ATHENIAN TECH

Remote Unauthenticated Code Execution Vulnerability in OpenSSH Server - regreSSHion

Aug, 2024

TABLE OF CONTENTS

Introduction

This report provides a detailed analysis of **regreSSHion,** the CVE-2024-6387 vulnerability that came to light in OpenSSH's server (sshd), on July 1, 2024, that was identified by the Qualys Threat Research Unit (TRU). This report aims to inform stakeholders, including IT security professionals, system administrators, and software developers, about the nature and severity of the CVE-2024-6387 vulnerability. This study also outlines the potential risks and gives guidance on measures to mitigate, the said vulnerability, and best practices to adopt, to secure systems affected, by this vulnerability.

Scope

This report focuses on the regreSSHion (CVE-2024-6387) vulnerability affecting OpenSSH server (sshd) versions. This study includes an overview of the vulnerability, its impact on security (of systems using OpenSSH's servers), and the conditions under which remote unauthenticated code execution (RCE) occurs as a result of this vulnerability. Additionally, this report dwells on the regression nature of the vulnerability, its discovery timeline, and the implications for system security and patch management practices.

Executive Summary

A critical Remote Unauthenticated Code Execution (RCE) vulnerability in OpenSSH's server (sshd), was caused by a timing flaw in the signal handling process of the sshd server that allowed unauthenticated remote code execution as root, hence posing a significant security risk for the systems using the vulnerable versions of OpenSSH server. Searches using Censys and Shodan revealed that over 14 million potentially vulnerable OpenSSH servers were exposed to the Internet, representing 31% of all Internet-facing OpenSSH instances globally.

This critical vulnerability was publicly disclosed on July 1, 2024, highlighting the risk of unauthenticated remote attackers exploiting the flaw in the vulnerable OpenSSH versions. This vulnerability has been rated High severity (CVSS 8.1). Even after the disclosure of this vulnerability, OpenSSH continues to remain a model of secure software design.

Methodology

This report employs a rigorous methodology to investigate and analyse regreSSHion (CVE-2024-6387), a critical Remote Unauthenticated Code Execution (RCE) vulnerability. The methodology starts with a comprehensive risk assessment using the Common Vulnerability Scoring System (CVSS) to evaluate the severity of CVE-2024-6387. This includes assessing the potential impact on affected systems, concentrating on risks posed by unauthorised access and the potential compromise of the system's integrity and confidentiality.

The scope and impact analysis define the vulnerable systems, particularly emphasising on glibc-based Linux environments running susceptible versions of OpenSSH. This analysis also considers the broader implications for organisational security, highlighting the exposure of Internet-facing sshd that were open to exploitation. Quantitative analysis using tools like Censys and Shodan further quantifies risks the vulnerability exposed the systems to. This involves gathering empirical data on the global prevalence and geographical distribution of affected systems.

Vulnerability Details

Description of CVE-2024-6387

regreSSHion is a high-severity vulnerability in the OpenSSH server versions 8.5p1 to 9.8p1 (sshd) that affects Linux systems. This flaw allows attackers to run harmful code on a server without the need for any passwords or permissions from the administrator, potentially giving the intruder full control of the system.

This problem is caused by a flaw in how the server handles connection signals. When the server tries to log information (using a function called syslog()), it uses unsafe functions (such as malloc() and free()) that can be exploited by attackers. This happens because the server's critical code is not sandboxed properly and thus gives full privileges to the intruder.

- CVEID: CVE-2024-6387
- CVSS Base score: 8.1
- CVSS Temporal Score: 7.3
- CVSS Vector: (CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H)

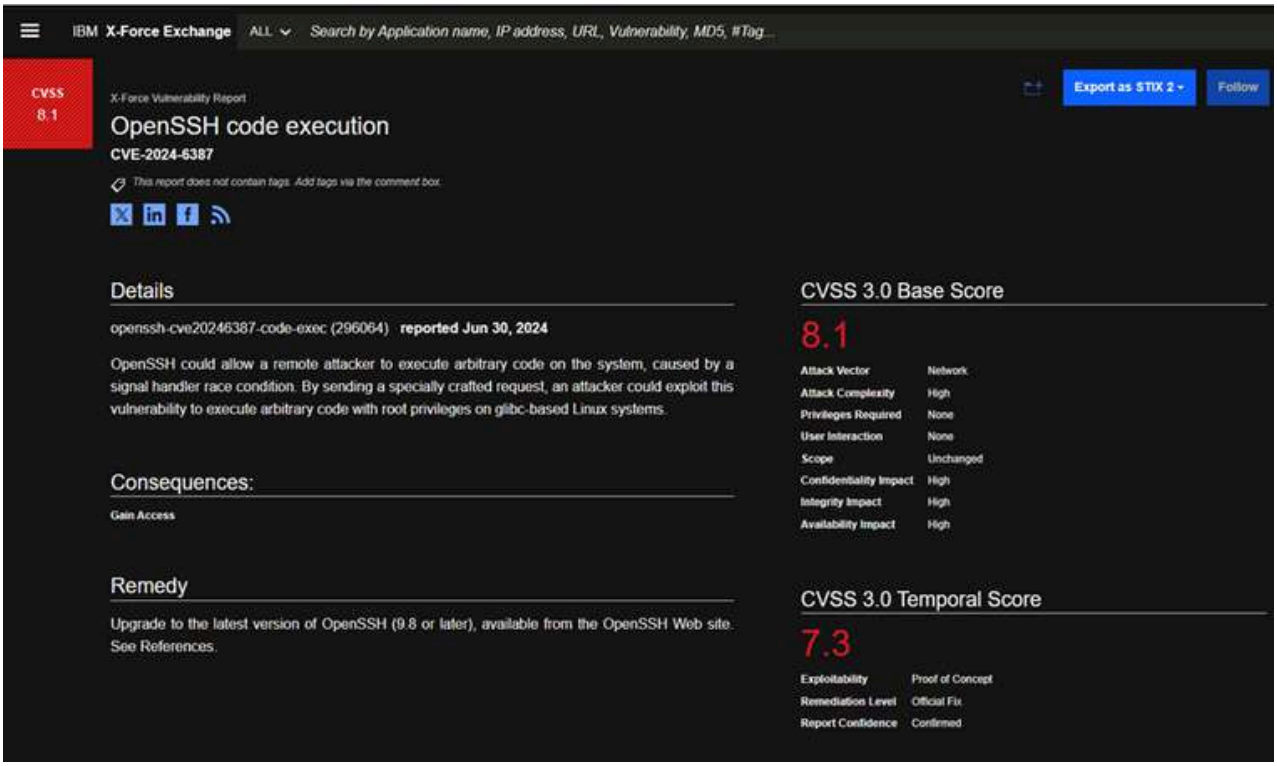

Figure 1: CVSS Score of regreSSHion vulnerability

Vulnerable Versions and Configurations

OpenSSH Version Range	Vulnerability Status
Earlier than 4.4p1	Vulnerable unless patched for CVE-2006-5051 and CVE-2008-4109
4.4p1 <= OpenSSH < 8.5p1	Not vulnerable due to a transformative patch for CVE-2006-5051
8.5p1 <= OpenSSH < 9.8p1	Vulnerable due to the accidental removal of a critical component in a function
OpenBSD systems	Not affected due to their secure mechanism preventing this vulnerability

Table 1: Vulnerable versions of OpenSSH

Technical Analysis

Overview of CVE-2024-6387 Vulnerability

CVE-2024-6387 is a critical vulnerability stemming from a signal handler race condition in OpenSSH's server (sshd). The vulnerability is a regression of a previously patched issue, CVE-2006-5051, and allows remote attackers to execute arbitrary code as root, leading to full system compromise.

Nature of the Signal Handler Race Condition

The vulnerability occurs due to the improper handling of the SIGALRM signal in the affected versions of OpenSSH. When a user fails to authenticate within the LoginGraceTime period (default time 120 seconds in versions (8.5p1 and beyond) and 600 seconds (in versions prior to 4.4p1), sshd's SIGALRM handler is triggered. This handler calls various functions, like syslog(), that are unsafe to call from within a signal handler. This creates a race condition where the signal handler can interrupt critical sections of code, potentially leaving the system in an inconsistent state.

Relation to CVE-2006-5051

This issue is a regression of the older vulnerability CVE-2006-5051, which was identified and patched in 2006. The regression occurred in October 2020 with the release of OpenSSH 8.5p1, during an update to the logging infrastructure. A crucial directive (#ifdef DO_LOG_SAFE_IN_SIGHAND) was inadvertently removed, making the sigdie() function unsafe again.

Specific Conditions for Exploit the Vulnerability

Timing Requirements

The exploit requires precise timing to hit the race condition window. Successful exploitation typically requires around 10,000 attempts.
The following pseudocode for the exploit demonstrates the precise timing required to interrupt the signal handler during memory allocation or deallocation functions like malloc() or free():

```
int attempt_race_condition(int sock, double parsing_time, uint64_t glibc_base) {
unsigned char final_packet[MAX_PACKET_SIZE];
create_public_key_packet(final_packet, sizeof(final_packet), glibc_base);

// Send all but the last byte
if (send(sock, final_packet, sizeof(final_packet) - 1, 0) < 0) {
    perror("send final packet");
    return 0;
}

// Precise timing for last byte
struct timespec start, current;
clock_gettime(CLOCK_MONOTONIC, &start);
```

```
while (1) {
 clock_gettime(CLOCK_MONOTONIC, &current);
 double elapsed = (current.tv_sec - start.tv_sec)
 + (current.tv_nsec - start.tv_nsec) / 1e9;
 if (elapsed >= (LOGIN_GRACE_TIME - parsing_time - 0.001)) { // 1ms before SIGALRM
 if (send(sock, &final_packet[sizeof(final_packet) - 1], 1, 0) < 0) {
 perror("send last byte");
 return 0;
 }
 break;
 }
}
}
```

Exploitation Mechanics

The exploit follows these steps:

- **Establish Connection:** Initiate multiple SSH connections to the target server, triggering the LoginGraceTime limit. The first step is to direct several SSH connections to the vulnerable server. The LoginGraceTime is the period during which the server waits for a user to authenticate, before terminating the connection. By default, this is set to 120 seconds, providing a window for potential exploitation. The attacker leverages this period to initiate multiple connections, thus creating the conditions necessary for the exploit.

- **Heap Manipulation:** Send a sequence of packets to manipulate the server's memory allocation patterns (heap spraying). Heap spraying involves sending a large number of specially crafted packets to the server to fill the heap (a memory region used for dynamic memory allocation) with specific data patterns. The goal is to arrange the heap in a predictable state that makes it easier to exploit the vulnerability. The attacker carefully constructs packets to influence how the memory is allocated and deallocated, setting up the server's memory to facilitate the exploitation of race conditions.

- **Timing the Exploit:** Measure server response times to fine-tune the timing of the exploit. The critical step involves sending a carefully timed packet to exploit the signal handler race condition. Exploiting the race condition requires precise timing because the vulnerability arises from mishandling of signals. The attacker measures the server's response times to fine-tune the exploit's timing. This involves sending packets at specific intervals to ensure that they hit the race condition window. The critical step here is to send a packet at a moment when the server is handling a signal, such as an alarm signal triggered by LoginGraceTime, and is in the middle of a sensitive operation like memory allocation or deallocation.

- **Arbitrary Code Execution:** Upon successful exploitation, deliver a shellcode payload for privileged execution. The attacker can execute arbitrary code once the race condition is successfully exploited. This typically involves delivering a payload, often referred to as shellcode, which runs with the same privileges as the vulnerable process. Since the OpenSSH server operates with root privileges, the exploit allows the attacker to execute commands with full administrative rights. This can lead to complete system compromise, allowing the attacker to install malware, create backdoors, or manipulate data.

Proof-of-Concept

We created a demo SSH server running version 8.5p1, which is vulnerable to regreSSHion (CVE-2024-6387). The exploit works by initiating multiple SSH connections to trigger the LoginGraceTime limit, stressing the server. Packets are sent to manipulate the server's memory allocation, a technique called heap spraying. The timing of these packets is fine-tuned based on server response time to exploit the race condition in the signal handler. This demonstration confirmed the vulnerability in our demo server.

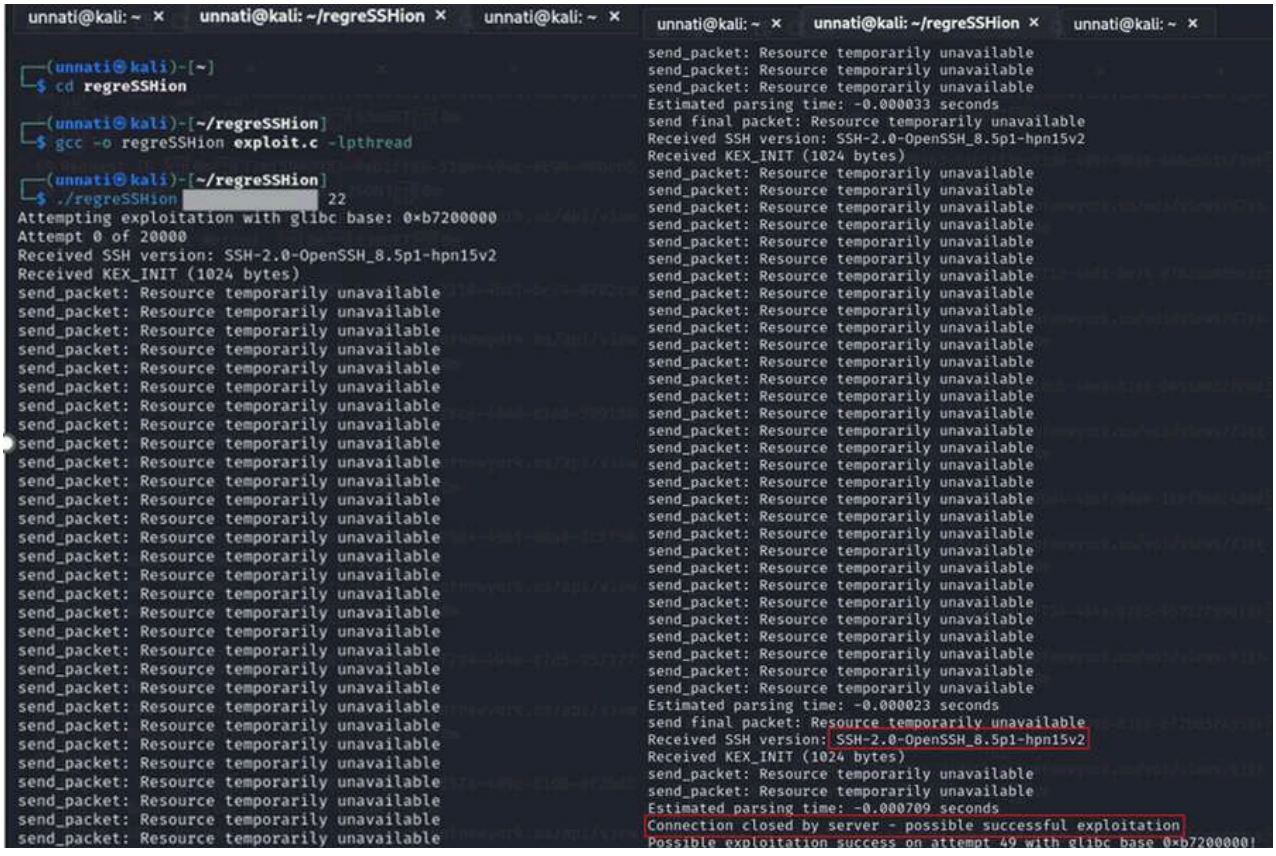

Figure 2: Running the exploit to confirm if the target server is vulnerable to regreSSHION

Figure 3: Exploit, confirming possibility of successful exploitation in the vulnerable SSH server

Impact Analysis

Potential Impact on Affected Systems

The OpenSSH vulnerability, affecting most Linux distributions, allows remote unauthenticated code execution with root privileges that leads to full system compromise. Once in, the attackers can install malware, manipulate data, and create persistent backdoors, hence bypassing security mechanisms like firewalls and intrusion detection systems (IDS). This can result in significant data breaches and provide a foothold for further network propagation, compromising additional systems within the organisation.

Following are some major impacts of regreSSHion:

- **System Compromise:**
  - Full system takeover by executing arbitrary code with root privileges.
  - Installation of malware and creation of persistent backdoors.
  - Manipulation and corruption of data.
- **Network Propagation:**
  - Use of compromised systems as a foothold to exploit other vulnerable systems within the organisation.
- **Bypassing Security Mechanisms:**
  - Evasion of firewalls, intrusion detection systems, and logging mechanisms.
  - Stealthy activities leading to difficult detection and response.
- **Data Breaches:**
  - Access to all data on the compromised systems, including sensitive or proprietary information.
  - Potential theft or public disclosure of valuable data.

Affected User Demographics

Country	Unique IP Addresses (in numbers)
United States	21,73,896
Germany	9,05,859
China	4,35,490
Singapore	2,96,226
Russia	2,75,197
The Netherlands	2,61,212
France	2,48,153
United Kingdom	2,37,329
India	2,30,320
Japan	2,27,663
Korea	1,36,852
Canada	1,19,924
Finland	1,10,516
Hong Kong	1,03,685
Australia	1,00,780

Table 2: Number of organisations affected by regreSSHion

Reference: https://www.akamai.com/blog/security-research/2024-openssh-vulnerability-regression-what-to-know-and-do

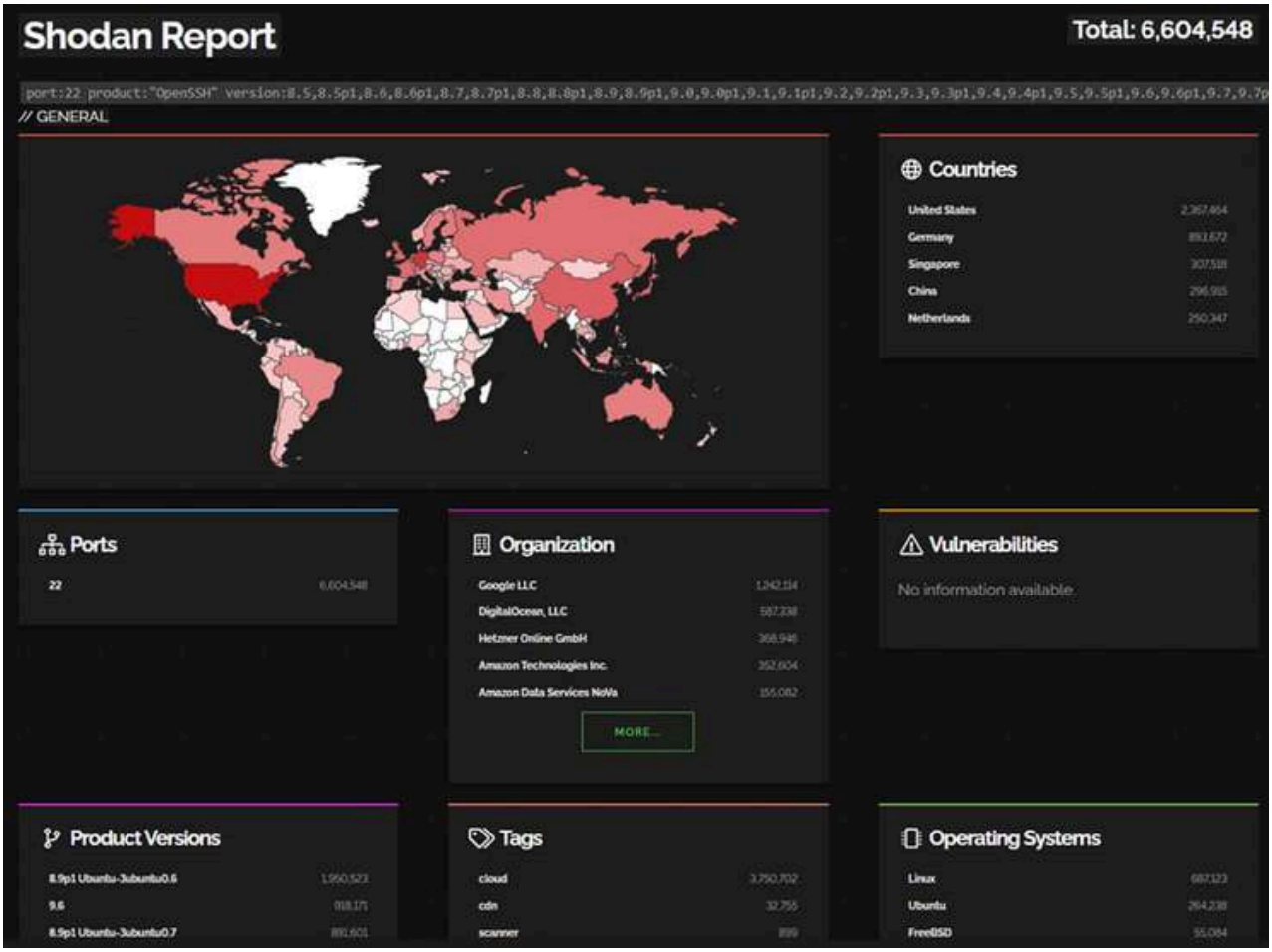

Figure 4: Shodan report for vulnerable systems over the internet

Reference: https://www.akamai.com/blog/security-research/2024-openssh-vulnerability-regression-what-to-know-and-do

Mitigation Strategies

The most effective strategy for countering the regreSSHion vulnerability is to update all vulnerable devices to the latest version, OpenSSH 9.8/9.8p1. If updating the systems is not possible, then other mitigation strategies include the following:

- **Reducing LoginGraceTime:** Shortening the LoginGraceTime period can significantly reduce the window of exploitation.

- **Implementing Network-level Mitigations:** Using network-level controls can help prevent exploitation attempts by making it harder for attackers to repeatedly connect and then try to exploit the vulnerability.

- **Firewalls:**
  - Configure firewalls to restrict access to SSH ports (default port 22) to known IP addresses or networks.
  - Use rate limiting to control the number of connection attempts from any single IP address.

- **Intrusion Detection/Prevention Systems (IDS/IPS):**
  - Deploy IDS/IPS to monitor and block suspicious activities and repeated attempts to connecting to the server.
  - Create rules to detect and block patterns associated with exploitation attempts, such as multiple connection attempts within a short period of time (which in the case of regreSSHion is 120 seconds).

- **Monitoring and Alerts:** Setting up monitoring and alerting for suspicious SSH connection patterns and high connection attempts.

General Security Practices for OpenSSH

- **Disable Password Authentication:** Use SSH key-based authentication instead of passwords whenever possible. SSH keys offer more robust security by eliminating the risk of brute-forcing passwords.

- **Keep OpenSSH Updated:** Regularly update OpenSSH to the latest version to patch known vulnerabilities and apply the latest security enhancements.

- **Limit SSH Access:** Restrict SSH access to only necessary users and hosts using firewall rules and SSH configuration settings (AllowUsers, AllowGroups, DenyUsers, DenyGroups).

- **Monitor SSH Logs:** Monitor SSH logs for unusual or suspicious activities, such as failed login attempts or unauthorised access attempts.

- **Use Strong Encryption:** Configure OpenSSH to protect data in transit using robust encryption algorithms and key exchange methods (e.g., AES-256, RSA, or ECDSA keys).

- **Implement Two-Factor Authentication (2FA):** Require two-factor authentication for SSH access to enhance security, adding an extra layer of protection even if SSH keys are compromised.

- **Disable Unused Features:** Disable any unused OpenSSH features or protocols to reduce the attack surface and potential vulnerabilities (e.g., SSHv1).

- **Implement Firewall Rules:** Use firewall rules to restrict SSH access to specific IP addresses or networks, limiting exposure to potential attacks.

By following these practices, organisations can enhance the security of their OpenSSH deployments, safeguarding against unauthorised access and potential security threats.

Conclusion

The regreSSHion (CVE-2024-6387) vulnerability, poses a significant threat to the security and integrity systems running on Linux OS systems running OpenSSH versions 8.5p1 to 9.8p1. This vulnerability is caused by a timing flaw in the signal handling process of the OpenSSH server that allows unauthenticated remote code execution as root, potentially leading to full system compromise. Despite the critical nature of this flaw, effective mitigation strategies are available such as, updating to the latest OpenSSH version (9.8/9.8p1) is the most robust defence against this vulnerability. For systems where immediate updating of the vulnerable version of OpenSSH server is not feasible, additional measures such as reducing the LoginGraceTime period, implementing network-level mitigations, and setting up robust monitoring and alerting mechanisms can help to minimise the risk of exploitation.

OpenSSH's central role in modern infrastructure, that is, providing secure remote access and data communication, underscores the importance of promptly addressing this vulnerability. The widespread deployment of OpenSSH across diverse sectors and its critical functions in secure server administration makes the remediation of CVE-2024-6387 a high priority for maintaining the confidentiality, integrity, and availability of the systems vulnerable to regreSSHion.

Athenian Tech

Contact Us

+91 9999326349

Ground Floor, Building No. EP, 2731, Golf Course Ext Rd, Block A,
Sushant Lok III, Sector 57, Gurugram, Haryana 122003

kanishk.gaur@atheniantech.com

www.atheniantech.com

https://www.linkedin.com/company/athenian-tech/