ownCloud Security and Encryption 2.0; A Technical Overview

ownCloud, Inc.
57 Bedford Street, Suite 102
Lexington, MA 02420
United States

phone: +1 (781) 778-7577
www.owncloud.com/contact

ownCloud GmbH
Schloßäckerstraße 26a
90443 Nürnberg
Germany

phone: +49 911 14888690
www.owncloud.com/de/kontakt

ownCloud Security and Encryption 2.0
A Technical Overview

As the use of file sharing increases across the industry, more attention is being paid to the inherent security of these solutions and the need for corporations to provide enterprise file sync and share (EFSS) solutions that meet IT's security parameters. The Enterprise Strategy Group (ESG) has been tracking online file sharing and Shadow IT and in a survey of IT professionals, found that sixty percent of the organizations surveyed either knew (32%) or suspected (28%) that unauthorized consumer file sharing services were being used by employees.

According to Terri McClure, senior analyst at ESG, even more concerning was the fact that when asked if employees were storing regulated data in unauthorized consumer accounts, 68% of those surveyed replied likely or very likely.

An enterprise-grade file sync and share solution that meets the strict IT security policies set in place will help protect the companies' most important resource – its data. This paper will review how to achieve enterprise-grade security with ownCloud Enterprise Edition.

Security Features

The ownCloud security features are designed to protect data in transit, while ownCloud's encryption protects data at rest. First we will look at some of ownCloud's more technical security features.

One of ownCloud's drivers for continued security improvement is to not only fix individual symptoms (e.g. the single bugs), but to also focus on identifying and resolving the root cause to prevent whole categories of vulnerabilities. ownCloud internal security processes and secure software development lifecycle aligns with industry standards such as ISOs 29147, 30111 and 27304.

- **Strict Content Security Policy**
  Content Security Policy (CSP) is one of the most useful and powerful web security features introduced in recent years. With CSP, applications can instruct the browser to follow a specified security model, including instructions to not execute any inline scripts or load remote resources. ownCloud employs a very strict policy boiling down to the following:
  - that if there is no policy to forbid the action,
  - execute only scripts served from the same domain,
  - execute only style sheets served from the same domain or embedded within the content,
  - load only images and fonts served from the same domain,
  - limit AJAX requests to the same domain and
  - additional less security sensitive but best-practice strict defaults for other components. For further details please take a look at the actual implementation.

Most security professionals agree that Cross-Site Scripting (XSS) is one of the most common web application vulnerabilities, and these have been exploited by cyber-criminals for more than a dozen years. Organizations such as Trustwave's Spiderlabs estimates that *"82% of web applications are vulnerable to XSS"*, and both WhiteHat Security and Trustwave reported that in 2014, XSS was the most prevalent vulnerability. For this reason, ownCloud has implemented a strict Content Security Policy. The implementation of this policy mitigates one of the most dangerous application issues, and ownCloud's default policy generally ensures that even if an attacker finds a potential XSS vulnerability, these are often not exploitable in a real-life scenario.

In fact, ownCloud was one of the first adopters of this technology, initiating its investment in Content-Security-Policy back in 2013. Over the past few years ownCloud has further hardened the default policy.

- **Data in Session is Stored Encrypted**
  PHP stores session related data within sessions. These are usually small files on the server containing data such as the login state or the username. We have hardened the PHP session storage in such a way that the ownCloud server can only read session data at the same time the user is using ownCloud.

  This is done by encrypting the stored session data with an encryption key stored in another cookie. If the user requests a page on ownCloud the encryption cookie will be sent by the sync clients or the web browser. Only with this cookie (which is not stored on the disk of the application server) can the session content be decrypted.

  This is especially relevant, for example, if a user uses external storage and selects *"Use login credentials"*. In this mode, ownCloud intercepts the password used at login and stores it in the PHP session to access other remote storages such as an internal SharePoint instance. However, the actual plaintext password will not be stored on the disk.

  While this is not a panacea, in order for someone to gain unauthorized access, that person would need to have administrator privileges and perform multiple malicious modifications to the ownCloud core server code. What is key is that these can be tracked and detected leveraging customer's existing intrusion detection systems. Furthermore, it helps prevent compliance violations such as accidentally storing the data on a backup tape.

- **Secure by Default Model**
  New ownCloud code uses the so called *"ownCloud App Framework"*, a modern MVC-like framework to develop code for ownCloud. Code relying on this framework uses a lot of secure defaults such as requiring CSRF (another specific kind of web vulnerability caused by the original design of the web) and authentication checks being opt-out rather than the more common (and less safe) opt-in. The default mode for every critical security feature in ownCloud is *"on"*, and requires the developer to deliberately *"opt-out"* of these security checks. These secure defaults are part of ownCloud's secure software development lifecycle. Secure defaults make it more difficult to accidentally trigger a security vulnerability Instead, it allows internal and external security professionals to easily assess the overall security of an ownCloud component.

- **Strict Comparison in PHP Technically Enforced**
  PHP has some peculiarities such as *"Type Juggling"*. This means that it will automatically try to convert data types when applicable such as in comparisons. An example would be the following comparison: "`0 == false`" where PHP will try to convert both values (integer and Boolean value) into a comparable state and thus, will return true.

  This can lead to unexpected behavior and potential security bugs if developers don't take this into consideration. ownCloud forces PHP to confirm that the data is exactly the same type by verifying the data type using strict comparisons as a best practice. These enforcements are applied using an automated code scanner as well ownCloud's strict coding guidelines.

- **Continual Code Hardening**
  In every release, ownCloud works to improve our API and introduce new hardening features to make the application more secure. Recent improvements include:

  - **"data/.htaccess" is updated after each update.** Since ownCloud 8.1, the existing .htaccess file in the data directory is updated after each release for enhanced security. Administrators are advised against performing any custom modifications to these files and, for an even more secure experience, encouraged to move the data folder outside of the web root.

  - **Trusted domains are a hard requirement.** A trusted domain is a domain that the ownCloud server accepts as Host header. So if you host *"demo.owncloud.org"*, the trusted domain will be *"demo.owncloud.org"* and users can't access it using another domain such as *"evil.com"* which could eventually lead to ownCloud generating URLs using the evil.com domain. To protect users, it is impossible to omit the *"trusted_domain"* settings.

  - **Request ID supports mod_unique_id.** Each request to an ownCloud instance is assigned an associated request ID which is used for logging purposes. ownCloud supports "`mod_unique_id`" which means that the request ID will not be generated by the ownCloud server, but by the web server instead. This allows administrators to better correlate log information within the ownCloud logs making it easier to track any potential security incident.

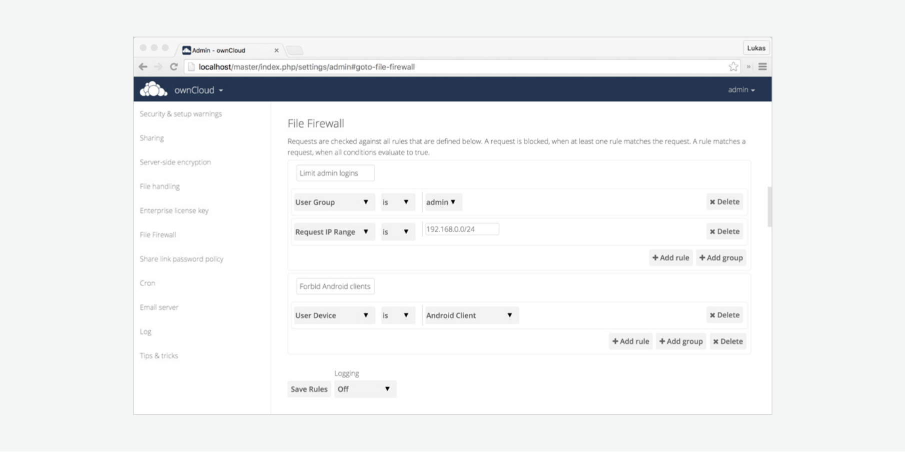
  - **Security-related headers can be sent by the web server.** When operating a web application it is often desirable to have some basic HTTP security headers (such as nosniffing instructions) enabled to prevent security pitfalls. For optimal security, administrators are encouraged to configure the web server to serve these HTTP headers. While this is not mandatory-- as ownCloud can also apply some basic headers--this is recommended for enhanced security.

  - **Enhancement of root certificate handling.** To avoid problems with proper HTTPS requests with PHP and misconfigured hosts missing proper certificate chains, ownCloud ships a root certificate bundle with ownCloud itself. This bundle contains the certificates shipped by Mozilla Firefox and is regularly sync'd with the upstream certificates.

  - **The internal file view class "OC\Files\View" prevents directory traversals.** ownCloud strives to have a *"security by default"* and *"defense-in-depth"* model in our code base. To support these, the ownCloud filesystem is built to prevent directory traversals by forbidding potentially dangerous character sequences such as "../" or "..\".

  These are just a few examples of some security optimizations ownCloud has implemented, and we are always working on adding further improvements.

- **File Firewall**
  Using the internal File Firewall of ownCloud's Enterprise Edition, enterprises can limit access to sensitive data even further. File Firewall is an application-level firewall that inspects all incoming ownCloud requests and evaluates them based on rules set by the administrator to only allow through *"approved"* requests for a finely granular level of control. Administrators can, for example, limit administrative logins to a pre-defined internal network to enhance security or allow access to shared folders only from a specific location to implement internal security guidelines.



localhost/master/index.php/settings/admin#goto-file-firewall

File Firewall

Requests are checked against all rules that are defined below. A request is blocked, when at least one rule matches the request. A rule matches a request, when all conditions evaluate to true.

Limit admin logins
User Group is admin
Request IP Range is 192.168.0.0/24

Forbid Android clients
User Device is Android Client

Logging: Off

Administrators can limit requests based on:

- Request IP Range
- Upload Size
- Subnet
- Request Type
- Request URL
- Request Time
- User Agent
- User Device
- User Group

Security Efforts

ownCloud employs a variety of best practices to continually review and improve the quality and security of the ownCloud code.

- **Internal Security Professionals**
  ownCloud employs security professionals to oversee the continued security of the ownCloud code base.

- **Internal Code Reviews**
  ownCloud performs internal code reviews as well as a consideration of a threat model for every new feature, functionality and code change to ensure that no new security vulnerabilities are created. A dedicated QA team tests each new release against the supported environments as outlined in the Minimum System Requirements.

- **Change Requirements**
  All code changes for the server component require two reviewers to review and approve changes before they are permitted into the code base. This is designed to make it difficult to introduce security problems either by accident or intention.

- **Automated Code Scans**
  In addition to having two separate developers review each piece of code, ownCloud also utilizes automated code scanners to review the code and identify every potential insecure code function.

- **External Penetration and Security Audits**
  External penetration and security audits are performed by external sources with each new release, providing another layer of security review for the ownCloud code base.

- **Utilization of CVE identifiers**
  The Common Vulnerabilities and Exposures (CVE) system provides a reference method for identifying and making known information security vulnerabilities and exposures. Following industry best practices, ownCloud has issued security advisories for each vulnerability identified, including very minor issues. ownCloud feels that these advisories contain information about the security of an application. ownCloud categorizes vulnerabilities as:

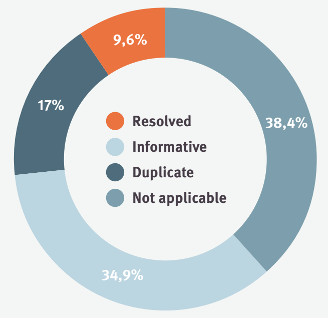
  - ***Critical*** – Vulnerabilities which may allow an adversary to gain complete control over the server or any files on it. This includes Remote Code Executions or SQL Injections.
  - ***Medium*** – Vulnerabilities allowing the adversary to gain complete control over a single user session. This includes Cross-Site-Scripting vulnerabilities.
  - ***Low*** – Vulnerabilities that can only be exploited in very rare cases or have marginal impact.

Striving for transparency and improved security, ownCloud's policy is to err on the side of releasing advisories, while focusing on continuing to fix the root causes. Below is a look at the number of vulnerability reports we have received since 2012. Note that a single advisory might have fixed multiple vulnerabilities. Also, a significant number of the vulnerabilities are located within the ownCloud Community apps and not the ownCloud Server itself. This means that Enterprise Edition users and anyone not using those apps are not affected by those vulnerabilities.

What does this mean? Security bugs have decreased over time and ownCloud is working to continue this trend.

- **Security Bug Bounty Program.**
  Consistent with other industry leaders, ownCloud implemented a Security Bug Bounty program. Partnering with HackerOne, ownCloud reached out to the Security Researcher community and personally invited the top 600 HackerOne users to provide an additional layer of review of ownCloud code. A reward has been offered to these skilled professionals for the validated security vulnerabilities they uncover. Below is a summary of the vulnerabilities identified to date via the Bug Bounty program. The ownCloud Bug Bounty Program is meanwhile running publicly and accessible to everyone at hackerone.com/owncloud

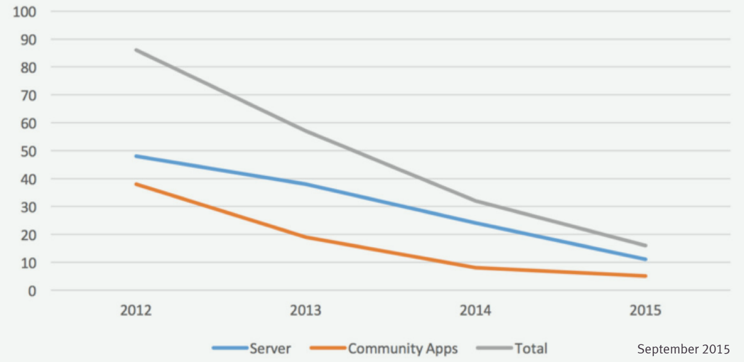
Types of Bugs Closed



September 2015

It should be noted that most of the resolved issues do not affect components related to the ownCloud Server or ownCloud Enterprise Edition and therefore don't affect the security of the ownCloud file sync and share

Fixed vulnerabilities per year



September 2015

solution. For complete transparency, a key value of ownCloud, Inc., all reports are published after a grace period on https://hackerone.com/owncloud

ownCloud Server-side Encryption

ownCloud has split the encryption app into two components to add additional modularity and flexibility into the overall encryption architecture. Customers are no longer bound by the out-of-the-box encryption module, and are able to implement precisely what they need for their environment, regulatory requirements, and business processes. With these enhancements, ownCloud has improved the server-side encryption to make it more customizable than ever before.

ownCloud gives customers the two things they want – the ability to manage their encryption keys in their own key stores, and the ability to customize the encryption behavior to meet their needs.

- **Server-side Encryption Threat Model.** The main usecase of the default encryption module is to protect data stored on remote storage or against a storage administrator checking the content of the files. A malicious ownCloud administrator however will be able to gain access to users' files as he can modify ownCloud in such a way to intercept the user's password. See also: https://owncloud.org/blog/how-owncloud-uses-encryption-to-protect-your-data/

  While many considerations are put in place to ensure the security of data, the default encryption module does not:

  - Hide the directory structure or folder names. A storage admin will be able to see the entire directory structure.

  - Encrypt files outside of the *"files"*, *"files_versions"* and *"files_trashbin"* folder. For example, this excludes previews or the index of the Lucene full text search app from encryption.

However, as Encryption 2.0 is highly flexible, you can integrate with existing security components such as a Hardware Security Module (HSM) to also protect key material from a malicious administrator. Also, you can completely adjust the way files are stored or change the encryption algorithms to comply with your internal security guidelines.

ownCloud's Server-side encryption application is designed to perform the following functions:

- The core component of ownCloud's server-side encryption allows administrators to ensure that files are stored encrypted at rest. ownCloud's encryption capabilities rely on *"encryption modules"*. These define the whole encryption logic. They can be written by implementing the "`\OCP\Encryption\IEncryptionModule`" interface.

  - This enables an ownCloud administrator to implement a custom encryption logic which is totally different from the default one shipped with ownCloud. This is useful when regulatory or internal requirements force the usage of specified or defined crypto components. It also can be valuable when trying to access existing encrypted data storages.

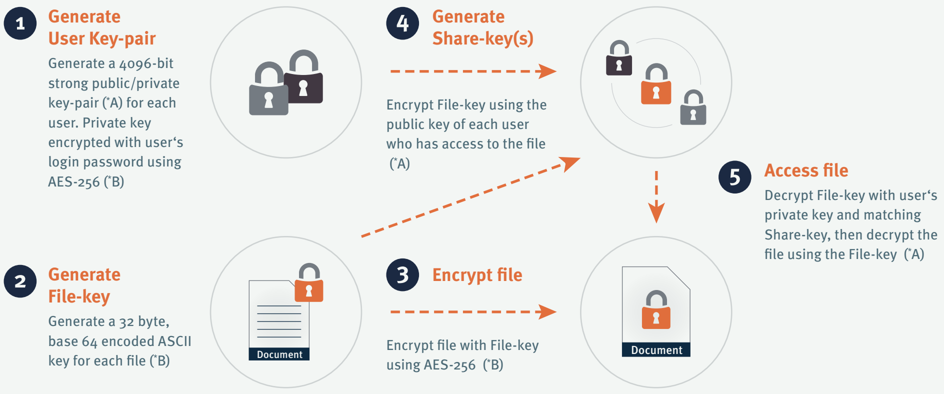
  - Enterprises implementing their own custom encryption module can decide what data they want to encrypt (or not) as well as implement custom key managements to suit their needs. Furthermore, they can easily re-use existing parts of the default encryption module and adjust the parts they want to change (such as storing the keys in a different place).

  - The default encryption module shipped with ownCloud is called *"Default encryption module"* (with an internal ID of "`OC_DEFAULT_MODULE`").

Below we will cover this module's functionality in more detail.

The following keys are generated by the default encryption module, and the usage of each of the keys is described below:

- Each user has a key-pair which consists of a private and a public key. The RSA key will be created the first time the user login using "`openssl_pkey_new`" with 4096 bits. They keys are stored in `data/$username/files_encryption/OC_DEFAULT_MODULE` as `$username.publicKey` and `$username.privateKey`



1 **Generate User Key-pair**
Generate a 4096-bit strong public/private key-pair (*A) for each user. Private key encrypted with user's login password using AES-256 (*B)

2 **Generate File-key**
Generate a 32 byte, base 64 encoded ASCII key for each file (*B)

3 **Encrypt file**
Encrypt file with File-key using AES-256 (*B)

4 **Generate Share-key(s)**
Encrypt File-key using the public key of each user who has access to the file (*A)

5 **Access file**
Decrypt File-key with user's private key and matching Share-key, then decrypt the file using the File-key (*A)

(*A) Encryption 2.0 enables customer to leverage an enterprise key store or separate path for key storage
(*B) Encryption 2.0 allows customers to implement a different algorithm and encryption flow

  - The encryption module creates two system key-pairs, one for public link shares and one for the recovery features

  - Each file will have a 256 bit cryptographically secure random file key stored at `data/$username/files_encryption/keys/files/filename/OC_DEFAULT_MODULE/fileKey`

  - Each file will have a automatically created share key for each user with access to the file, share keys are stored at `data/$username/files_encryption/keys/files/$filename/OC_DEFAULT_MODULE/username.shareKey`

- In order to secure the user's private key, the key is encrypted and stored on the disk using the default encryption method explained below, as the password to encrypt the key is a derived form of the login password used. The login password is run through a PBKDF2 key derivation with 100,000 iterations, an instance-specific salt and 32 (AES-256) or 16 (AES-128) byte key size depending on the configured key size.

- After login, the private key is decrypted and stored within the user's PHP session. As explained above, ownCloud encrypts all session content with AES-128 using an additional cookie sent by the user for each request so the private key is not stored in plaintext on the disk.

- Users sometimes forget their passwords. ownCloud allows administrators to optionally enable a recovery key feature that can be used to restore data access in the case of a lost password. The recovery key feature is enabled centrally, after which each user can choose whether to enable it for their ownCloud account.

- To enable public link sharing the encryption module uses a special key-pair for which the private key is encrypted with an empty password in order to allow anonymous access to files shared publicly.

- When a new file is added or sync'd, ownCloud generates an associated file-key and uses it to encrypt the file, ensuring that every file known to ownCloud has a unique file-key.

  - The default encryption module creates a 256 bit cryptographically secure random file key for each single file. This file key is then encrypted to each public key for all users with access to the file, using openssl_seal in RC4 mode and a 128 bit long random secret key. The encryption module knows this secret key as *"share key"*.

- An HMAC is calculated for each chunk of the encrypted file, the HMAC key is generated by hashing together the private file encryption key, the file version as well as the location of the chunk appended by an "a". When reading files the integrity is verified, if the file has been tampered with a warning will be logged and shown to the end-user.

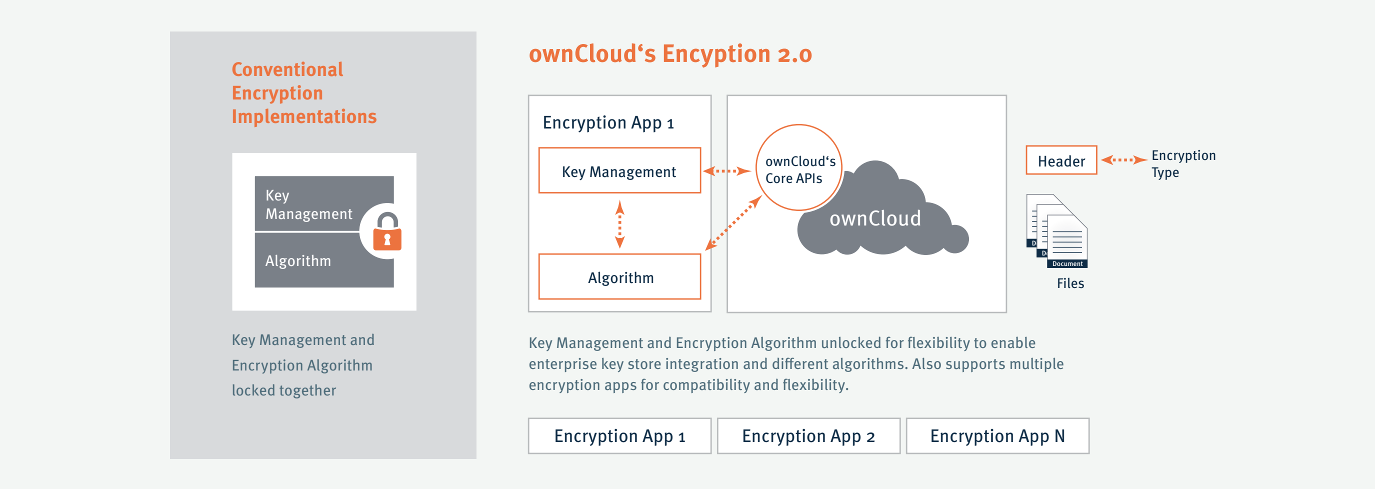
- When an authorized user asks to access a file, the encryption module decrypts the file-key with a combination of the user's private key and the appropriate share-key, which then uses the file-key to decrypt the physical data file.

  - The specific files are then en- and decrypted via openssl_decrypt or openssl_encrypt using the specified cipher in config.php (either AES-128-CFB or AES-256-CFB (default)), and a 96 bit random IV (generated using openssl_random_pseudo_bytes).

- When a file is subsequently shared with a new user, the file-key is again encrypted with the new user's public key to create a new share-key. Although this abstraction requires ownCloud to re-encrypt the file-key, it eliminates the much more expensive task of re-encrypting the entire physical file when the file is shared with new users. The same benefit is achieved when revoking a user's access to one or more files.

- ownCloud's default encryption module stores the following data encrypted on the disk:

  - Files regularly created via the web interface or WebDAV
  - Versions of files
  - Files stored in the Trashbin
  - Private encryption keys



Conventional Encryption Implementations

Key Management and Encryption Algorithm locked together

ownCloud's Encyption 2.0

Key Management and Encryption Algorithm unlocked for flexibility to enable enterprise key store integration and different algorithms. Also supports multiple encryption apps for compatibility and flexibility.

Advantages of ownCloud's Encryption Model

- It is highly secure – it has been implemented using proven, broadly adopted technologies like OpenSSL and standards such as AES-256 that are endorsed by organizations such as NIST.
- It is optimized to perform well even when an organization has many users and very large files.
- Files can be stored securely on any ownCloud-accessible storage, in any supported format, and they can be stored externally without exposing file content to 3rd parties.
- Unlike cloud-only FSS vendors, ownCloud administrators maintain complete control over their encryption keys.

Encryption is customizable to match the internal requirements of an organization such as custom key managers or using another encryption approach.

Summary

ownCloud's data encryption model combines proven server-side encryption for data at rest with an architecture that can be easily extended to support other advanced security requirements. Based on a proven, broadly adopted foundation, ownCloud offers data protection across a variety of storage formats without putting data at risk. Importantly, ownCloud's encryption model is highly scalable and allows administrators to maintain complete control over their encryption keys.

The combination of ownCloud's security features, security efforts including the ownCloud Security Bug Bounty program, and server-side encryption provide an enterprise-grade file sync and share solution that is protected, fast, scalable and flexible. ownCloud offers peace of mind to organizations that need to securely meet a broad range of file sharing objectives.

For more information also check out the *"Optimizing ownCloud Security"* whitepaper at https://owncloud.com/whitepapers

ownCloud, Inc.
57 Bedford Street
Suite 102
Lexington, MA 02420
United States

www.owncloud.com/contact
phone: +1 (781) 778-7577

ownCloud GmbH
Schloßäckerstr. 26a
90443 Nürnberg
Germany

Visitor's Address
ownCloud GmbH
Leipziger Platz 21
90491 Nürnberg
Deutschland

www.owncloud.com/de/kontakt
phone: +49 911 14888690

www.owncloud.com

@ownCloud
facebook.com/owncloud
gplus.is/owncloud
linkedin.com/company/owncloud

Glossary

AES

Is a military grade encryption block cipher with a block size of 128 bits and with a key length of either 128 or 256 bits.

AJAX

short for Asynchronous JavaScript and XML, is a set of web development techniques utilizing many web technologies used on the client-side to create asynchronous Web applications.

CFB

Ciphertext feedback (CFB) is a mode of operation for a block cipher (in this case AES). In contrast to the cipher block chaining (CBC) mode, which encrypts a set number of bits of plaintext at a time, it is at times desirable to encrypt and transfer some plaintext values instantly one at a time, for which ciphertext feedback is a method.

Cipher

an algorithm for performing encryption or decryption—a series of well-defined steps that can be followed as a procedure in cryptography.

Client–side encryption

In client-side encryption, the end-user is responsible for maintaining their keys, accessed with a password, which keeps control over their data in their hands. This can limit the risk of outside access to their information. An advantage of this approach over the server-side option is that an administrator or service provider cannot be compelled, against the users' wishes, to produce the keys and the data for law enforcement requests.

Content Security Policy (CSP)

is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross-Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware.

Cookies

a small piece of data sent from a website and stored in the user's web browser while the user is browsing it. Every time the user loads the website, the browser sends the cookie back to the server. Cookies are usually used to authenticate the user against web services.

Cross-Site Scripting (XSS)

Cross-site Request Forgery, also known as a one-click attack or session riding and abbreviated as CSRF or XSRF, is a type of malicious exploit of a website whereby unauthorized commands are transmitted from a user that the website trusts. Unlike cross-site scripting (XSS), which exploits the trust a user has for a particular site, CSRF exploits the trust that a site has in a user's browser.

CVE

Common Vulnerabilities and Exposures system provides a reference-method for publicly known information-security vulnerabilities and exposures. CVE is maintained by MITRE and many standard enterprise security monitoring products do keep track of those and inform the administrator in case an unpatched application has been found.

EFSS

Enterprise File Sync and Share - a service that allows users to save files in cloud or on-premises storage and then access them on other desktop and mobile computing devices.

Encode

The purpose of encoding is to transform data so that it can be properly (and safely) consumed by a different type of system, e.g. binary data being sent over email, or viewing special characters on a web page. The goal is **not** to keep information secret, but rather to ensure that it's able to be properly consumed.

Encryption

the process of transforming messages or information in such a way that only authorized parties can read it

Encryption Key

Key that ownCloud uses to de- and encrypt files.

HMAC

a **keyed-hash message authentication code (HMAC)** is a specific construction for calculating a message authentication code (MAC) involving a cryptographic hash function in combination with a secret cryptographic key. HMAC is used by ownCloud to guarantee the integrity of the encrypted payload.

HTTP Security Headers HTTP headers are additional meta information transferred between browser and server. Some headers do serve security purposes and are enabled automatically by ownCloud.

MVC The Model-View-Controller architectural pattern separates an application into three main components: the model, the view, and the controller.

NIST National Institute of Stands and Technology, the federal technology agency that works with industry to develop and apply technology, measurements, and standards.

PHP The programming language that ownCloud server is written in.

Private encryption key a private or secret key is an encryption/decryption key known only to the party or parties that exchange secret messages. In traditional secret key cryptography, a key would be shared by the communicators so that each could encrypt and decrypt messages. ownCloud uses public-key cryptography so that that keys are not shared between users.

Public encryption key a value provided by a designated authority as an encryption key that, combined with a private key, can be used to effectively encrypt messages and digital signatures.

Recovery key A private encryption key to which all files are encrypted as well, the administrator has the ability to recover data using this key.

Server-side encryption the cloud storage provider manages the encryption keys along with your data. Many of the most well-known cloud storage providers use this configuration. Server-side encryption limits the complexity of the environment, while still maintaining the isolation of your data. With ownCloud being an on-premise solution the keys never leave your own server.

SharePoint a web application platform in the Microsoft Office server suite. SharePoint combines various functions which are traditionally separate applications: intranet, extranet, content management, document management, personal cloud, enterprise social networking, enterprise search, business intelligence, workflow management, web content management, and an enterprise application store.

Type Juggling the name given by the PHP developers to the "feature" in PHP that allows a programmer to compare values of different data types without explicitly converting them. (such as comparing the string “1” to the integer 1)

WebDAV **Web Distributed Authoring and Versioning** is an extension of the Hypertext Transfer Protocol (HTTP) that allows clients to perform remote Web content authoring operations. The WebDAV protocol provides a framework for users to create, change and move documents on a server, typically a web server or web share. The most important features of the WebDAV protocol include the maintenance of properties about an author or modification date, namespace management, collections, and overwrite protection.

XSS **Cross-Site Scripting** – a type of computer security vulnerability found in many web applications. XSS enables attackers to inject client-side script into web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy. Using a XSS attack an attacker may gain the same privileges as the logged-in user.