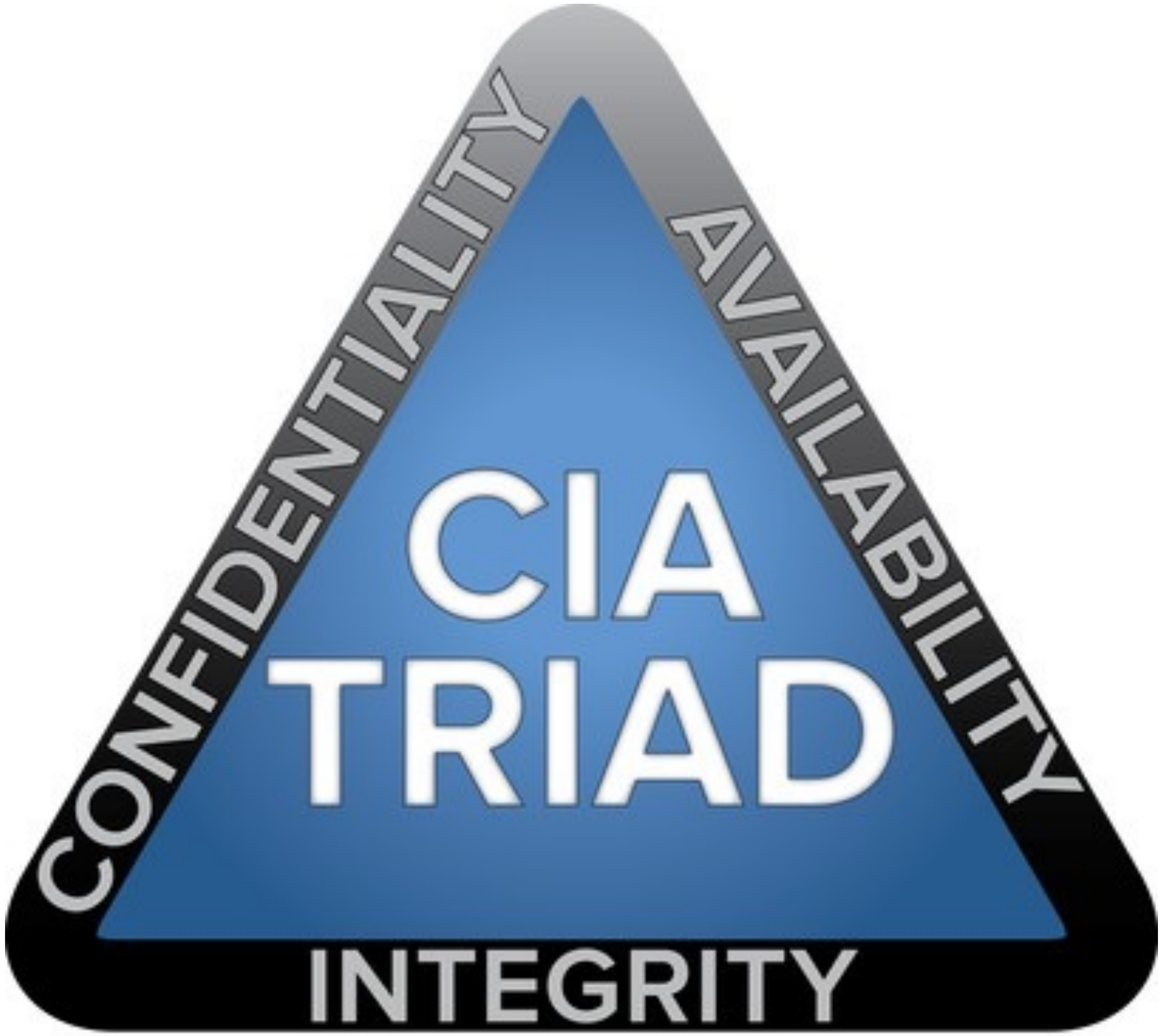

CIA TRIAD
CONFIDENTIALITY
AVAILABILITY
INTEGRITY

Confidentiality

CIA
TRIAD
CONFIDENTIALITY
AVAILABILITY
INTEGRITY

Ensuring that information is accessible only to those authorized to have access

- data encryption
- access controls
- authentication mechanisms

Integrity

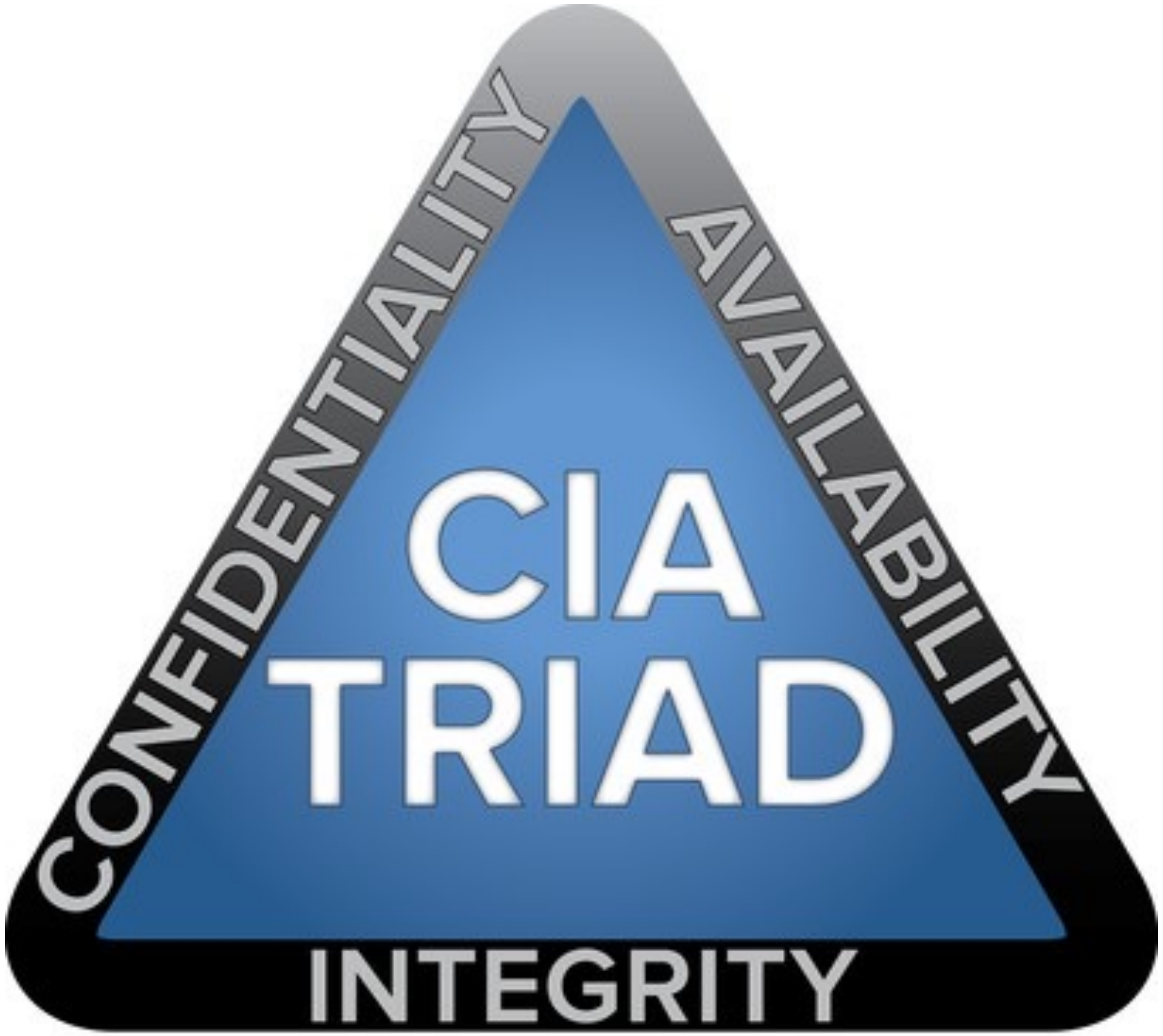
Maintaining the accuracy and consistency of data over its entire lifecycle

- checksums
- cryptographic hashes
- digital signatures
- access and modification logs

Availability

Ensuring that information and resources are accessible to authorized users when needed

- redundant systems
- backups
- failover processes
- disaster recovery plans


CIA
TRIAD
CONFIDENTIALITY
AVAILABILITY
INTEGRITY

Software Vulnerabilities

Software Vulnerabilities

1. Common Vulnerabilities
2. Software Race Condition Vulnerabilities

Software Vulnerabilities

1. Common Vulnerabilities
2. Software Race Condition Vulnerabilities

CWE - Common Weakness Enumeration

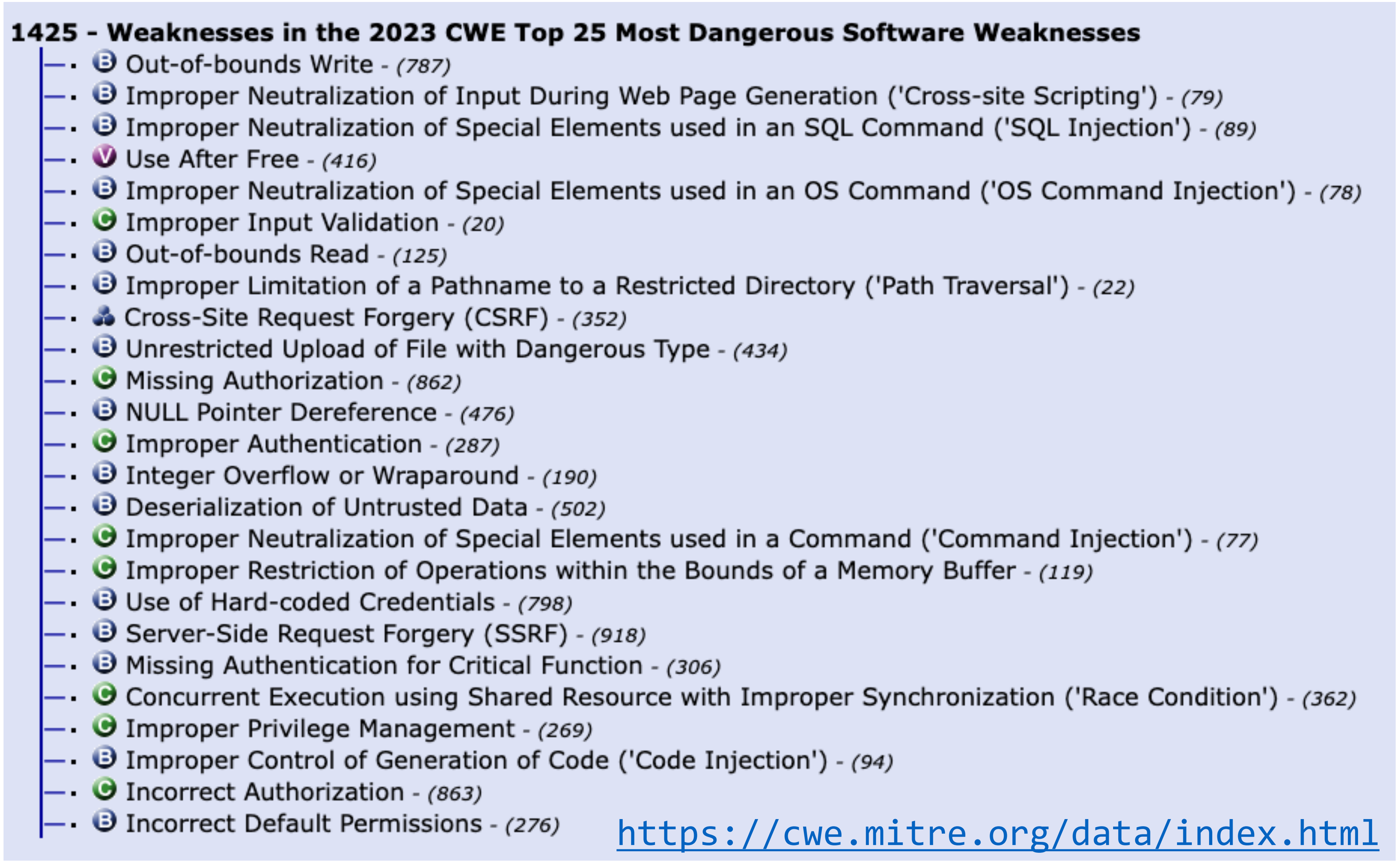
CWE is a category system for software weaknesses and vulnerabilities

- Provides a unified, measurable set of criteria for assessing the severity of software vulnerabilities

- CWE aims to serve as a common language for describing software security weaknesses in architecture, design, or code

CWE™ Common Weakness Enumeration

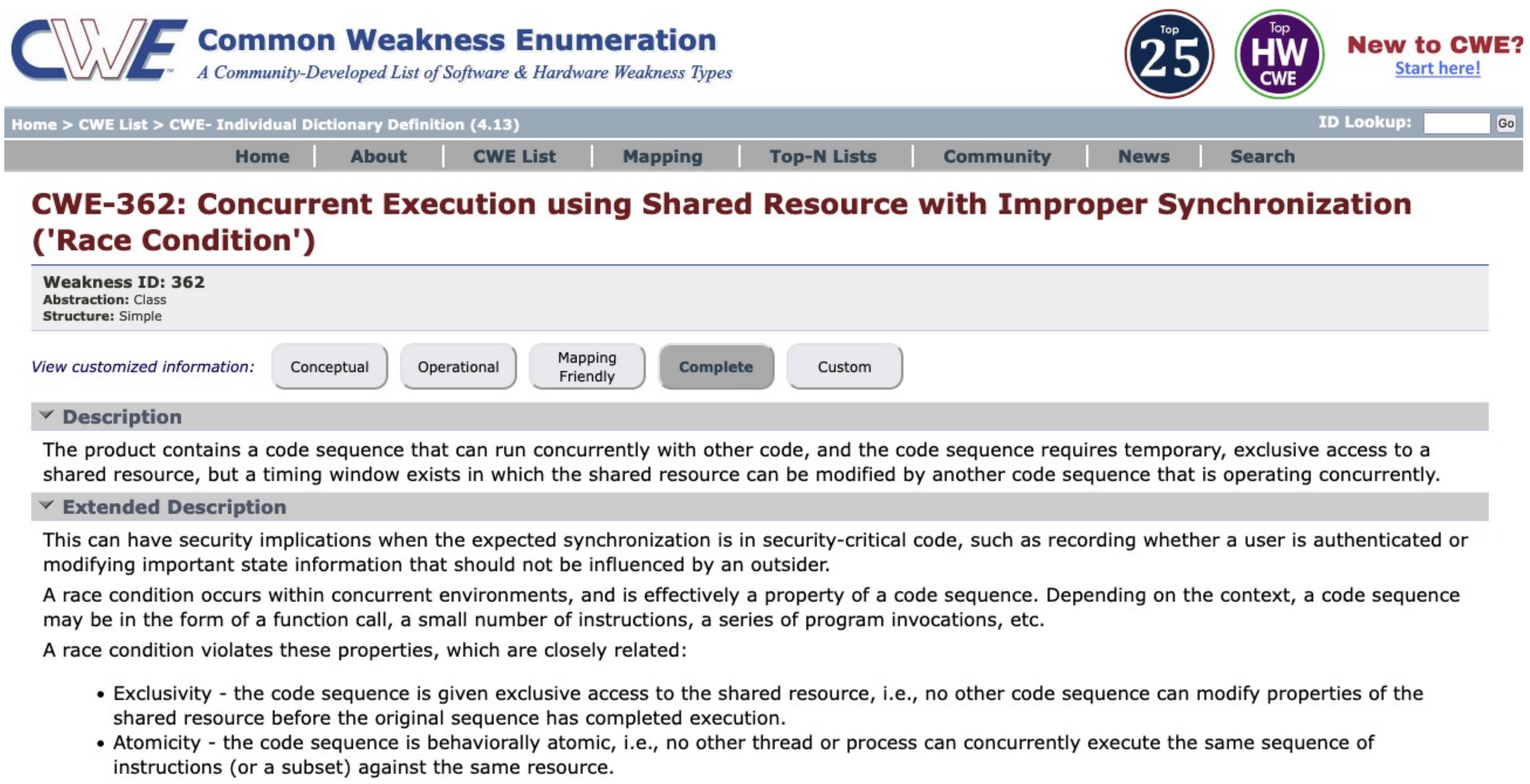
A Community-Developed List of Software & Hardware Weakness Types



1425 - Weaknesses in the 2023 CWE Top 25 Most Dangerous Software Weaknesses

- Ⓑ Out-of-bounds Write - *(787)*
- Ⓑ Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') - *(79)*
- Ⓑ Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') - *(89)*
- Ⓥ Use After Free - *(416)*
- Ⓑ Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') - *(78)*
- Ⓒ Improper Input Validation - *(20)*
- Ⓑ Out-of-bounds Read - *(125)*
- Ⓑ Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') - *(22)*
- ♣ Cross-Site Request Forgery (CSRF) - *(352)*
- Ⓑ Unrestricted Upload of File with Dangerous Type - *(434)*
- Ⓒ Missing Authorization - *(862)*
- Ⓑ NULL Pointer Dereference - *(476)*
- Ⓒ Improper Authentication - *(287)*
- Ⓑ Integer Overflow or Wraparound - *(190)*
- Ⓑ Deserialization of Untrusted Data - *(502)*
- Ⓒ Improper Neutralization of Special Elements used in a Command ('Command Injection') - *(77)*
- Ⓒ Improper Restriction of Operations within the Bounds of a Memory Buffer - *(119)*
- Ⓑ Use of Hard-coded Credentials - *(798)*
- Ⓑ Server-Side Request Forgery (SSRF) - *(918)*
- Ⓑ Missing Authentication for Critical Function - *(306)*
- Ⓒ Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') - *(362)*
- Ⓒ Improper Privilege Management - *(269)*
- Ⓑ Improper Control of Generation of Code ('Code Injection') - *(94)*
- Ⓒ Incorrect Authorization - *(863)*
- Ⓑ Incorrect Default Permissions - *(276)*

https://cwe.mitre.org/data/index.html


CWE
Common Weakness Enumeration
A Community-Developed List of Software & Hardware Weakness Types
Top 25
Top HW CWE
New to CWE?
Start here!
Home > CWE List > CWE- Individual Dictionary Definition (4.13)
ID Lookup:
Go
Home
About
CWE List
Mapping
Top-N Lists
Community
News
Search
View customized information:
Conceptual
Operational
Mapping Friendly
Complete
Custom

CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')

Weakness ID: 362
Abstraction: Class
Structure: Simple

Description

The product contains a code sequence that can run concurrently with other code, and the code sequence requires temporary, exclusive access to a shared resource, but a timing window exists in which the shared resource can be modified by another code sequence that is operating concurrently.

Extended Description

This can have security implications when the expected synchronization is in security-critical code, such as recording whether a user is authenticated or modifying important state information that should not be influenced by an outsider.

A race condition occurs within concurrent environments, and is effectively a property of a code sequence. Depending on the context, a code sequence may be in the form of a function call, a small number of instructions, a series of program invocations, etc.

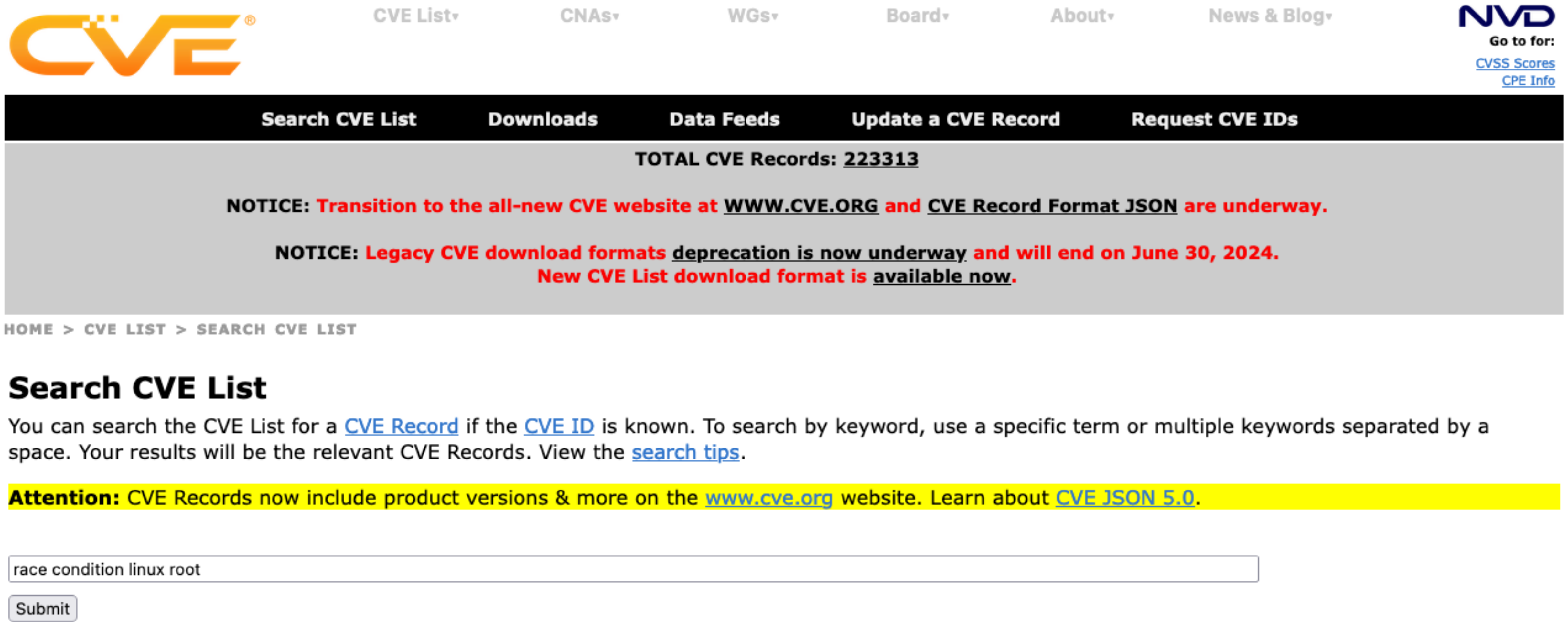
A race condition violates these properties, which are closely related:

- Exclusivity - the code sequence is given exclusive access to the shared resource, i.e., no other code sequence can modify properties of the shared resource before the original sequence has completed execution.
- Atomicity - the code sequence is behaviorally atomic, i.e., no other thread or process can concurrently execute the same sequence of instructions (or a subset) against the same resource.

CVE - Common Vulnerabilities and Exposures

CVE is a <u>specific</u> publicly disclosed cybersecurity vulnerability and exposure

- A CVE entry includes an identification number, a description
- It is focused on <u>specific vulnerabilities</u> in software and hardware
- Used to ensure that security professionals are discussing the same issue
- Serve as a common language for describing software security weaknesses



TOTAL CVE Records: 223313

NOTICE: Transition to the all-new CVE website at WWW.CVE.ORG and CVE Record Format JSON are underway.

NOTICE: Legacy CVE download formats deprecation is now underway and will end on June 30, 2024. New CVE List download format is available now.

HOME > CVE LIST > SEARCH CVE LIST

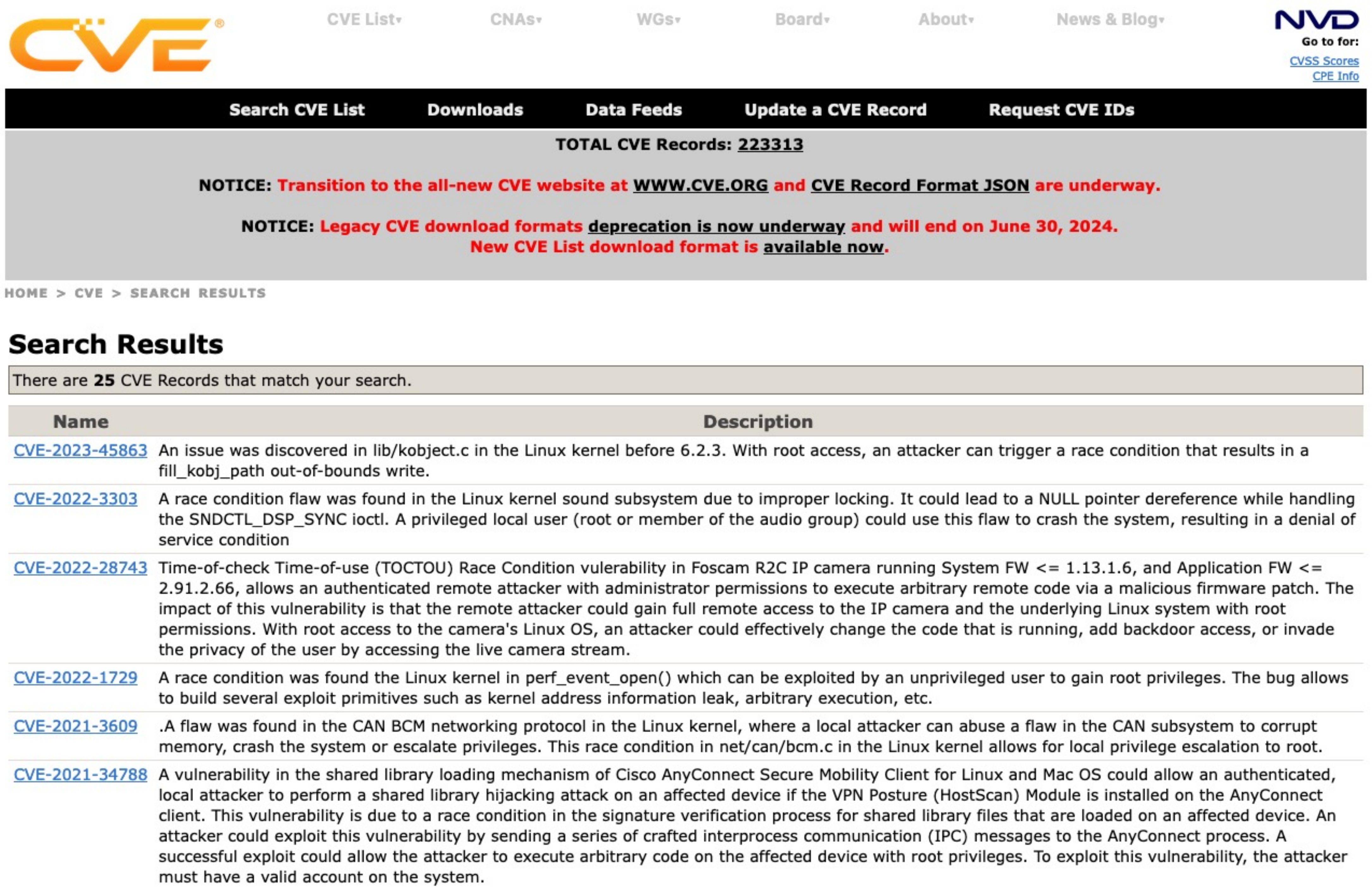
Search CVE List

You can search the CVE List for a CVE Record if the CVE ID is known. To search by keyword, use a specific term or multiple keywords separated by a space. Your results will be the relevant CVE Records. View the search tips.

Attention: CVE Records now include product versions & more on the www.cve.org website. Learn about CVE JSON 5.0.

race condition linux root

https://cve.mitre.org/cve/search_cve_list.html


CVE
CVE List▾ CNAs▾ WGs▾ Board▾ About▾ News & Blog▾
NVD
Go to for:
CVSS Scores
CPE Info
Search CVE List Downloads Data Feeds Update a CVE Record Request CVE IDs

TOTAL CVE Records: 223313

NOTICE: Transition to the all-new CVE website at WWW.CVE.ORG and CVE Record Format JSON are underway.

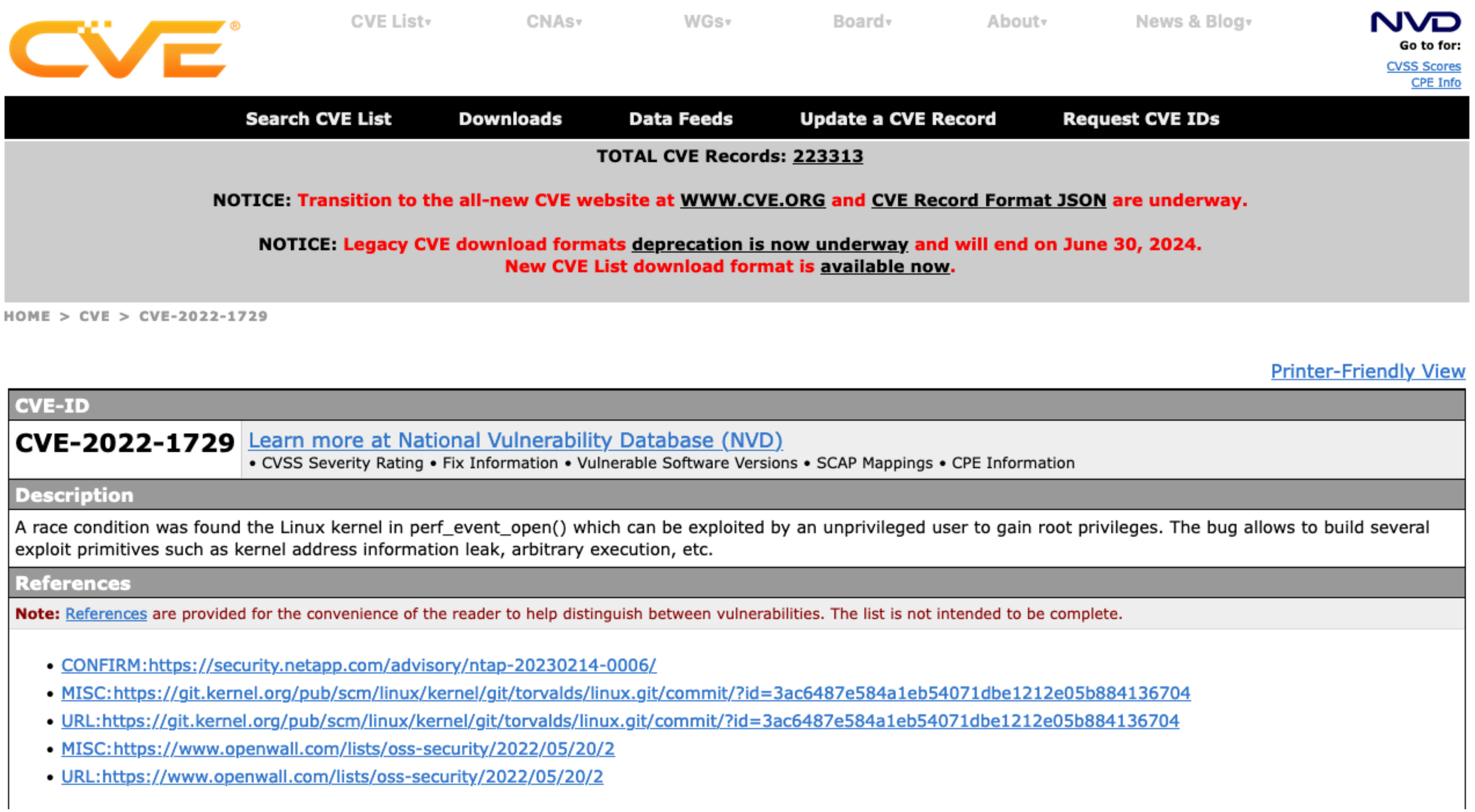
NOTICE: Legacy CVE download formats deprecation is now underway and will end on June 30, 2024.
New CVE List download format is available now.

HOME > CVE > SEARCH RESULTS

Search Results

There are **25** CVE Records that match your search.

Name	Description
CVE-2023-45863	An issue was discovered in lib/kobject.c in the Linux kernel before 6.2.3. With root access, an attacker can trigger a race condition that results in a fill_kobj_path out-of-bounds write.
CVE-2022-3303	A race condition flaw was found in the Linux kernel sound subsystem due to improper locking. It could lead to a NULL pointer dereference while handling the SNDCTL_DSP_SYNC ioctl. A privileged local user (root or member of the audio group) could use this flaw to crash the system, resulting in a denial of service condition
CVE-2022-28743	Time-of-check Time-of-use (TOCTOU) Race Condition vulerability in Foscam R2C IP camera running System FW <= 1.13.1.6, and Application FW <= 2.91.2.66, allows an authenticated remote attacker with administrator permissions to execute arbitrary remote code via a malicious firmware patch. The impact of this vulnerability is that the remote attacker could gain full remote access to the IP camera and the underlying Linux system with root permissions. With root access to the camera's Linux OS, an attacker could effectively change the code that is running, add backdoor access, or invade the privacy of the user by accessing the live camera stream.
CVE-2022-1729	A race condition was found the Linux kernel in perf_event_open() which can be exploited by an unprivileged user to gain root privileges. The bug allows to build several exploit primitives such as kernel address information leak, arbitrary execution, etc.
CVE-2021-3609	.A flaw was found in the CAN BCM networking protocol in the Linux kernel, where a local attacker can abuse a flaw in the CAN subsystem to corrupt memory, crash the system or escalate privileges. This race condition in net/can/bcm.c in the Linux kernel allows for local privilege escalation to root.
CVE-2021-34788	A vulnerability in the shared library loading mechanism of Cisco AnyConnect Secure Mobility Client for Linux and Mac OS could allow an authenticated, local attacker to perform a shared library hijacking attack on an affected device if the VPN Posture (HostScan) Module is installed on the AnyConnect client. This vulnerability is due to a race condition in the signature verification process for shared library files that are loaded on an affected device. An attacker could exploit this vulnerability by sending a series of crafted interprocess communication (IPC) messages to the AnyConnect process. A successful exploit could allow the attacker to execute arbitrary code on the affected device with root privileges. To exploit this vulnerability, the attacker must have a valid account on the system.



CVE List▾ CNAs▾ WGs▾ Board▾ About▾ News & Blog▾

NVD
Go to for:
CVSS Scores
CPE Info

Search CVE List | Downloads | Data Feeds | Update a CVE Record | Request CVE IDs

TOTAL CVE Records: 223313

NOTICE: Transition to the all-new CVE website at WWW.CVE.ORG and CVE Record Format JSON are underway.

NOTICE: Legacy CVE download formats deprecation is now underway and will end on June 30, 2024.
New CVE List download format is available now.

HOME > CVE > CVE-2022-1729

Printer-Friendly View

CVE-ID	
CVE-2022-1729	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

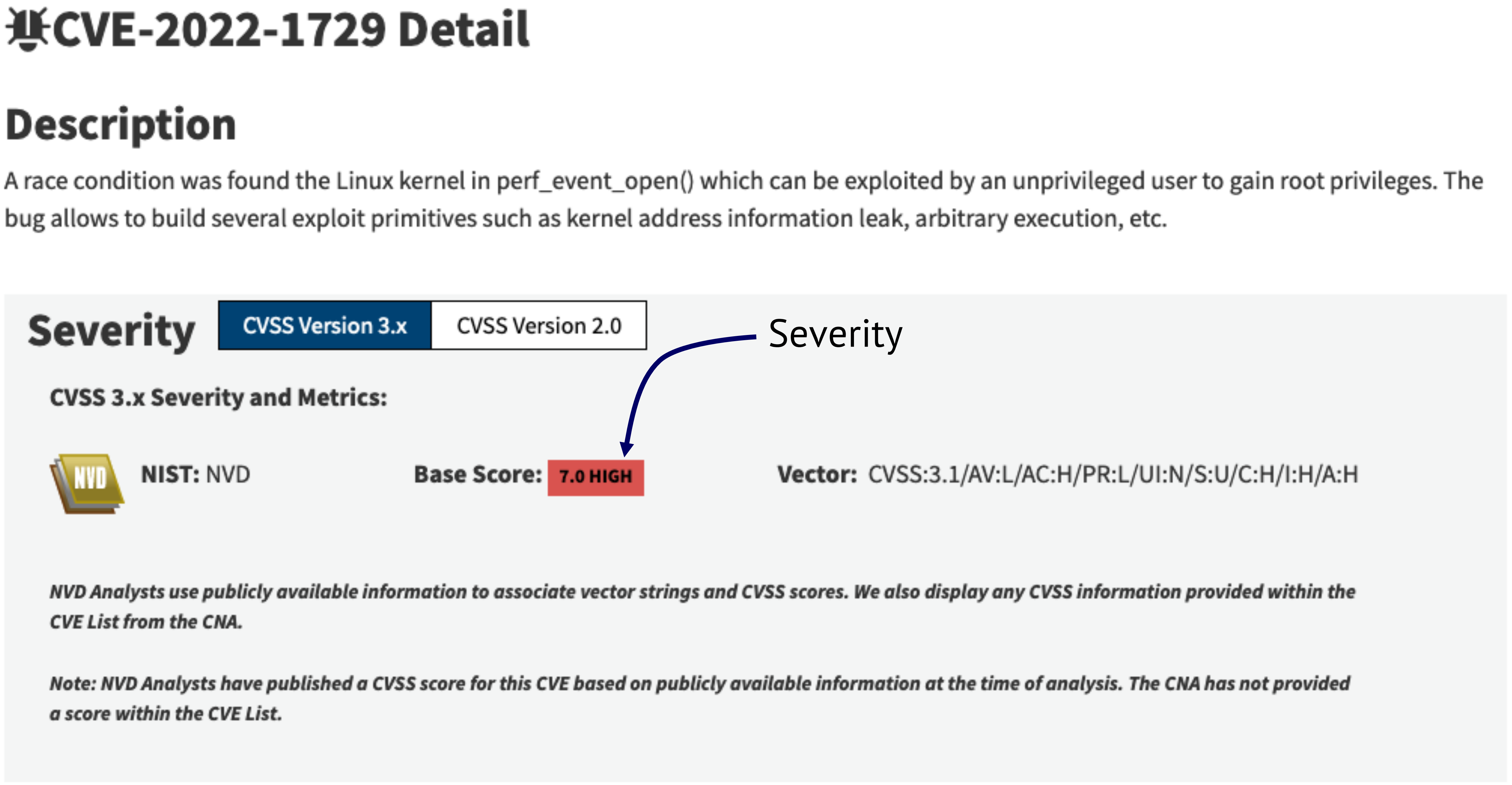
Description

A race condition was found the Linux kernel in perf_event_open() which can be exploited by an unprivileged user to gain root privileges. The bug allows to build several exploit primitives such as kernel address information leak, arbitrary execution, etc.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- CONFIRM:https://security.netapp.com/advisory/ntap-20230214-0006/
- MISC:https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=3ac6487e584a1eb54071dbe1212e05b884136704
- URL:https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=3ac6487e584a1eb54071dbe1212e05b884136704
- MISC:https://www.openwall.com/lists/oss-security/2022/05/20/2
- URL:https://www.openwall.com/lists/oss-security/2022/05/20/2



CVE-2022-1729 Detail

Description

A race condition was found the Linux kernel in perf_event_open() which can be exploited by an unprivileged user to gain root privileges. The bug allows to build several exploit primitives such as kernel address information leak, arbitrary execution, etc.

Severity

CVSS Version 3.x | CVSS Version 2.0

Severity

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** **7.0 HIGH** **Vector:** CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

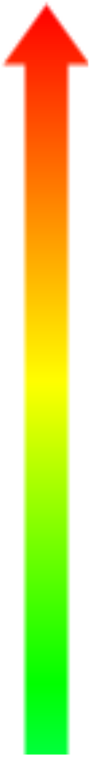
Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

Severity Score Metrics

1. Attack Vector (AV)
2. Attack Complexity (AC)
3. Privileges Required (PR)
4. User Interaction (UI)
5. Scope (S)
6. Confidentiality (C)
7. Integrity (I)
8. Availability (A)

Attack Vector (**AV**)

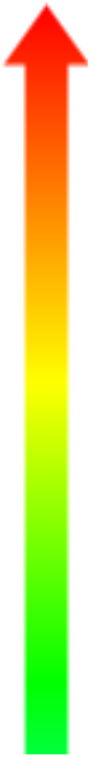
This metric reflects how the vulnerability is exploited



- **N** (Network): vulnerability can be exploited remotely through a network
- **A** (Adjacent): attack must be launched from the same shared physical or logical network
- **L** (Local): attack requires local access or physical access to the device
- **P** (Physical): attacker needs to physically touch or manipulate the vulnerable component

Attack Complexity (**AC**)

This metric describes the conditions beyond the attacker's control that must exist to exploit the vulnerability



- **L** (Low): Specialized access conditions or extenuating circumstances do not exist. The attacker can expect repeatable success

- **H** (High): A successful attack depends on conditions beyond the attacker's control, reducing the probability of a successful attack

Privileges Required (**PR**)

This metric describes the level of privileges an attacker must possess before successfully exploiting the vulnerability

- **N** (None): attacker does not require any privileges to exploit the vulnerability

- **L** (Low): attacker requires privileges that provide significant user capabilities (e.g., basic user privileges)

- **H** (High): attacker requires privileges that offer significant control over the system (e.g., admin privileges)

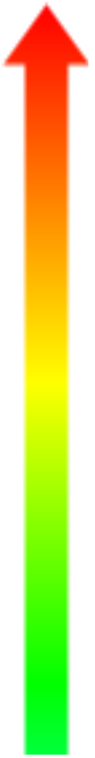
Privileges Required (**PR**)

This metric describes the level of privileges an attacker must possess before successfully exploiting the vulnerability

- **N** (None): attacker does not require any privileges to exploit the vulnerability
- **L** (Low): attacker requires privileges that provide significant user capabilities (e.g., basic user privileges)
- **H** (High): attacker requires privileges that offer significant control over the system (e.g., admin privileges)

User Interaction (**UI**)

This metric captures whether the vulnerability requires user interaction to be exploited

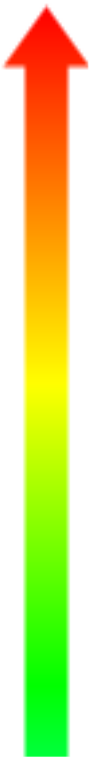


- **N** (None): vulnerability can be exploited without any user interaction

- **R** (Required): vulnerability requires a user to take some form of action

Scope (**S**)

This metric reflects whether a vulnerability in one software component impacts resources in components beyond its security scope

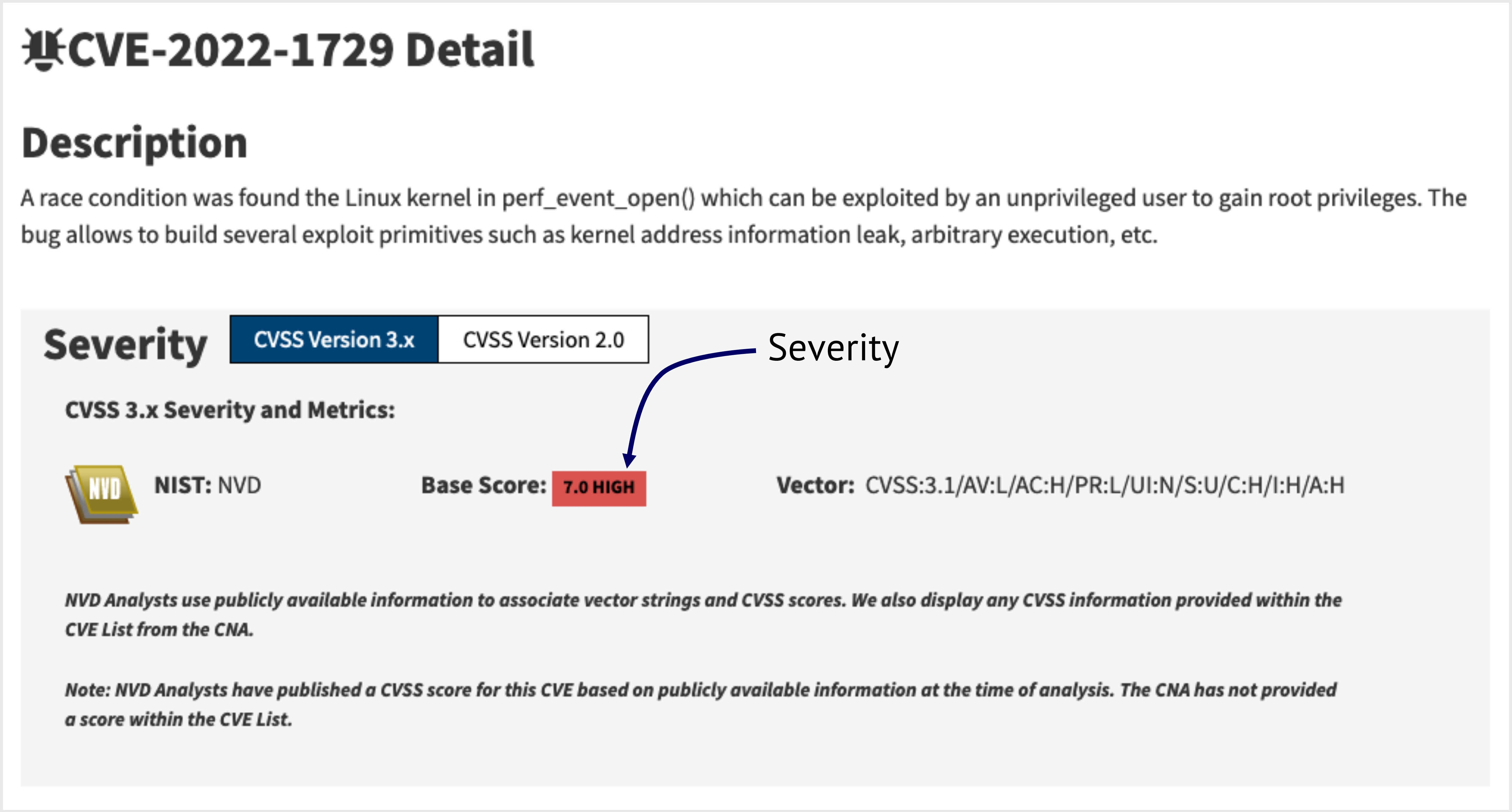


- **C** (Changed): exploited vulnerability can affect resources beyond its security scope (i.e., it impacts other components)
- **U** (Unchanged): exploited vulnerability affects the same software component

Confidentiality (C) Integrity (I) Availability(A)

These metrics measure the impact on confidentiality, integrity, and availability of the exploited system

- **H** (High): total loss of confidentiality, integrity, or availability
- **L** (Low): limited impact of confidentiality, integrity, or availability
- **N** (None): no impact on the confidentiality, integrity, or availability



CVE-2022-1729 Detail

Description

A race condition was found the Linux kernel in perf_event_open() which can be exploited by an unprivileged user to gain root privileges. The bug allows to build several exploit primitives such as kernel address information leak, arbitrary execution, etc.

Severity

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** **7.0 HIGH** **Vector:** CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

Software Vulnerabilities

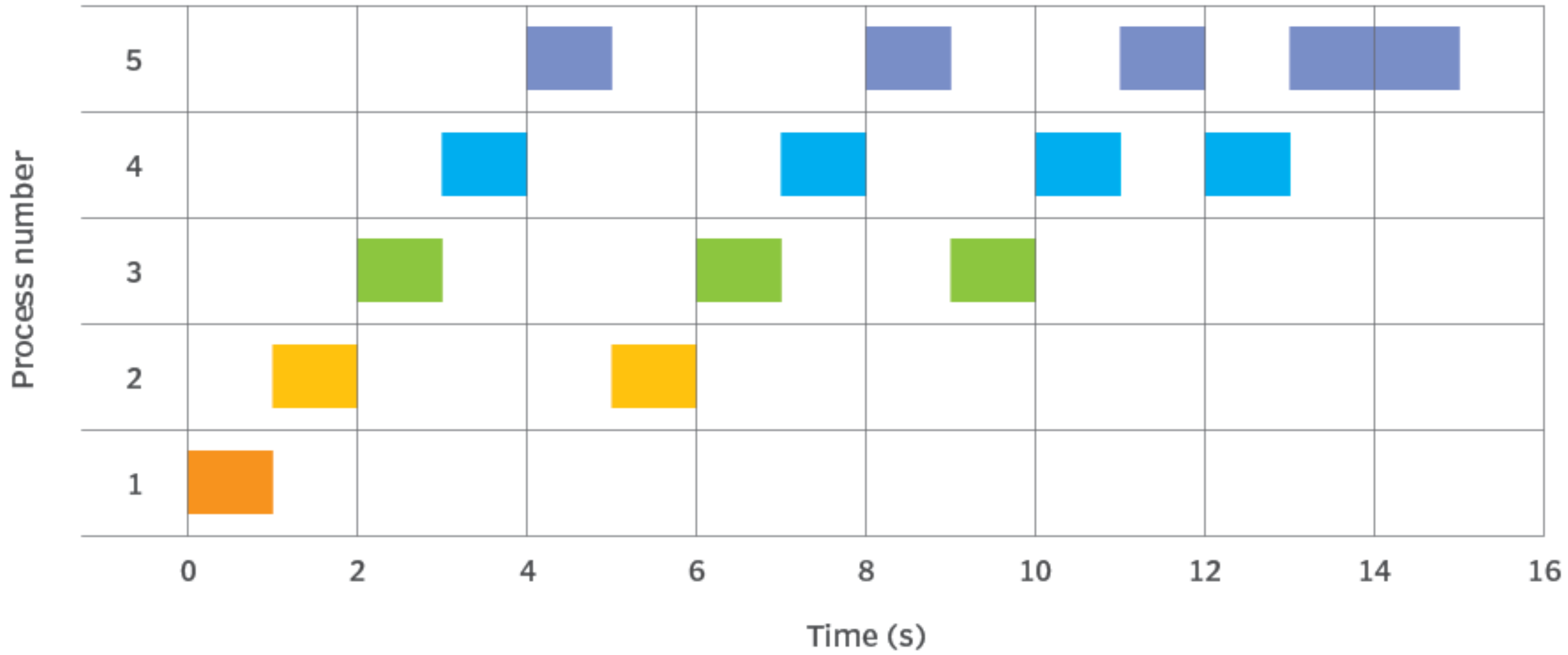
1. Common Vulnerabilities
2. Software Race Condition Vulnerabilities

What Are Race Conditions?

Race conditions occur when the outcome of a software process is dependent on the sequence or timing of uncontrollable events

They arise in concurrent systems where processes or threads operate in parallel

Round robin scheduling example



©2022 TECHTARGET. ALL RIGHTS RESERVED TechTarget

```
void withdraw(float amount)
{
    float accountBalance = getBalance();

    if (amount <= accountBalance)
    {
        // Calculate the new balance.
        accountBalance = accountBalance - amount;

        notifyCustomer("Successfull withdraw");

        // Update the account balance.
        updateBalance(accountBalance);

        // Give the money to the customer.
        dispenseCash(amount);
    }
    else
    {
        notifyCustomer("Insufficient funds in account");
    }
}
```

```
void withdraw(float amount)
{
    float accountBalance = getBalance();

    if (amount <= accountBalance)
    {
        // Calculate the new balance.
        accountBalance = accountBalance - amount;

        notifyCustomer("Successfull withdraw");

        // Update the account balance.
        updateBalance(accountBalance);

        // Give the money to the customer.
        dispenseCash(amount);
    }
    else
    {
        notifyCustomer("Insufficient funds in account");
    }
}
```

Balance: $100

ATM 1	ATM 2
$90	$90
Dispense $90 Cash	Dispense $90 Cash

New Balance: $10

```
void withdraw(float amount)
{
    float accountBalance = getBalance();

    if (amount <= accountBalance)
    {
        // Calculate the new balance.
        accountBalance = accountBalance - amount;

        notifyCustomer("Successfull withdraw");

        // Update the account balance.
        updateBalance(accountBalance);

        // Give the money to the customer.
        dispenseCash(amount);
    }
    else
    {
        notifyCustomer("Insufficient funds in account");
    }
}
```

Balance: $100

<u>ATM 1</u>	<u>ATM 2</u>
$90	$90