Miscellaneous

Additional and miscellaneous information, often requested by users and customers, but that wouldn't fit in any specific knowledge base category.

- No, we are not affected by Log4j vulnerability (CVE-2021-44228)
- How Syncplify.me Server! prevents SSHPsycho attacks
- W3C log file format and UTC timestamps
- Firewalls and FTP external IP address for PASV
- Where do I download the old v4/v5 installers?
- Why PGP is an extremely bad choice for a file server's at-rest encryption, and how to do it right
- Timeout logging into the SuperAdmin or Admin UI? This is how you may fix it.
- Understanding FTP and Its Variants
- SFTP and SCP: Secure File Transfer Protocols

No, we are not affected by Log4j vulnerability (CVE-2021-44228)

No Syncplify software uses (nor has ever used) anything written in Java. Furthermore, and more specifically, no Syncplify software uses (nor has ever used) Log4j. Therefore, none of our software is affected by (nor it has ever been) any Log4j bug or vulnerability.

How Syncplify.me Server! prevents SSHPsycho attacks

According to the SANS ISC nearly 80% of all SSH-based brute force attacks are caused by SSHPsycho or one of its variants. This seems to be confirmed by the LongTail honeypot real-time report provided by the Marist College. So, yes, SSHPsycho is a big deal, and it's a problem. And traditional blacklisting mechanisms (simply banning certain "well known" IP addresses and networks) have proved to be inefficient against it.

> "LongTail shows that Cisco and Level 3's recent announcement about blocking sshPsycho's 4 class C IP ranges (also known as "Group 93" and the "Hee Thai Campaign") has done nothing to stop their brutal attacks. [Source: SANS ISC]"

Syncplify.me Server!'s intelligent and automatic blacklist (called "**Protector**"), though, shows to be **extremely effective at preventing** such type of attack. Its real-time dynamic attack pattern identification and prevention technology can quickly recognize SSHPsycho attacks (and the like) and **proactively stop them** as soon as they begin. Even at its "Normal" sensitivity threshold, Protector already identifies and blocks all types of SSHPsycho attacks, in most cases **before they even get to try the password authentication**.

Of course, not even Protector can keep you safe if you have a user whose username is "*test*" and its password is "*123456*", so it's strongly recommended to read the LongTail report and **avoid using the most common username/password combinations** that would make your SSH/SFTP server inherently vulnerable, not only to SSHPsycho but pretty much to any known and unknown attack. But again, Syncplify.me Server! helps you by **enforcing password complexity rules** that prevent users from using passwords that would be too easy to guess.

W3C log file format and UTC timestamps

Every once in a while we receive a support request from some customers asking us how to **“fix” the timestamp** in the log files because it’s few hours ahead/behind.

The thing is that such timestamp is not ahead nor behind: it’s always in UTC (Coordinated Universal Time), and that is not our arbitrary choice; in fact, the W3C Extended Log File Format official working document clearly states that the timestamp **must** refer to the GMT time zone without daylight savings bias, which is – indeed – called UTC for brevity.

For such reason, all existing log analysis software products are designed to take that into account and adapt the generated reports to the current time zone of the machine that runs the analysis.

There are tens of log analysis software titles out there, but since sometimes our users ask us for a recommendation, here’s two among our favorites:

- Sawmill
- SmarterStats

Both of them are very high-quality products, and have been tested and confirmed to work flawlessly with the log files products by Syncplify.me Server!

Firewalls and FTP external IP address for PASV

Most firewalls (we'd say all the ones we know) have NAT/PAT capabilities, and many are able to perform a protocol-level inspection when the connection is not encrypted. SSH (and SFTP) are always encrypted, but FTP can be either encrypted or not; yet, theoretically, **protocol inspection should only prevent protocol-related attacks, not modify client requests or server responses.**

Yet, one customer with a perfectly configured instance of Syncplify Server! reported experiencing a **weird behavior**: FTPS/FTPES (encrypted) sessions were working perfectly, while plain FTP sessions were dropped upon every attempt to open a data connection to transfer files.

Now, theoretically, this doesn't make any sense, as the core engine for the FTP protocol is the same, and the TLS channel is layered on top of (or, more precisely, encapsulates) it. The only possible explanation was that something between the client and the server **was actively modifying the protocol**, and could obviously do so only when the FTP connection was performed in clear, with no TLS encryption.

In this particular case, a SonicWall firewall was **rewriting Syncplify Server!'s response to the PASV command** in order to modify the IP address and Port for the next incoming (requested) passive data connection.

Now, Syncplify Server! – as most FTP/S servers – allows the server's administrator to configure an "*external IP for PASV connections*" to be used specifically when running our server behind NAT/PAT. This allows the server to tell the client to which public IP:Port it should connect to initiate the data transfer connection. If the firewall rewrites such information and re-routes the data connection towards a different IP or Port, then Syncplify Server! would **not accept the new incoming data connection request, because it is – de facto – tampered or spoofed**.

If you have a firewall that acts like this (more like a reverse proxy, rather than just a simple firewall) removing the "*external IP for PASV connections*" from Syncplify Server!'s configuration **may fix the issue**, as it did in our example case. But please keep in mind that such a configuration setting is there for a reason, and if your firewall is behaving transparently it is necessary to use it when behind NAT/PAT.

Where do I download the old v4/v5 installers?

> **WARNING**: both *version 4* and *version 5* are now discontinued and retired, so please be aware that downloading the following installers comes with no guarantee of support or fitness-for-purpose of any kind. **Syncplify, Inc. is also not liable for any malfunction or damages these versions of the software may cause, for any reason, to the maximum extent permitted by law. If you continue, you take full responsibility, and do so at your own risk.**

Download: version 4.2.5 installer

Download: version 5.1.30 installer

Why PGP is an extremely bad choice for a file server's at-rest encryption, and how to do it right

Pretty Good Privacy (PGP and all of its variants) is a well-known encryption program that provides cryptographic privacy and authentication for data communication. While PGP is excellent for securing emails and their attachments, using it for at-rest file encryption on a file server is not advisable. This article explains why PGP is unsuitable for this purpose and why a streaming encryption method is a better alternative.

Limitations of PGP for At-Rest File Encryption

Performance Overhead

PGP is designed for encrypting and decrypting individual files (like an email attachment, for example) or messages. When used for at-rest encryption on a file server, it introduces significant performance overhead. Each file must be encrypted and decrypted in its entirety, which can be time-consuming and resource-intensive, especially for large files or a high volume of files.

Scalability Issues

File servers often handle large amounts of data and numerous files. PGP's approach of encrypting files individually does not scale well in such environments. The process of encrypting and decrypting each file separately can lead to bottlenecks, reducing the overall efficiency of the file server.

Complex Key Management

PGP relies on a system of public and private keys for encryption and decryption. Managing these keys for a large number of files on a file server can become complex and cumbersome. Ensuring that the correct keys are used and securely stored adds an additional layer of complexity.

Lack of Real-Time Encryption

PGP does not support real-time encryption and decryption. Files must be fully written to disk before they can be encrypted and fully read before they can be decrypted. This lack of real-time processing can lead to security vulnerabilities, as files are temporarily stored in an unencrypted state.

Not Post-Quantum Secure

PGP relies on cryptographic algorithms like RSA and DSA, which are not considered secure against potential quantum computing attacks. Quantum computers could potentially break these algorithms, rendering the encrypted data vulnerable. In contrast, other encryption methods, such as AES-256 GCM, are considered resistant to quantum crypto-analytic attacks.

The Need for Streaming Encryption

Real-Time Encryption and Decryption

Streaming encryption methods allow for real-time encryption and decryption of data as it is written to or read from storage. This ensures that data is never stored in an unencrypted state, significantly enhancing security.

Improved Performance

Streaming encryption is designed to handle data in a continuous flow, which reduces the performance overhead associated with encrypting and decrypting entire files. This approach is more efficient and better suited for environments with high data throughput.

Scalability

Streaming encryption methods are inherently more scalable than PGP. They can handle large volumes of data and numerous files without the performance bottlenecks associated with PGP. This makes them ideal for use on file servers that need to manage extensive data storage.

Simplified Key Management

Many streaming encryption solutions use symmetric key encryption, which simplifies key management. A single key can be used to encrypt and decrypt data streams, reducing the complexity of managing multiple keys for individual files.

Conclusion

While PGP is a robust tool for securing individual files and communications, it is not suitable for at-rest file encryption on a file server. The performance overhead, scalability issues, complex key management, and lack of real-time encryption make it an impractical choice. Instead, streaming encryption methods provide real-time encryption and decryption, improved performance, scalability, and simplified key management, making them the ideal solution for securing data on a file server.

How Syncplify Server! does it □□

At-rest encryption for a file server, and more specifically for an SFTP/FTP(E/S) server, is typically an Enterprise need; for such reason the Ultimate edition of Syncplify Server! has built-in top-notch at-rest streaming encryption at VFS level. This uses the **AES-256** algorithm (which is found to be post-quantum secure, unlike PGP) in **GCM** mode, which has also the added benefit of being **"*authenticated encryption*"** meaning it gives you the absolute certainty your data streams cannot be tampered with during the reading or writing operations.

Timeout logging into the SuperAdmin or Admin UI? This is how you may fix it.

Are you experiencing a timeout while attempting to log into the SuperAdmin or Admin UI? If your software ran for too long without a valid license, or was repeatedly started stopped hundreds or even thousands of times without ever clearing the notifications, that might be the culprit. Notifications stack up, and then the *clean-then-load* process that occurs at login may take too long to complete, causing a timeout.

But fear not! As of v6.2.49 there is an extremely easy solution to that.

Simply open a terminal or command prompt (do it **"as Administrator"** if you are on Windows), then change directory to the folder where Syncplify Server! is installed and type the following command:

On Windows (as Administrator)

```
.\ss6-webrest.exe dbcmd --clearallnotifications
```

On Linux

```
sudo ./ss6-webrest dbcmd --clearallnotifications
```

After that, if the excessive number of pending un-acked notifications was the culprit, you should be able to log in without further issues.

Understanding FTP and Its Variants

FTP (File Transfer Protocol) is a standard network protocol used for transferring files between a client and server on a computer network. As a junior system administrator, it's crucial to understand the different variants of FTP and their implications for security and network configuration.

Plain FTP

Plain FTP is the original, unencrypted version of the protocol. It typically operates on port 21 for control connections and port 20 for data transfers. However, it's important to note that plain FTP is inherently insecure, as it transmits data and login credentials in clear text.

Implicit FTPS

Implicit FTPS is an early attempt to secure FTP connections using SSL/TLS. It uses port 990 for control connections and assumes that the connection should be encrypted from the start. While more secure than plain FTP, it's considered deprecated and may not be supported by all modern FTP clients.

Explicit FTPS (aka FTPES)

Explicit FTPS (also known as FTPES) is the current standard for secure FTP transfers. It uses the same port as plain FTP (21) but allows the client to request encryption during the authorization phase (using the AUTH command). This method is more flexible and widely supported by all modern FTP clients.

Active vs. Passive FTP

Understanding the difference between active and passive FTP is crucial for proper network configuration.

Active FTP (PORT)

In active mode, the client opens a port and waits for the server to connect back to it. This can cause issues with firewalls and NAT (Network Address Translation) devices.

Passive FTP (PASV)

Passive mode allows the client to initiate both connections to the server, which is generally more firewall-friendly.

> **Pro Tip:** Always use passive FTP unless both client and server are on the same physical LAN. Active FTP often fails when routing is involved due to firewall and NAT complications.

Security Considerations

When setting up an FTP server, consider the following:

1. Use FTPES whenever possible for enhanced security.
2. Configure your firewall to allow passive FTP connections by forwarding all ports in your server's "*passive port range*".
3. Regularly update your FTP server software.

Remember, while FTP is a powerful tool for file transfer, some of its limitations may make it unsuitable for transmitting data securely and efficiently. Unless a protocol of the FTP family is your only options, SFTP - which is a subsystem of SSH-2, and not at all related to FTP(E/S) - is almost always your best option.

SFTP and SCP: Secure File Transfer Protocols

SFTP (SSH File Transfer Protocol) and SCP (Secure Copy Protocol) are both secure file transfer protocols that operate as subsystems of SSH-2 (Secure Shell version 2). These protocols provide encrypted and authenticated methods for transferring files between systems.

SFTP (SSH File Transfer Protocol)

SFTP is a robust and versatile protocol that offers secure file transfer capabilities:

- It operates over an encrypted SSH connection, typically on port 22.
- SFTP provides strong encryption and authentication to protect against various attacks.
- It supports a wide range of file operations, including uploading, downloading, and remote file manipulation.

Key Features:

- Encryption of data during transit.
- Simplified use with a single port connection.
- Support for advanced features that most other file-transfer protocols don't have.

SCP (Secure Copy Protocol)

SCP is another secure file transfer protocol that, like SFTP, operates as an SSH-2 subsystem:

- It is designed specifically for securely copying files between hosts on a network.
- SCP uses the same authentication and security mechanisms as SSH.

Comparison to SFTP:

- SCP is simple to use in scripts, but lacks many of SFTP's advanced features.
- SFTP offers more functionality, including the ability to resume interrupted transfers and perform remote file system operations.

> **Poison pill:** in order to fill the functionality gap between SFTP and SCP, some SCP clients attempt to use a parallel SSH-2 Shell to perform some of the operations the SCP protocol lacks, but in doing so they create a potential side-channel for attacks. If all you need are secure file-transfers, and SCP is requesting to perform Shell operations, the safest choice is

> to drop SCP altogether and switch to SFTP. Do not allow Shell just because SCP may ask for it, unless you know exactly how to keep it safe.

SSH-2 Subsystems

It's important to note that SFTP and SCP are not the only subsystems of SSH-2. The SSH-2 protocol is a versatile framework that supports various subsystems:

1. **File Transfer:** SFTP and SCP.
2. **Remote Command Execution:** Allows running individual commands on the remote system.
3. **Shell Access:** Provides a full interactive shell session on the remote machine.

This flexibility makes SSH-2 a powerful tool for secure remote system management and file transfer, but - as all powerful tools - it requires careful configuration from a knowledgeable administrator.

Security Considerations

Both SFTP and SCP offer significant security advantages over traditional, unencrypted file-transfer protocols:

- They use strong encryption to protect data in transit.
- They provide authentication mechanisms to verify the identity of both the client and server.
- They ensure data integrity, preventing unauthorized modifications during transfer.

Choosing Between SFTP and SCP

When deciding between SFTP and SCP, consider:

- **Functionality:** SFTP offers more advanced file management features.
- **Compatibility:** SCP might be the only choice in old/legacy environments where SFTP support is limited or absent.
- **Fitness:** In most modern scenarios, SFTP is the preferred choice due to its broader feature set and widespread support.