rsync.net Systems and Data Security

(Last updated October 10, 2022)

Overview

rsync.net storage arrays run the OpenSSH server software and no other server software. It is not possible to access an rsync.net storage array using any other protocol than SSH or on any other network port than TCP port 22.

The root (superuser) account on these arrays is not accessible from the network. The root account can only be accessed by a specific user which, itself, can only be accessed from a protected "jumphost". This jumphost access is enforced by an IP restriction in the "jump user" authorized_keys file which is then locked immutable with chflags.

There is no connection between tasks and processes running off of the storage array (such as stats collection or debugging or business processes) that utilize the root account on the storage arrays *or cause any processes to be run, as root, on the storage arrays.* The root account on an rsync.net storage array is used ONLY for housekeeping processes confined to that particular storage array OR for non-automated, specific tasks that are run *manually, by a human being*.

All such access to the rsync.net storage arrays is performed by full-time, salaried employees of rsync.net who have been background checked[1] and is performed with an SSH key that contains a passphrase (the SSH key of the "jump user"). Access requires "something we have" combined with "something we know". Normal password authentication is disabled for the "jump user" who can elevate to root.

The SSH keys and the key passphrases are *different* for each rsync.net storage array.

These SSH keys are deployed only to approved, secured devices running approved operating systems. These devices cannot be mobile phones.

Attack Surface

An rsync.net storage array exposes almost zero attack surface - only TCP port 22 (which can be verified with nmap - for instance, against usw-s005.rsync.net).

All of the typical attack surface related to mail service, serving of web pages, database servers, etc., take place on servers that contain no customer backup data *and which have no ability to interact with the storage arrays as the root user*. The "jump user" public key for any of the rsync.net storage arrays does not exist on any of our utility (web, mail, DB, etc.) servers. All interaction

between these servers and rsync.net storage arrays (for the purpose of usage, stats, etc.) is performed as an unprivileged user.

This means that there is no connection between the attack surface of these utility systems and the servers that they run (httpd, sendmail, inetd, etc.) and the rsync.net storage arrays.

rsync.net Storage Arrays

Every account deployed on an rsync.net storage array is deployed in its own "jail". The inability of one account to traverse into another account is enforced not on an application, or configuration level, but on a low level kernel and filesystem level.

Every rsync.net filesystem on every rsync.net storage array is mounted noexec,nosuid. This means that it is impossible for an rsync.net customer to execute arbitrary code that they upload. **In fact, it is impossible for them to execute *any code whatsoever* from their filesystem.**

The very limited whitelist of commands that a customer can run (see: https://www.rsync.net/resources/howto/remote_commands.html) is on a separate filesystem that is locked immutable by the chflag system call.

There are no interpreters (shell, python, perl, etc.) available in this very limited environment. Helper programs that we host (such as borg, rdiff- backup, rclone, etc.) that *appear to be* python scripts are actually frozen binary executables. An actual python script would not be able to run as there is no python interpreter … or any interpreter.

Risks

It is possible, although historically extremely rare - for a remote-root OpenSSH vulnerability to be discovered. This could allow an attacker to - even one without a customer login - to gain access to an rsync.net storage array remotely and escalate privilege to root. Not only is a remote root OpenSSH vulnerability extremely rare, it is by some measures unprecedented in a FreeBSD deployment such as ours.

It is possible for the much more complicated utility servers that rsync.net runs - such as our mail server, web server, database servers, etc., to be compromised by any number of software pieces that are necessarily run on such systems. Such a compromise would, at worst, leak customer meta- data (although NOT payment card data) such as their names, addresses, emails, etc. This would not present a risk to the actual data stored on the actual rsync.net storage arrays.

It is possible that one of our employees could have their laptop or phone stolen which would allow an attacker to access, for a short period of time, our customer database and internal email server. Again, this would not present a risk to the actual data stored on the actual rsync.net storage arrays.

It is possible that an attacker could compromise an employee laptop and install a "rootkit", such as a keylogger combined with a traffic log and local filesystem access. In this scenario, the attacker would eavesdrop on the connection to our secure jumphost (whose IP address is kept hidden) *and* would have access to the public key of the "jump user" that is used to login *and* could log the keystrokes of the passphrase of that key. However, even in this very dangerous and compromised scenario, the attacker would then find it impossible to make an SSH connection to the jumphost to use those credentials. rsync.net does not disclose how this is achieved - it is a method whose security rests in its secrecy. In such a scenario, the attacker would possess the SSH public key of the "jump user" and would know the passphrase for that key, but their SSH connection to the jumphost would time out. Further, their attempts to connect to the jumphost without this secret and out of band authentication would very clearly indicate an attack and would cause the jumphost to shut down until we could intervene. *This scenario has never occurred.*

It is possible that an rsync.net employee could attack rsync.net from within and destroy or exfiltrate customer data from the rsync.net storage arrays. The main mitigation against this threat is disallowing all access to rsync.net storage arrays except from full-time, salaried, background checked employees. **Contractors and datacenter partners, etc., *have no access of any kind* to the rsync.net storage arrays.**

Secrets

The vast majority of the security of the rsync.net storage arrays is dependent not on secrets, but on secure configuration - so it is possible for rsync.net to publish almost every piece of the configuration of these systems without compromising their security.

There are two items, however, whose security relies on their secrecy and they are:

1. The address of our SSH jumphost

2. The final, out of band, authentication information that allows our jump user to SSH to our jumphost

Every other piece of our configuration is public knowledge and is disclosable - its security rests in the configuration, not its secrecy.

[1] rsync.net uses Goodhire to perform background checks on employees.