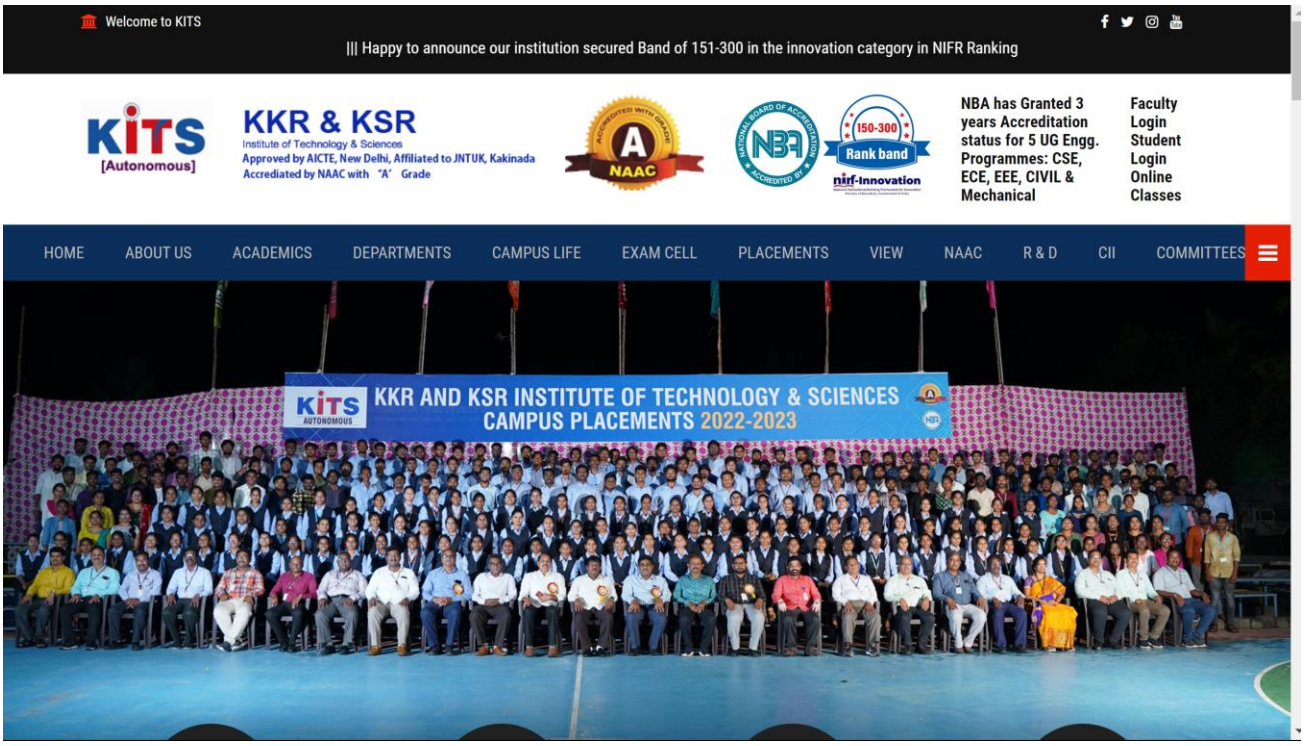
SUBMITTED BY: Avvari Pratheek

REGISTRATION NO: 21BCE7819

E-mail: pratheek.21bce7819@vitapstudent.ac.in

WEBSITE-

https://kitsguntur.ac.in/site/kits.php



IPV4 ADDRESS -

108.181.19.209

IPv4 address	Revalidate in
108.181.19.209	4h

VULNERABILITY NAME-

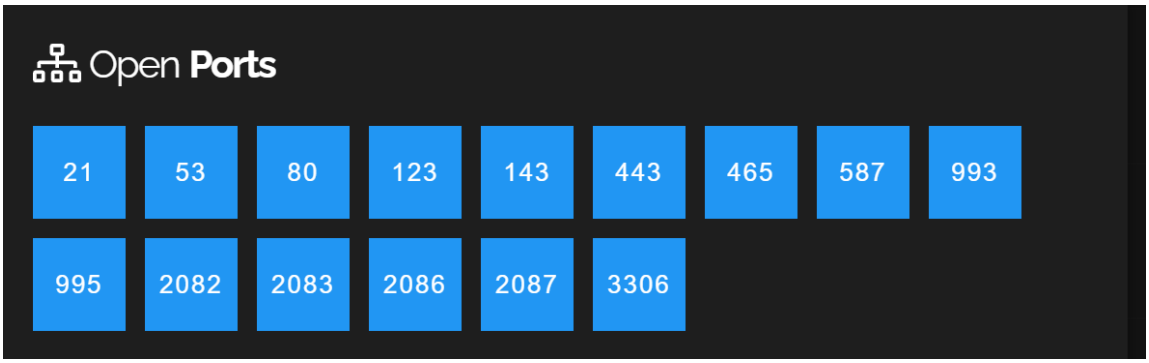
CVE-2022-37452

CVE-2022-37452	Exim before 4.95 has a heap-based buffer overflow for the alias list in host_name_lookup in host.c when sender_host_name is set.

Description

Exim before 4.95 has a heap-based buffer overflow for the alias list in host_name_lookup in host.c when sender_host_name is set.

OPEN PORTS -



CWE-787: Out-of-bounds Write

Description

The product writes data past the end, or before the beginning, of the intended buffer.

BUSSINESS IMPACT

Common Weakness Enumeration (CWE) identifier 787, also known as "Out-of-Bounds Write," refers to a type of software vulnerability where a program writes data beyond the boundaries of an allocated memory buffer. This can result in a range of unpredictable and potentially severe consequences, leading to various business impacts. Here are some potential business impacts associated with CWE-787:

1. Data Corruption: Out-of-bounds writers can overwrite critical data structures or memory regions, leading to data corruption. This can result in incorrect calculations, inaccurate reporting, and the potential for data loss, which can negatively impact business operations and decision-making.

2. Application Crashes: When an out-of-bounds write occurs, it can destabilize the application, causing it to crash or become unresponsive. This can disrupt user experiences, damage the organization's reputation, and potentially lead to lost revenue if the application is a critical part of the business.

3. Security Vulnerabilities: Out-of-bound writers can also be exploited by malicious actors to execute arbitrary code, bypass security mechanisms, or gain unauthorized access to sensitive data. This can result in data breaches, financial losses, and legal consequences if customer or proprietary information is compromised.

To mitigate the business impacts associated with CWE-787, organizations should prioritize secure coding practices, conduct regular code reviews and security

assessments, and implement runtime protections like bounds checking to prevent out-of-bounds writes from occurring in their software applications.

OSWAP

Buffer Overflow

Buffer overflow errors are characterized by the overwriting of memory fragments of the process, which should have never been modified intentionally or unintentionally. Overwriting values of the IP (Instruction Pointer), BP (Base Pointer), and other registers cause exceptions, segmentation faults, and other errors to occur. Usually, these errors end the execution of the application in an unexpected way. Buffer overflow errors occur when we operate on buffers of char type.

Buffer overflows can consist of overflowing the stack [Stack overflow] or overflowing the heap [Heap overflow]. We do not distinguish between these two in this article to avoid confusion.

Below examples are written in C language under GNU/Linux system on x86 architecture.

Description

Buffer overflow is the best-known form of software security vulnerability. Most software developers know what a buffer overflow vulnerability is, but buffer overflow attacks against both legacy and newly developed applications are still quite common. Part of the problem is due to the wide variety of ways buffer overflows can occur, and part is due to the error-prone techniques often used to prevent them.

Buffer overflows are not easy to discover and even when one is discovered, it is extremely difficult to exploit. Nevertheless, attackers have managed to identify buffer overflows in a staggering array of products and components.

In a classic buffer overflow exploit, the attacker sends data to a program, which it stores in an undersized stack buffer. The result is that information on the call stack is overwritten, including the function’s return pointer. The data sets the value of the return pointer so that when the function returns, it transfers control to malicious code contained in the attacker’s data.

Although this type of stack buffer overflow is still common on some platforms and in some development communities, there are a variety of other types of buffer overflow, including Heap buffer overflow and Off-by-one Error, among others. Another remarkably similar class of flaws is known as Format string attack. There are several excellent books that provide detailed information on how buffer overflow attacks work, including Building Secure Software [1], Writing Secure Code [2], and The Shellcoder's Handbook [3].

At the code level, buffer overflow vulnerabilities usually involve the violation of a programmer's assumptions. Many memory manipulation functions in C and C++ do not perform bounds checking and can easily overwrite the allocated bounds of the buffers they operate upon. Even bounded functions, such as strncpy(), can cause vulnerabilities when used incorrectly. The combination of memory manipulation and mistaken assumptions about the size or makeup of a piece of data is the root cause of buffer overflows.

Buffer overflow vulnerabilities typically occur in code that:

- Relies on external data to control its behavior
- Depends upon properties of the data that are enforced outside of the immediate scope of the code
- Is so complex that a programmer cannot accurately predict its behavior

PRODUCTS AFFECTED BY CVE-2022-37452

CPE	Name	Operator	Version
exim:exim	exim	lt	4.95
debian:debian_linux	debian debian linux	eq	10.0

PROOF OF CONCEPT

```
if (!(png_ptr->mode & PNG_HAVE_PLTE)) {
  /* Should be an error, but we can cope with it */
  png_warning(png_ptr, "Missing PLTE before tRNS");
}
else if (length > (png_uint_32)png_ptr->num_palette) {
  png_warning(png_ptr, "Incorrect tRNS chunk length");
  png_crc_finish(png_ptr, length);
  return;
}
...
png_crc_read(png_ptr, readbuf, (png_size_t)length);
```

REMEDIATION

- **Understand the Vulnerability:** Obtain detailed information about CVE-2023-38408, including its nature, severity, and potential impact. You can usually find this information in the CVE entry, security advisories, or from security researchers.

- **Identify Affected Systems:** Determine which systems or software in your environment are vulnerable to CVE-2023-38408. Conduct a thorough inventory and assessment.

- **Apply Patches or Updates:** Check if the software vendor or project maintainers have released patches or updates to address the vulnerability. If they have, apply these fixes as soon as possible.

- **Implement Workarounds:** If patches are not immediately available or you cannot apply them for some reason, consider implementing temporary workarounds to mitigate the risk. These workarounds may involve disabling specific features or restricting access.

- **Monitor for Exploitation:** Continuously monitor your systems and networks for signs of exploitation related to CVE-2023-38408. This includes reviewing logs and intrusion detection systems.

- **Review Access Controls:** Ensure that access controls and permissions are appropriately configured to limit exposure to the vulnerability.

- **Communication:** Communicate the vulnerability and remediation steps to relevant stakeholders within your organization, including IT teams, system administrators, and management.

- **Testing:** After applying patches or implementing workarounds, thoroughly test your systems to ensure that the remediation efforts do not introduce contemporary issues or disrupt critical services.

- **Review Security Policies:** Take this opportunity to review and potentially update your organization's security policies and procedures to prevent similar vulnerabilities in the future.

- **Keep Informed:** Stay informed about security developments and added information related to CVE-2023-38408. Security researchers and organizations may provide additional guidance or updates over time.