memory as part of my capacity to perform tasks efficiently is an emerging side effect of better backup, restoration, and archival capacity.
- The security of the archive, in the sense of the protection objectives and their prioritization is central to meeting these needs, but must be traded off with efficiency and effectiveness. I think I have this well balanced for the present state of hardware, software, trustworthiness, and my objectives.

So utility comes first, that utility being the low cost, low overhead, maximum usability without sacrificing reasonable and prudent protection meeting my objectives.

## *Conclusions*

Remember first that you paid nothing to me for this article and assume it is worth at least that much in terms of any advice you take from it. Each situation is different, and when we do this sort of work for clients, it is never as simple as our examples, which are not all that simple either as it turns out

This is our most recent backup scheme, updated from our last one that took more resources and was less systematic, leaving a fragmented set of backups. As it is being unified and consolidated, we have more cold standby machines, use less power, and are more certain we can find and restore in every covered situation. We are not covering some things of course, and that is a risk management decision that we have made.

For everyone out there who has encountered ransomware and didn't know what to do about it before, you now know what to do about it. Backups! Recovery! And systematic safe storage of historical versions are the critical element. To do this well, so that nobody can get in and screw it up, the simple precautions of; NAT gateways initiating only outbound from the backups system; not interpreting or otherwise using any of the backed up content other than for storing and retrieving it; and not updating the backup systems thereby preventing supply chain attacks from reaching your backup systems will get you about 99% of the way there.

Architecting this into a viable solution for individual environments is something I do for clients, mostly after they have already been slammed by a malicious situation or foolishly failed to do any reasonable backups for way too long. Sadly, the solutions to most of our cyber-security problems are right in front of us, has been published in the open literature, is based on things we have known for decades, and has been ignored by the industry, by the so-called experts, by the educational systems, and by the training and standards bodies.

It's time to stop ignoring what we already know in favor of some notion that new is better, and start using what we have to provide feasible solutions to the real-world problems we face every day. It isn't shiny and new, fancy, amazing, wonderful, or any of those sorts of things. It just works better. **Build Better Backups!**

  

### *Some nuances*

There are lots of little problems in creating an open source fully automated system of this sort that gets past all sorts of error modes, etc. But for those who want to try, here are some hints:

- cd $PLACE ; nice -n 19 rsync -azuSHv -e "ssh" $SYSTEM: .
  - This is used to do an rsync backup from a remote SYSTEM in PLACE
  - Reverse the ssh source and the directory (‘.’) to do a restore to the same place
- For disk to disk, use rsync locally on mounted disks with names associated to UUID of the disks.
- To approve new ssh signatures, have the requirement sent to an admin who replied by providing an approval or disapproval.
- To send status and related information, use a file in /tmp on the receiving machine with a name appropriate to the type of content and the user, perhaps in a *tmp*UID directory set up for the purpose by the user.
- Use crontab to configure periodic processes, and be careful not to schedule them to overlap themselves or others lest you create high loads on the backup system.
- Use a standardized filter for controlling inbound requests to the backup system. You will usually require only letters, numbers, and a few special characters (/, ., etc.) for pathnames.
- Don’t allow things like ‘..’ or initial ‘/’ or backquote, or parens, or other special characters to the shell unless you can handle them properly. Which is to say, don’t process user provided content using the shell or any other similar mechanism that over-interprets string inputs.
- Watch out for bandwidth and large volumes of content. Perhaps in your environment, these files are backed up differently. Things like large database files are changed on every database change, and it’s better to backup a state every once in a while and the transaction log for replay more often.
- The more you can structure what you back up and the sooner you can separate the wheat from the chaff, the better it will operate.
- These mechanisms retain what they back up. Don’t commit to online in backup areas what you don’t want made available later.
- In many cases, backups are already done to file shares on a network file serer. Consider backing the file shares only, or making the file share a lower surety repository to enforce backups from users and back up from there.
- Watch it run for a while, tracking disk usage, bandwidth, performance hits on target systems, etc. to get it effective without interfering with normal operations.
- Remember to rotate the backups of the backups to off-site or fire safe storage unless you have a distributed location backup system.
- Consider what to encrypt on disk in the backup system. If someone takes the disk they get a lot, but if it becomes inaccessible you may lose a lot. There are tradeoffs.

My incremental backup system, being what it is, means that I tend to click a few buttons to backup changes ridiculously often (one click on the right window, an up arrow, and enter). To get a sense of this, I have saved the changes to this document about a dozen times in the last hour or two, and I have done a full incremental backup of the Primary system I am editing it on at least 4 times, simply because I don’t want to lose the changes. It has become so easy to do that I do it often, and even though my system crashes far more often than it should (an average of more than once a month these days, largely due to lack of reliability of the electronic drives I use for high performance), as far as I can tell, I have not lost one word of what I was doing over the last year.

  