similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	P	693	Protection Mechanism Failure	1520

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Security Hardware *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Authentication Authorization	Gain Privileges or Assume Identity Execute Unauthorized Code or Commands Modify Memory	High

Detection Methods

Automated Dynamic Analysis

Automated testing can verify that RoT components are immutable.

Effectiveness = High

Architecture or Design Review

Root of trust elements and memory should be part of architecture and design reviews.

Effectiveness = High

Potential Mitigations

Phase: Architecture and Design

When architecting the system, the RoT should be designated for storage in a memory that does not allow further programming/writes.

Phase: Implementation

During implementation and test, the RoT memory location should be demonstrated to not allow further programming/writes.

Demonstrative Examples

Example 1:

The RoT is stored in memory. This memory can be modified by an adversary. For example, if an SoC implements "Secure Boot" by storing the boot code in an off-chip/on-chip flash, the contents of the flash can be modified by using a flash programmer. Similarly, if the boot code is stored in ROM (Read-Only Memory) but the public key or the hash of the public key (used to enable "Secure Boot") is stored in Flash or a memory that is susceptible to modifications or writes, the implementation is vulnerable.

In general, if the boot code, key materials and data that enable "Secure Boot" are all mutable, the implementation is vulnerable.

Good architecture defines RoT as immutable in hardware. One of the best ways to achieve immutability is to store boot code, public key or hash of the public key and other relevant data in Read-Only Memory (ROM) or One-Time Programmable (OTP) memory that prevents further programming or writes.

Example 2:

The example code below is a snippet from the bootrom of the HACK@DAC'19 buggy OpenPiton SoC [REF-1348]. The contents of the bootrom are critical in implementing the hardware root of trust.

It performs security-critical functions such as defining the system's device tree, validating the hardware cryptographic accelerators in the system, etc. Hence, write access to bootrom should be strictly limited to authorized users or removed completely so that bootrom is immutable. In this example (see the vulnerable code source), the boot instructions are stored in bootrom memory, mem. This memory can be read using the read address, addr_i, but write access should be restricted or removed.

Example Language: Verilog *(Bad)*

```
...
  always_ff @(posedge clk_i) begin
    if (req_i) begin
      if (!we_i) begin
        raddr_q <= addr_i[$clog2(RomSize)-1+3:3];
      end else begin
        mem[addr_i[$clog2(RomSize)-1+3:3]] <= wdata_i;
      end
    end
  end
...
// this prevents spurious Xes from propagating into the speculative fetch stage of the core
assign rdata_o = (raddr_q < RomSize) ? mem[raddr_q] : '0;
...
```

The vulnerable code shows an insecure implementation of the bootrom where bootrom can be written directly by enabling write enable, we_i, and using write address, addr_i, and write data, wdata_i.

To mitigate this issue, remove the write access to bootrom memory. [REF-1349]

Example Language: Verilog *(Good)*

```
...
  always_ff @(posedge clk_i) begin
    if (req_i) begin
      raddr_q <= addr_i[$clog2(RomSize)-1+3:3];
    end
  end
...
// this prevents spurious Xes from propagating into the speculative fetch stage of the core
assign rdata_o = (raddr_q < RomSize) ? mem[raddr_q] : '0;
...
```

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1196	Security Flow Issues	1194	2469
MemberOf	C	1413	Comprehensive Categorization: Protection Mechanism Failure	1400	2542

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
68	Subvert Code-signing Facilities
679	Exploitation of Improperly Configured or Implemented Memory Protections

References

[REF-1152]Trusted Computing Group. "TCG Roots of Trust Specification". 2018 July. < https://trustedcomputinggroup.org/wp-content/uploads/TCG_Roots_of_Trust_Specification_v0p20_PUBLIC_REVIEW.pdf >.

[REF-1153]GlobalPlatform Security Task Force. "Root of Trust Definitions and Requirements". 2017 March. < https://globalplatform.org/wp-content/uploads/2018/06/GP_RoT_Definitions_and_Requirements_v1.0.1_PublicRelease_CC.pdf >.

[REF-1348]"bootrom.sv". 2019. < https://github.com/HACK-EVENT/hackatdac19/blob/619e9fb0ef32ee1e01ad76b8732a156572c65700/bootrom/bootrom.sv#L263C19-L263C19 >.2023-09-18.

[REF-1349]"bootrom.sv". 2019. < https://github.com/HACK-EVENT/hackatdac19/blob/ba6abf58586b2bf4401e9f4d46e3f084c664ff88/bootrom/bootrom.sv#L259C9-L259C9 >.2023-09-18.

CWE-1327: Binding to an Unrestricted IP Address

Weakness ID : 1327
Structure : Simple
Abstraction : Base

Description

The product assigns the address 0.0.0.0 for a database server, a cloud service/instance, or any computing resource that communicates remotely.

Extended Description

When a server binds to the address 0.0.0.0, it allows connections from every IP address on the local machine, effectively exposing the server to every possible network. This might be much broader access than intended by the developer or administrator, who might only be expecting the server to be reachable from a single interface/network.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf		668	Exposure of Resource to Wrong Sphere	1469

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name	Page
MemberOf	C	417	Communication Channel Errors	2325

Applicable Platforms

Language : Other *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Web Server *(Prevalence = Undetermined)*

Technology : Client Server *(Prevalence = Undetermined)*

Technology : Cloud Computing *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Availability	DoS: Amplification	High

Potential Mitigations

Phase: System Configuration

Assign IP addresses that are not 0.0.0.0.

Effectiveness = High

Phase: System Configuration

Strategy = Firewall

Unwanted connections to the configured server may be denied through a firewall or other packet filtering measures.

Effectiveness = High

Demonstrative Examples

Example 1:

The following code snippet uses 0.0.0.0 in a Puppet script.

Example Language: Other *(Bad)*

```
signingserver::instance {
  "nightly-key-signing-server":
    listenaddr => "0.0.0.0",
    port => "9100",
    code_tag => "SIGNING_SERVER",
  }
```

The Puppet code snippet is used to provision a signing server that will use 0.0.0.0 to accept traffic. However, as 0.0.0.0 is unrestricted, malicious users may use this IP address to launch frequent requests and cause denial of service attacks.

Example Language: Other *(Good)*

```
signingserver::instance {
  "nightly-key-signing-server":
    listenaddr => "127.0.0.1",
    port => "9100",
    code_tag => "SIGNING_SERVER",
  }
```

Observed Examples

Reference	Description
CVE-2022-21947	Desktop manager for Kubernetes and container management binds a service to 0.0.0.0, allowing users on the network to make requests to a dashboard API. *https://www.cve.org/CVERecord?id=CVE-2022-21947*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1403	Comprehensive Categorization: Exposed Resource	1400	2528

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
1	Accessing Functionality Not Properly Constrained by ACLs

References

[REF-1158]Akond Rahman, Md Rayhanur Rahman, Chris Parnin and Laurie Williams. "Security Smells in Ansible and Chef Scripts: A Replication Study". 2020 June 0. < https://arxiv.org/pdf/1907.07159.pdf >.

[REF-1159]Akond Rahman, Chris Parnin and Laurie Williams. "The Seven Sins: Security Smells in Infrastructure as Code Scripts". ICSE '19: Proceedings of the 41st International Conference on Software Engineering. 2019 May. < https://dl.acm.org/doi/10.1109/ICSE.2019.00033 >.2023-04-07.

CWE-1328: Security Version Number Mutable to Older Versions

Weakness ID : 1328
Structure : Simple
Abstraction : Base

Description

Security-version number in hardware is mutable, resulting in the ability to downgrade (roll-back) the boot firmware to vulnerable code versions.

Extended Description

A System-on-Chip (SoC) implements secure boot or verified boot. It might support a security version number, which prevents downgrading the current firmware to a vulnerable version. Once downgraded to a previous version, an adversary can launch exploits on the SoC and thus compromise the security of the SoC. These downgrade attacks are also referred to as roll-back attacks.

The security version number must be stored securely and persistently across power-on resets. A common weakness is that the security version number is modifiable by an adversary, allowing roll-back or downgrade attacks or, under certain circumstances, preventing upgrades (i.e. Denial-of-Service on upgrades). In both cases, the SoC is in a vulnerable state.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	285	Improper Authorization	684
PeerOf	B	757	Selection of Less-Secure Algorithm During Negotiation ('Algorithm Downgrade')	1581

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Security Hardware *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality Integrity Authentication Authorization	Other *Impact includes roll-back or downgrade to a vulnerable version of the firmware or DoS (prevent upgrades).*	High

Detection Methods

Automated Dynamic Analysis

Mutability of stored security version numbers and programming with older firmware images should be part of automated testing.

Effectiveness = High

Architecture or Design Review

Anti-roll-back features should be reviewed as part of Architecture or Design review.

Effectiveness = High

Potential Mitigations

Phase: Architecture and Design

When architecting the system, security version data should be designated for storage in registers that are either read-only or have access controls that prevent modification by an untrusted agent.

Phase: Implementation

During implementation and test, security version data should be demonstrated to be read-only and access controls should be validated.

Demonstrative Examples

Example 1:

A new version of firmware is signed with a security version number higher than the previous version. During the firmware update process the SoC checks for the security version number and upgrades the SoC firmware with the latest version. This security version number is stored in persistent memory upon successful upgrade for use across power-on resets.

In general, if the security version number is mutable, the implementation is vulnerable. A mutable security version number allows an adversary to change the security version to a lower value to allow roll-back or to a higher value to prevent future upgrades.

The security version number should be stored in immutable hardware such as fuses, and the writes to these fuses should be highly access-controlled with appropriate authentication and authorization protections.

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1196	Security Flow Issues	1194	2469
MemberOf	C	1396	Comprehensive Categorization: Access Control	1400	2519

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
176	Configuration/Environment Manipulation

CWE-1329: Reliance on Component That is Not Updateable

Weakness ID : 1329
Structure : Simple
Abstraction : Base

Description

The product contains a component that cannot be updated or patched in order to remove vulnerabilities or significant bugs.

Extended Description

If the component is discovered to contain a vulnerability or critical bug, but the issue cannot be fixed using an update or patch, then the product's owner will not be able to protect against the issue. The only option might be replacement of the product, which could be too financially or operationally expensive for the product owner. As a result, the inability to patch or update can leave the product open to attacker exploitation or critical operation failures. This weakness can be especially difficult to manage when using ROM, firmware, or similar components that traditionally have had limited or no update capabilities.

In industries such as healthcare, "legacy" devices can be operated for decades. As a US task force report [REF-1197] notes, "the inability to update or replace equipment has both large and small health care delivery organizations struggle with numerous unsupported legacy systems that cannot easily be replaced (hardware, software, and operating systems) with large numbers of vulnerabilities and few modern countermeasures."

While hardware can be prone to this weakness, software systems can also be affected, such as when a third-party driver or library is no longer actively maintained or supported but is still critical for the required functionality.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	P	664	Improper Control of a Resource Through its Lifetime	1454
ChildOf	C	1357	Reliance on Insufficiently Trustworthy Component	2254
ParentOf	B	1277	Firmware Not Updateable	2116
ParentOf	B	1310	Missing Ability to Patch ROM Code	2179

Weakness Ordinalities

Primary :

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Technology : ICS/OT *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality Integrity Access Control Authentication Authorization Other	Gain Privileges or Assume Identity Bypass Protection Mechanism Execute Unauthorized Code or Commands DoS: Crash, Exit, or Restart Quality Degradation Reduce Maintainability *If an attacker can identify an exploitable vulnerability in one product that has no means of patching, the attack may be used against all affected versions of that product.*	

Detection Methods

Architecture or Design Review

Check the consumer or maintainer documentation, the architecture/design documentation, or the original requirements to ensure that the documentation includes details for how to update the firmware.

Effectiveness = Moderate

Potential Mitigations

Phase: Requirements

Specify requirements that each component should be updateable, including ROM, firmware, etc.

Phase: Architecture and Design

Design the product to allow for updating of its components. Include the external infrastructure that might be necessary to support updates, such as distribution servers.

Phase: Architecture and Design

Phase: Implementation

With hardware, support patches that can be programmed in-field or during manufacturing through hardware fuses. This feature can be used for limited patching of devices after shipping, or for the next batch of silicon devices manufactured, without changing the full device ROM.

Effectiveness = Moderate

Some parts of the hardware initialization or signature verification done to authenticate patches will always be "not patchable." Hardware-fuse-based patches will also have limitations in terms of size and the number of patches that can be supported.

Phase: Implementation

Implement the necessary functionality to allow each component to be updated.

Demonstrative Examples

Example 1:

A refrigerator has an Internet interface for the official purpose of alerting the manufacturer when that refrigerator detects a fault. Because the device is attached to the Internet, the refrigerator is a target for hackers who may wish to use the device other potentially more nefarious purposes.

Example Language: Other *(Bad)*

```
The refrigerator has no means of patching and is hacked becoming a spewer of email spam.
```

Example Language: Other *(Good)*

The device automatically patches itself and provides considerable more protection against being hacked.

Example 2:

A System-on-Chip (SOC) implements a Root-of-Trust (RoT) in ROM to boot secure code. However, at times this ROM code might have security vulnerabilities and need to be patched. Since ROM is immutable, it can be impossible to patch.

ROM does not have built-in application-programming interfaces (APIs) to patch if the code is vulnerable. Implement mechanisms to patch the vulnerable ROM code.

Example 3:

The example code is taken from the JTAG module of the buggy OpenPiton SoC of HACK@DAC'21. JTAG is protected with a password checker. Access to JTAG operations will be denied unless the correct password is provided by the user. This user-provided password is first sent to the HMAC module where it is hashed with a secret crypto key. This user password hash (pass_hash) is then compared with the hash of the correct password (exp_hash). If they match, JTAG will then be unlocked.

Example Language: Verilog *(Bad)*

```
module dmi_jtag(...)(...);
...
      PassChkValid: begin
      if(hashValid) begin
          if(exp_hash == pass_hash) begin
            pass_check = 1'b1;
          end else begin
            pass_check = 1'b0;
          end
          state_d = Idle;
        end else begin
        state_d = PassChkValid;
        end
      end
...
  hmac hmac(
...
    .key_i(256'h24e6fa2254c2ff632a41b...),
...
  );
...
endmodule
```

However, the SoC's crypto key is hardcoded into the design and cannot be updated [REF-1387]. Therefore, if the key is leaked somehow, there is no way to reprovision the key without having the device replaced.

To fix this issue, a local register should be used (hmac_key_reg) to store the crypto key. If designers need to update the key, they can upload the new key through an input port (hmac_key_i) to the local register by enabling the patching signal (hmac_patch_en) [REF-1388].

Example Language: Verilog *(Good)*

```
module dmi_jtag(...
) (
  input logic [255:0] hmac_key_i,
  input logic hmac_patch_en,
  ...
  reg [255:0] hmac_key_reg;
  ...
```

```
);
...
   always_ff @(posedge tck_i or negedge trst_ni) begin
   ...
   if (hmac_patch_en)
      hmac_key_reg <= hmac_key_i;
   ...
   end
...
   hmac hmac(
   ...
   .key_i(hmac_key_reg),
   ...
   );
...
endmodule
```

Observed Examples

Reference	Description
CVE-2020-9054	Chain: network-attached storage (NAS) device has a critical OS command injection (CWE-78) vulnerability that is actively exploited to place IoT devices into a botnet, but some products are "end-of-support" and cannot be patched (CWE-1277). [REF-1097] *https://www.cve.org/CVERecord?id=CVE-2020-9054*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1208	Cross-Cutting Problems	1194	2474
MemberOf	C	1368	ICS Dependencies (& Architecture): External Digital Systems	1358	2505
MemberOf	C	1415	Comprehensive Categorization: Resource Control	1400	2544

References

[REF-1197]Health Care Industry Cybersecurity Task Force. "Report on Improving Cybersecurity in the Health Care Industry". 2017 June. < https://www.phe.gov/Preparedness/planning/CyberTF/Documents/report2017.pdf >.

[REF-1097]Brian Krebs. "Zyxel Flaw Powers New Mirai IoT Botnet Strain". 2020 March 0. < https://krebsonsecurity.com/2020/03/zxyel-flaw-powers-new-mirai-iot-botnet-strain/ >.

[REF-1387]"dmi_jtag.sv line 324". 2021. < https://github.com/HACK-EVENT/hackatdac21/blob/main/piton/design/chip/tile/ariane/src/riscv-dbg/src/dmi_jtag.sv#L324C9-L324C87 >.2024-01-16.

[REF-1388]"Fix for dmi_jtag.sv". 2021. < https://github.com/HACK-EVENT/hackatdac21/commit/c94ce5f9487a41c77ede0bbc8daf4da66c39f42a >.2024-01-16.

CWE-1330: Remanent Data Readable after Memory Erase

Weakness ID : 1330
Structure : Simple
Abstraction : Variant

Description

Confidential information stored in memory circuits is readable or recoverable after being cleared or erased.

Extended Description

Data remanence occurs when stored, memory content is not fully lost after a memory-clear or -erase operation. Confidential memory contents can still be readable through data remanence in the hardware.

Data remanence can occur because of performance optimization or memory organization during 'clear' or 'erase' operations, like a design that allows the memory-organization metadata (e.g., file pointers) to be erased without erasing the actual memory content. To protect against this weakness, memory devices will often support different commands for optimized memory erase and explicit secure erase.

Data remanence can also happen because of the physical properties of memory circuits in use. For example, static, random-access-memory (SRAM) and dynamic, random-access-memory (DRAM) data retention is based on the charge retained in the memory cell, which depends on factors such as power supply, refresh rates, and temperature.

Other than explicit erase commands, self-encrypting, secure-memory devices can also support secure erase through cryptographic erase commands. In such designs, only the decryption keys for encrypted data stored on the device are erased. That is, the stored data are always remnant in the media after a cryptographic erase. However, only the encrypted data can be extracted. Thus, protection against data recovery in such designs relies on the strength of the encryption algorithm.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	B	1301	Insufficient or Incomplete Data Removal within Hardware Component	2170

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name	Page
ChildOf	B	1301	Insufficient or Incomplete Data Removal within Hardware Component	2170

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Security Hardware *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality	Modify Memory Read Memory *Confidential data are readable to untrusted agent.*	

Detection Methods

Architecture or Design Review

Testing of memory-device contents after clearing or erase commands. Dynamic analysis of memory contents during device operation to detect specific, confidential assets. Architecture and design analysis of memory clear and erase operations.

Dynamic Analysis with Manual Results Interpretation

Testing of memory-device contents after clearing or erase commands. Dynamic analysis of memory contents during device operation to detect specific, confidential assets. Architecture and design analysis of memory clear and erase operations.

Potential Mitigations

Phase: Architecture and Design

Support for secure-erase commands that apply multiple cycles of overwriting memory with known patterns and of erasing actual content. Support for cryptographic erase in self-encrypting, memory devices. External, physical tools to erase memory such as ultraviolet-rays-based erase of Electrically erasable, programmable, read-only memory (EEPROM). Physical destruction of media device. This is done for repurposed or scrapped devices that are no longer in use.

Demonstrative Examples

Example 1:

Consider a device that uses flash memory for non-volatile-data storage. To optimize flash-access performance or reliable-flash lifetime, the device might limit the number of flash writes/erases by maintaining some state in internal SRAM and only committing changes to flash memory periodically.

The device also supports user reset to factory defaults with the expectation that all personal information is erased from the device after this operation. On factory reset, user files are erased using explicit, erase commands supported by the flash device.

In the given, system design, the flash-file system can support performance-optimized erase such that only the file metadata are erased and not the content. If this optimized erase is used for files containing user data during factory-reset flow, then device, flash memory can contain remanent data from these files.

On device-factory reset, the implementation might not erase these copies, since the file organization has changed and the flash file system does not have the metadata to track all previous copies.

A flash-memory region that is used by a flash-file system should be fully erased as part of the factory-reset flow. This should include secure-erase flow for the flash media such as overwriting patterns multiple times followed by erase.

Observed Examples

Reference	Description
CVE-2019-8575	Firmware Data Deletion Vulnerability in which a base station factory reset might not delete all user information. The impact of this enables a new owner of a used device that has been "factory-default reset" with a vulnerable firmware version can still retrieve, at least, the previous owner's wireless network name, and the previous owner's wireless security (such as WPA2) key. This issue was addressed with improved, data deletion. *https://www.cve.org/CVERecord?id=CVE-2019-8575*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1416	Comprehensive Categorization: Resource Lifecycle Management	1400	2545

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
37	Retrieve Embedded Sensitive Data
150	Collect Data from Common Resource Locations
545	Pull Data from System Resources

References

[REF-1154]National Institute of Standards and Technology. "NIST Special Publication 800-88 Revision 1: Guidelines for Media Sanitization". 2014 December. < https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-88r1.pdf >.2023-04-07.

CWE-1331: Improper Isolation of Shared Resources in Network On Chip (NoC)

Weakness ID : 1331
Structure : Simple
Abstraction : Base

Description

The Network On Chip (NoC) does not isolate or incorrectly isolates its on-chip-fabric and internal resources such that they are shared between trusted and untrusted agents, creating timing channels.

Extended Description

Typically, network on chips (NoC) have many internal resources that are shared between packets from different trust domains. These resources include internal buffers, crossbars and switches, individual ports, and channels. The sharing of resources causes contention and introduces interference between differently trusted domains, which poses a security threat via a timing channel, allowing attackers to infer data that belongs to a trusted agent. This may also result in introducing network interference, resulting in degraded throughput and latency.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	668	Exposure of Resource to Wrong Sphere	1469
ChildOf	C	653	Improper Isolation or Compartmentalization	1437
PeerOf	B	1189	Improper Isolation of Shared Resources on System-on-a-Chip (SoC)	1976

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name	Page
PeerOf	B	1189	Improper Isolation of Shared Resources on System-on-a-Chip (SoC)	1976

Weakness Ordinalities

Primary :

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Security Hardware *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Background Details

"Network-on-chip" (NoC) is a commonly-used term used for hardware interconnect fabrics used by multicore Systems-on-Chip (SoC). Communication between modules on the chip uses packet-based methods, with improved efficiency and scalability compared to bus architectures [REF-1241].

Common Consequences

Scope	Impact	Likelihood
Confidentiality Availability	DoS: Resource Consumption (Other) Varies by Context Other *Attackers may infer data that belongs to a trusted agent. The methods used to perform this attack may result in noticeably increased resource consumption.*	Medium

Detection Methods

Manual Analysis

Providing marker flags to send through the interfaces coupled with examination of which users are able to read or manipulate the flags will help verify that the proper isolation has been achieved and is effective.

Effectiveness = Moderate

Potential Mitigations

Phase: Architecture and Design

Phase: Implementation

Implement priority-based arbitration inside the NoC and have dedicated buffers or virtual channels for routing secret data from trusted agents.

Demonstrative Examples

Example 1:

Consider a NoC that implements a one-dimensional mesh network with four nodes. This supports two flows: Flow A from node 0 to node 3 (via node 1 and node 2) and Flow B from node 1 to node 2. Flows A and B share a common link between Node 1 and Node 2. Only one flow can use the link in each cycle.

One of the masters to this NoC implements a cryptographic algorithm (RSA), and another master to the NoC is a core that can be exercised by an attacker. The RSA algorithm performs a modulo multiplication of two large numbers and depends on each bit of the secret key. The algorithm examines each bit in the secret key and only performs multiplication if the bit is 1. This algorithm is known to be prone to timing attacks. Whenever RSA performs multiplication, there is additional network traffic to the memory controller. One of the reasons for this is cache conflicts.

Since this is a one-dimensional mesh, only one flow can use the link in each cycle. Also, packets from the attack program and the RSA program share the output port of the network-on-chip. This contention results in network interference, and the throughput and latency of one flow can be affected by the other flow's demand.

Example Language: *(Attack)*

The attacker runs a loop program on the core they control, and this causes a cache miss in every iteration for the RSA algorithm. Thus, by observing network-traffic bandwidth and timing, the attack program can determine when the RSA algorithm is doing a multiply operation (i.e., when the secret key bit is 1) and eventually extract the entire, secret key.

There may be different ways to fix this particular weakness.

Example Language: Other *(Good)*

Implement priority-based arbitration inside the NoC and have dedicated buffers or virtual channels for routing secret data from trusted agents.

Observed Examples

Reference	Description
CVE-2021-33096	Improper isolation of shared resource in a network-on-chip leads to denial of service *https://www.cve.org/CVERecord?id=CVE-2021-33096*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1203	Peripherals, On-chip Fabric, and Interface/IO Problems	1194	2472
MemberOf	C	1418	Comprehensive Categorization: Violation of Secure Design Principles	1400	2549

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
124	Shared Resource Manipulation

References

[REF-1155]Hassan M. G. Wassel, Ying Gao, Jason K. Oberg, Tedd Huffmire, Ryan Kastner, Frederic T. Chong, Timothy Sherwood. "SurfNoC: A Low Latency and Provably Non-Interfering Approach to Secure Networks-On-Chip". 2013. < http://cseweb.ucsd.edu/~kastner/papers/isca13-surfNOC.pdf >.

[REF-1241]Wikipedia. "Network on a chip". < https://en.wikipedia.org/wiki/Network_on_a_chip >.2021-10-24.

[REF-1242]Subodha Charles and Prabhat Mishra. "A Survey of Network-on-Chip Security Attacks and Countermeasures". ACM Computing Surveys. 2021 May. < https://dl.acm.org/doi/fullHtml/10.1145/3450964 >.2023-04-07.

[REF-1245]Subodha Charles. "Design of Secure and Trustworthy Network-on-chip Architectures". 2020. < https://www.cise.ufl.edu/research/cad/Publications/charlesThesis.pdf >.

CWE-1332: Improper Handling of Faults that Lead to Instruction Skips

Weakness ID : 1332
Structure : Simple
Abstraction : Base

Description

The device is missing or incorrectly implements circuitry or sensors that detect and mitigate the skipping of security-critical CPU instructions when they occur.

Extended Description

The operating conditions of hardware may change in ways that cause unexpected behavior to occur, including the skipping of security-critical CPU instructions. Generally, this can occur due to electrical disturbances or when the device operates outside of its expected conditions.

In practice, application code may contain conditional branches that are security-sensitive (e.g., accepting or rejecting a user-provided password). These conditional branches are typically implemented by a single conditional branch instruction in the program binary which, if skipped, may lead to effectively flipping the branch condition - i.e., causing the wrong security-sensitive branch to be taken. This affects processes such as firmware authentication, password verification, and other security-sensitive decision points.

Attackers can use fault injection techniques to alter the operating conditions of hardware so that security-critical instructions are skipped more frequently or more reliably than they would in a "natural" setting.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	1384	Improper Handling of Physical or Environmental Conditions	2257

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name	Page
PeerOf	B	1247	Improper Protection Against Voltage and Clock Glitches	2044

Weakness Ordinalities

Primary :

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : System on Chip *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality Integrity Authentication	Bypass Protection Mechanism Alter Execution Logic Unexpected State *Depending on the context, instruction skipping can have a broad range of consequences related to the generic bypassing of security critical code.*	High

Detection Methods

Automated Static Analysis

This weakness can be found using automated static analysis once a developer has indicated which code paths are critical to protect.

Effectiveness = Moderate

Simulation / Emulation

This weakness can be found using automated dynamic analysis. Both emulation of a CPU with instruction skips, as well as RTL simulation of a CPU IP, can indicate parts of the code that are sensitive to faults due to instruction skips.

Effectiveness = Moderate

Manual Analysis

This weakness can be found using manual (static) analysis. The analyst has security objectives that are matched against the high-level code. This method is less precise than emulation, especially if the analysis is done at the higher level language rather than at assembly level.

Effectiveness = Moderate

Potential Mitigations

Phase: Architecture and Design

Design strategies for ensuring safe failure if inputs, such as Vcc, are modified out of acceptable ranges.

Phase: Architecture and Design

Design strategies for ensuring safe behavior if instructions attempt to be skipped.

Phase: Architecture and Design

Identify mission critical secrets that should be wiped if faulting is detected, and design a mechanism to do the deletion.

Phase: Implementation

Add redundancy by performing an operation multiple times, either in space or time, and perform majority voting. Additionally, make conditional instruction timing unpredictable.

Phase: Implementation

Use redundant operations or canaries to detect and respond to faults.

Phase: Implementation

Ensure that fault mitigations are strong enough in practice. For example, a low power detection mechanism that takes 50 clock cycles to trigger at lower voltages may be an insufficient security mechanism if the instruction counter has already progressed with no other CPU activity occurring.

Demonstrative Examples

Example 1:

A smart card contains authentication credentials that are used as authorization to enter a building. The credentials are only accessible when a correct PIN is presented to the card.

Example Language: *(Bad)*

The card emits the credentials when a voltage anomaly is injected into the power line to the device at a particular time after providing an incorrect PIN to the card, causing the internal program to accept the incorrect PIN.

There are several ways this weakness could be fixed.

Example Language: *(Good)*

- add an internal filter or internal power supply in series with the power supply pin on the device
- add sensing circuitry to reset the device if out of tolerance conditions are detected

- add additional execution sensing circuits to monitor the execution order for anomalies and abort the action or reset the device under fault conditions

Observed Examples

Reference	Description
CVE-2019-15894	fault injection attack bypasses the verification mode, potentially allowing arbitrary code execution. *https://www.cve.org/CVERecord?id=CVE-2019-15894*

Functional Areas

- Power

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1206	Power, Clock, Thermal, and Reset Concerns	1194	2473
MemberOf	C	1365	ICS Communications: Unreliability	1358	2502
MemberOf	C	1388	Physical Access Issues and Concerns	1194	2518
MemberOf	C	1405	Comprehensive Categorization: Improper Check or Handling of Exceptional Conditions	1400	2531

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
624	Hardware Fault Injection
625	Mobile Device Fault Injection

References

[REF-1161]Josep Balasch, Benedikt Gierlichs and Ingrid Verbauwhede. "An In-depth and Black-box Characterization of the Effects of Clock Glitches on 8-bit MCUs". 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography (IEEE). 2011 September. < https://ieeexplore.ieee.org/document/6076473 >.

[REF-1222]Alexandre Menu, Jean-Max Dutertre, Olivier Potin and Jean-Baptiste Rigaud. "Experimental Analysis of the Electromagnetic Instruction Skip Fault Model". IEEE Xplore. 2020 April 0. < https://ieeexplore.ieee.org/document/9081261 >.

[REF-1223]Niek Timmers, Albert Spruyt and Marc Witteman. "Controlling PC on ARM using Fault Injection". 2016 June 1. < https://fdtc.deib.polimi.it/FDTC16/shared/FDTC-2016-session_2_1.pdf >.2023-04-07.

[REF-1224]Colin O'Flynn. "Attacking USB Gear with EMFI". Circuit Cellar. 2019 May. < https://www.totalphase.com/media/pdf/whitepapers/Circuit_Cellar_TP.pdf >.

[REF-1286]Lennert Wouters, Benedikt Gierlichs and Bart Preneel. "On The Susceptibility of Texas Instruments SimpleLink Platform Microcontrollers to Non-Invasive Physical Attacks". 2022 March 4. < https://eprint.iacr.org/2022/328.pdf >.

CWE-1333: Inefficient Regular Expression Complexity

Weakness ID : 1333
Structure : Simple
Abstraction : Base

Description

The product uses a regular expression with an inefficient, possibly exponential worst-case computational complexity that consumes excessive CPU cycles.

Extended Description

Some regular expression engines have a feature called "backtracking". If the token cannot match, the engine "backtracks" to a position that may result in a different token that can match. Backtracking becomes a weakness if all of these conditions are met:

- The number of possible backtracking attempts are exponential relative to the length of the input.
- The input can fail to match the regular expression.
- The input can be long enough.

Attackers can create crafted inputs that intentionally cause the regular expression to use excessive backtracking in a way that causes the CPU consumption to spike.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	407	Inefficient Algorithmic Complexity	992

Relevant to the view "Weaknesses for Simplified Mapping of Published Vulnerabilities" (CWE-1003)

Nature	Type	ID	Name	Page
ChildOf	C	407	Inefficient Algorithmic Complexity	992

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name	Page
MemberOf	C	1226	Complexity Issues	2481

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Alternate Terms

ReDoS : ReDoS is an abbreviation of "Regular expression Denial of Service".

Regular Expression Denial of Service : While this term is attack-focused, this is commonly used to describe the weakness.

Catastrophic backtracking : This term is used to describe the behavior of the regular expression as a negative technical impact.

Likelihood Of Exploit

High

Common Consequences

Scope	Impact	Likelihood
Availability	DoS: Resource Consumption (CPU)	High

Potential Mitigations

Phase: Architecture and Design

Use regular expressions that do not support backtracking, e.g. by removing nested quantifiers.

Effectiveness = High

This is one of the few effective solutions when using user-provided regular expressions.

Phase: System Configuration

Set backtracking limits in the configuration of the regular expression implementation, such as PHP's pcre.backtrack_limit. Also consider limits on execution time for the process.

Effectiveness = Moderate

Phase: Implementation

Do not use regular expressions with untrusted input. If regular expressions must be used, avoid using backtracking in the expression.

Effectiveness = High

Phase: Implementation

Limit the length of the input that the regular expression will process.

Effectiveness = Moderate

Demonstrative Examples

Example 1:

This example attempts to check if an input string is a "sentence" [REF-1164].

Example Language: JavaScript *(Bad)*

```
var test_string = "Bad characters: $@#";
var bad_pattern = /^(\w+\s?)*$/i;
var result = test_string.search(bad_pattern);
```

The regular expression has a vulnerable backtracking clause inside (\w+\s?)*$ which can be triggered to cause a Denial of Service by processing particular phrases.

To fix the backtracking problem, backtracking is removed with the ?= portion of the expression which changes it to a lookahead and the \2 which prevents the backtracking. The modified example is:

Example Language: JavaScript *(Good)*

```
var test_string = "Bad characters: $@#";
var good_pattern = /^((?=(\w+))\2\s?)*$/i;
var result = test_string.search(good_pattern);
```

Note that [REF-1164] has a more thorough (and lengthy) explanation of everything going on within the RegEx.

Example 2:

This example attempts to check if an input string is a "sentence" and is modified for Perl [REF-1164].

Example Language: Perl *(Bad)*

```
my $test_string = "Bad characters: \$\@\#";
my $bdrslt = $test_string;
$bdrslt =~ /^(\w+\s?)*$/i;
```

The regular expression has a vulnerable backtracking clause inside (\w+\s?)*$ which can be triggered to cause a Denial of Service by processing particular phrases.

To fix the backtracking problem, backtracking is removed with the ?= portion of the expression which changes it to a lookahead and the \2 which prevents the backtracking. The modified example is:

Example Language: Perl *(Good)*

```
my $test_string = "Bad characters: \$\@\#";
my $gdrslt = $test_string;
$gdrslt =~ /^((?=(\w+))\2\s?)*$/i;
```

Note that [REF-1164] has a more thorough (and lengthy) explanation of everything going on within the RegEx.

Observed Examples

Reference	Description
CVE-2020-5243	server allows ReDOS with crafted User-Agent strings, due to overlapping capture groups that cause excessive backtracking. *https://www.cve.org/CVERecord?id=CVE-2020-5243*
CVE-2021-21317	npm package for user-agent parser prone to ReDoS due to overlapping capture groups *https://www.cve.org/CVERecord?id=CVE-2021-21317*
CVE-2019-16215	Markdown parser uses inefficient regex when processing a message, allowing users to cause CPU consumption and delay preventing processing of other messages. *https://www.cve.org/CVERecord?id=CVE-2019-16215*
CVE-2019-6785	Long string in a version control product allows DoS due to an inefficient regex. *https://www.cve.org/CVERecord?id=CVE-2019-6785*
CVE-2019-12041	Javascript code allows ReDoS via a long string due to excessive backtracking. *https://www.cve.org/CVERecord?id=CVE-2019-12041*
CVE-2015-8315	ReDoS when parsing time. *https://www.cve.org/CVERecord?id=CVE-2015-8315*
CVE-2015-8854	ReDoS when parsing documents. *https://www.cve.org/CVERecord?id=CVE-2015-8854*
CVE-2017-16021	ReDoS when validating URL. *https://www.cve.org/CVERecord?id=CVE-2017-16021*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1416	Comprehensive Categorization: Resource Lifecycle Management	1400	2545

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
492	Regular Expression Exponential Blowup

References

[REF-1180]Scott A. Crosby. "Regular Expression Denial of Service". 2003 August. < https://web.archive.org/web/20031120114522/http://www.cs.rice.edu/~scrosby/hash/slides/USENIX-RegexpWIP.2.ppt >.

[REF-1162]Jan Goyvaerts. "Runaway Regular Expressions: Catastrophic Backtracking". 2019 December 2. < https://www.regular-expressions.info/catastrophic.html >.

[REF-1163]Adar Weidman. "Regular expression Denial of Service - ReDoS". < https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS >.

[REF-1164]Ilya Kantor. "Catastrophic backtracking". 2020 December 3. < https://javascript.info/regexp-catastrophic-backtracking >.

[REF-1165]Cristian-Alexandru Staicu and Michael Pradel. "Freezing the Web: A Study of ReDoS Vulnerabilities in JavaScript-based Web Servers". USENIX Security Symposium. 2018 July 1. < https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-staicu.pdf >.

[REF-1166]James C. Davis, Christy A. Coghlan, Francisco Servant and Dongyoon Lee. "The Impact of Regular Expression Denial of Service (ReDoS) in Practice: An Empirical Study at the Ecosystem Scale". 2018 August 1. < https://fservant.github.io/papers/Davis_Coghlan_Servant_Lee_ESECFSE18.pdf >.2023-04-07.

[REF-1167]James Davis. "The Regular Expression Denial of Service (ReDoS) cheat-sheet". 2020 May 3. < https://levelup.gitconnected.com/the-regular-expression-denial-of-service-redos-cheat-sheet-a78d0ed7d865 >.

CWE-1334: Unauthorized Error Injection Can Degrade Hardware Redundancy

Weakness ID : 1334
Structure : Simple
Abstraction : Base

Description

An unauthorized agent can inject errors into a redundant block to deprive the system of redundancy or put the system in a degraded operating mode.

Extended Description

To ensure the performance and functional reliability of certain components, hardware designers can implement hardware blocks for redundancy in the case that others fail. This redundant block can be prevented from performing as intended if the design allows unauthorized agents to inject errors into it. In this way, a path with injected errors may become unavailable to serve as a redundant channel. This may put the system into a degraded mode of operation which could be exploited by a subsequent attack.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	P	284	Improper Access Control	680

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Integrity Availability	DoS: Crash, Exit, or Restart DoS: Instability Quality Degradation DoS: Resource Consumption (CPU) DoS: Resource Consumption (Memory) DoS: Resource Consumption (Other) Reduce Performance Reduce Reliability Unexpected State	

Potential Mitigations

Phase: Architecture and Design

Ensure the design does not allow error injection in modes intended for normal run-time operation. Provide access controls on interfaces for injecting errors.

Phase: Implementation

Disallow error injection in modes which are expected to be used for normal run-time operation. Provide access controls on interfaces for injecting errors.

Phase: Integration

Add an access control layer atop any unprotected interfaces for injecting errors.

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1198	Privilege Separation and Access Control Issues	1194	2470
MemberOf	C	1396	Comprehensive Categorization: Access Control	1400	2519

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
624	Hardware Fault Injection
625	Mobile Device Fault Injection

CWE-1335: Incorrect Bitwise Shift of Integer

Weakness ID : 1335
Structure : Simple
Abstraction : Base

Description

An integer value is specified to be shifted by a negative amount or an amount greater than or equal to the number of bits contained in the value causing an unexpected or indeterminate result.

Extended Description

Specifying a value to be shifted by a negative amount is undefined in various languages. Various computer architectures implement this action in different ways. The compilers and interpreters when generating code to accomplish a shift generally do not do a check for this issue.

Specifying an over-shift, a shift greater than or equal to the number of bits contained in a value to be shifted, produces a result which varies by architecture and compiler. In some languages, this action is specifically listed as producing an undefined result.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	P	682	Incorrect Calculation	1499

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name	Page
MemberOf	C	189	Numeric Errors	2312

Applicable Platforms

Language : C *(Prevalence = Undetermined)*

Language : C++ *(Prevalence = Undetermined)*

Language : C# *(Prevalence = Undetermined)*

Language : Java *(Prevalence = Undetermined)*

Language : JavaScript *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Integrity	DoS: Crash, Exit, or Restart	

Potential Mitigations

Phase: Implementation

Implicitly or explicitly add checks and mitigation for negative or over-shift values.

Demonstrative Examples

Example 1:

A negative shift amount for an x86 or x86_64 shift instruction will produce the number of bits to be shifted by taking a 2's-complement of the shift amount and effectively masking that amount to the lowest 6 bits for a 64 bit shift instruction.

Example Language: C *(Bad)*

```
unsigned int r = 1 << -5;
```

The example above ends up with a shift amount of -5. The hexadecimal value is FFFFFFFFFFFFFFFD which, when bits above the 6th bit are masked off, the shift amount becomes a binary shift value of 111101 which is 61 decimal. A shift of 61 produces a very different result than -5. The previous example is a very simple version of the following code which is probably more realistic of what happens in a real system.

Example Language: C *(Bad)*

```
int choose_bit(int reg_bit, int bit_number_from_elsewhere)
{
  if (NEED_TO_SHIFT)
  {
    reg_bit -= bit_number_from_elsewhere;
  }
  return reg_bit;
}
unsigned int handle_io_register(unsigned int *r)
{
  unsigned int the_bit = 1 << choose_bit(5, 10);
  *r |= the_bit;
  return the_bit;
}
```

Example Language: C *(Good)*

```
int choose_bit(int reg_bit, int bit_number_from_elsewhere)
{
  if (NEED_TO_SHIFT)
  {
    reg_bit -= bit_number_from_elsewhere;
  }
  return reg_bit;
}
unsigned int handle_io_register(unsigned int *r)
{
  int the_bit_number = choose_bit(5, 10);
  if ((the_bit_number > 0) && (the_bit_number < 63))
  {
    unsigned int the_bit = 1 << the_bit_number;
    *r |= the_bit;
  }
  return the_bit;
}
```

Note that the good example not only checks for negative shifts and disallows them, but it also checks for over-shifts. No bit operation is done if the shift is out of bounds. Depending on the program, perhaps an error message should be logged.

Observed Examples

Reference	Description
CVE-2009-4307	An unexpected large value in the ext4 filesystem causes an overshift condition resulting in a divide by zero. *https://www.cve.org/CVERecord?id=CVE-2009-4307*
CVE-2012-2100	An unexpected large value in the ext4 filesystem causes an overshift condition resulting in a divide by zero - fix of CVE-2009-4307. *https://www.cve.org/CVERecord?id=CVE-2012-2100*
CVE-2020-8835	An overshift in a kernel allowed out of bounds reads and writes resulting in a root takeover. *https://www.cve.org/CVERecord?id=CVE-2020-8835*
CVE-2015-1607	Program is not properly handling signed bitwise left-shifts causing an overlapping memcpy memory range error. *https://www.cve.org/CVERecord?id=CVE-2015-1607*
CVE-2016-9842	Compression function improperly executes a signed left shift of a negative integer. *https://www.cve.org/CVERecord?id=CVE-2016-9842*
CVE-2018-18445	Some kernels improperly handle right shifts of 32 bit numbers in a 64 bit register. *https://www.cve.org/CVERecord?id=CVE-2018-18445*

Reference	Description
CVE-2013-4206	Putty has an incorrectly sized shift value resulting in an overshift. *https://www.cve.org/CVERecord?id=CVE-2013-4206*
CVE-2018-20788	LED driver overshifts under certain conditions resulting in a DoS. *https://www.cve.org/CVERecord?id=CVE-2018-20788*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1408	Comprehensive Categorization: Incorrect Calculation	1400	2534

CWE-1336: Improper Neutralization of Special Elements Used in a Template Engine

Weakness ID : 1336
Structure : Simple
Abstraction : Base

Description

The product uses a template engine to insert or process externally-influenced input, but it does not neutralize or incorrectly neutralizes special elements or syntax that can be interpreted as template expressions or other code directives when processed by the engine.

Extended Description

Many web applications use template engines that allow developers to insert externally-influenced values into free text or messages in order to generate a full web page, document, message, etc. Such engines include Twig, Jinja2, Pug, Java Server Pages, FreeMarker, Velocity, ColdFusion, Smarty, and many others - including PHP itself. Some CMS (Content Management Systems) also use templates.

Template engines often have their own custom command or expression language. If an attacker can influence input into a template before it is processed, then the attacker can invoke arbitrary expressions, i.e. perform injection attacks. For example, in some template languages, an attacker could inject the expression "{{7*7}}" and determine if the output returns "49" instead. The syntax varies depending on the language.

In some cases, XSS-style attacks can work, which can obscure the root cause if the developer does not closely investigate the root cause of the error.

Template engines can be used on the server or client, so both "sides" could be affected by injection. The mechanisms of attack or the affected technologies might be different, but the mistake is fundamentally the same.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	B	94	Improper Control of Generation of Code ('Code Injection')	219

Nature	Type	ID	Name	Page
PeerOf	B	917	Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')	1818

Applicable Platforms

Language : Java *(Prevalence = Undetermined)*

Language : PHP *(Prevalence = Undetermined)*

Language : Python *(Prevalence = Undetermined)*

Language : JavaScript *(Prevalence = Undetermined)*

Language : Interpreted *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Technology : Client Server *(Prevalence = Undetermined)*

Alternate Terms

Server-Side Template Injection / SSTI : This term is used for injection into template engines being used by a server.

Client-Side Template Injection / CSTI : This term is used for injection into template engines being used by a client.

Common Consequences

Scope	Impact	Likelihood
Integrity	Execute Unauthorized Code or Commands	

Potential Mitigations

Phase: Architecture and Design

Choose a template engine that offers a sandbox or restricted mode, or at least limits the power of any available expressions, function calls, or commands.

Phase: Implementation

Use the template engine's sandbox or restricted mode, if available.

Observed Examples

Reference	Description
CVE-2017-16783	server-side template injection in content management server *https://www.cve.org/CVERecord?id=CVE-2017-16783*
CVE-2020-9437	authentication / identity management product has client-side template injection *https://www.cve.org/CVERecord?id=CVE-2020-9437*
CVE-2020-12790	Server-Side Template Injection using a Twig template *https://www.cve.org/CVERecord?id=CVE-2020-12790*
CVE-2021-21244	devops platform allows SSTI *https://www.cve.org/CVERecord?id=CVE-2021-21244*
CVE-2020-4027	bypass of Server-Side Template Injection protection mechanism with macros in Velocity templates *https://www.cve.org/CVERecord?id=CVE-2020-4027*
CVE-2020-26282	web browser proxy server allows Java EL expressions from Server-Side Template Injection *https://www.cve.org/CVERecord?id=CVE-2020-26282*
CVE-2020-1961	SSTI involving mail templates and JEXL expressions *https://www.cve.org/CVERecord?id=CVE-2020-1961*
CVE-2019-19999	product does not use a "safe" setting for a FreeMarker configuration, allowing SSTI

Reference	Description
	https://www.cve.org/CVERecord?id=CVE-2019-19999
CVE-2018-20465	product allows read of sensitive database username/password variables using server-side template injection *https://www.cve.org/CVERecord?id=CVE-2018-20465*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1409	Comprehensive Categorization: Injection	1400	2535

Notes

Relationship

Since expression languages are often used in templating languages, there may be some overlap with CWE-917 (Expression Language Injection). XSS (CWE-79) is also co-located with template injection.

Maintenance

The interrelationships and differences between CWE-917 and CWE-1336 need to be further clarified.

References

[REF-1193]James Kettle. "Server-Side Template Injection". 2015 August 5. < https://portswigger.net/research/server-side-template-injection >.2023-04-07.

[REF-1194]James Kettle. "Server-Side Template Injection: RCE For The Modern Web App". 2015 December 7. < https://www.youtube.com/watch?v=3cT0uE7Y87s >.

CWE-1338: Improper Protections Against Hardware Overheating

Weakness ID : 1338
Structure : Simple
Abstraction : Base

Description

A hardware device is missing or has inadequate protection features to prevent overheating.

Extended Description

Hardware, electrical circuits, and semiconductor silicon have thermal side effects, such that some of the energy consumed by the device gets dissipated as heat and increases the temperature of the device. For example, in semiconductors, higher-operating frequency of silicon results in higher power dissipation and heat. The leakage current in CMOS circuits increases with temperature, and this creates positive feedback that can result in thermal runaway and damage the device permanently.

Any device lacking protections such as thermal sensors, adequate platform cooling, or thermal insulation is susceptible to attacks by malicious software that might deliberately operate the device in modes that result in overheating. This can be used as an effective denial of service (DoS) or permanent denial of service (PDoS) attack.

Depending on the type of hardware device and its expected usage, such thermal overheating can also cause safety hazards and reliability issues. Note that battery failures can also cause device overheating but the mitigations and examples included in this submission cannot reliably protect against a battery failure.

There can be similar weaknesses with lack of protection from attacks based on overvoltage or overcurrent conditions. However, thermal heat is generated by hardware operation and the device should implement protection from overheating.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	P	693	Protection Mechanism Failure	1520

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Technology : ICS/OT *(Prevalence = Undetermined)*

Technology : Power Management Hardware *(Prevalence = Undetermined)*

Technology : Processor Hardware *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Availability	DoS: Resource Consumption (Other)	High

Detection Methods

Dynamic Analysis with Manual Results Interpretation

Dynamic tests should be performed to stress-test temperature controls.

Effectiveness = High

Architecture or Design Review

Power management controls should be part of Architecture and Design reviews.

Effectiveness = High

Potential Mitigations

Phase: Architecture and Design

Temperature maximum and minimum limits should be enforced using thermal sensors both in silicon and at the platform level.

Phase: Implementation

The platform should support cooling solutions such as fans that can be modulated based on device-operation needs to maintain a stable temperature.

Demonstrative Examples

Example 1:

Malicious software running on a core can execute instructions that consume maximum power or increase core frequency. Such a power-virus program could execute on the platform for an extended time to overheat the device, resulting in permanent damage.

Execution core and platform do not support thermal sensors, performance throttling, or platform-cooling countermeasures to ensure that any software executing on the system cannot cause overheating past the maximum allowable temperature.

The platform and SoC should have failsafe thermal limits that are enforced by thermal sensors that trigger critical temperature alerts when high temperature is detected. Upon detection of high temperatures, the platform should trigger cooling or shutdown automatically.

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1206	Power, Clock, Thermal, and Reset Concerns	1194	2473
MemberOf	C	1367	ICS Dependencies (& Architecture): External Physical Systems	1358	2504
MemberOf	C	1413	Comprehensive Categorization: Protection Mechanism Failure	1400	2542

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
624	Hardware Fault Injection
625	Mobile Device Fault Injection

References

[REF-1156]Leonid Grustniy. "Loapi--This Trojan is hot!". 2017 December. < https://www.kaspersky.com/blog/loapi-trojan/20510/ >.

CWE-1339: Insufficient Precision or Accuracy of a Real Number

Weakness ID : 1339
Structure : Simple
Abstraction : Base

Description

The product processes a real number with an implementation in which the number's representation does not preserve required accuracy and precision in its fractional part, causing an incorrect result.

Extended Description

When a security decision or calculation requires highly precise, accurate numbers such as financial calculations or prices, then small variations in the number could be exploited by an attacker.

There are multiple ways to store the fractional part of a real number in a computer. In all of these cases, there is a limit to the accuracy of recording a fraction. If the fraction can be represented in a fixed number of digits (binary or decimal), there might not be enough digits assigned to represent the number. In other cases the number cannot be represented in a fixed number of digits due to repeating in decimal or binary notation (e.g. 0.333333...) or due to a transcendental number such as Π or $\sqrt{2}$. Rounding of numbers can lead to situations where the computer results do not adequately match the result of sufficiently accurate math.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	P	682	Incorrect Calculation	1499
PeerOf	B	190	Integer Overflow or Wraparound	472
CanPrecede	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	293
CanPrecede	C	834	Excessive Iteration	1754

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name	Page
MemberOf	C	189	Numeric Errors	2312

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Background Details

There are three major ways to store real numbers in computers. Each method is described along with the limitations of how they store their numbers.

- Fixed: Some implementations use a fixed number of binary bits to represent both the integer and the fraction. In the demonstrative example about Muller's Recurrence, the fraction 108.0 - ((815.0 - 1500.0 / z) / y) cannot be represented in 8 binary digits. The numeric accuracy within languages such as PL/1, COBOL and Ada is expressed in decimal digits rather than binary digits. In SQL and most databases, the length of the integer and the fraction are specified by the programmer in decimal. In the language C, fixed reals are implemented according to ISO/IEC TR18037
- Floating: The number is stored in a version of scientific notation with a fixed length for the base and the significand. This allows flexibility for more accuracy when the integer portion is smaller. When dealing with large integers, the fractional accuracy is less. Languages such as PL/1, COBOL and Ada set the accuracy by decimal digit representation rather than using binary digits. Python also implements decimal floating-point numbers using the IEEE 754-2008 encoding method.
- Ratio: The number is stored as the ratio of two integers. These integers also have their limits. These integers can be stored in a fixed number of bits or in a vector of digits. While the vector of digits method provides for very large integers, they cannot truly represent a repeating or transcendental number as those numbers do not ever have a fixed length.

Common Consequences

Scope	Impact	Likelihood
Availability	DoS: Crash, Exit, or Restart *This weakness will generally lead to undefined results and therefore crashes. In some implementations the program will halt if the weakness causes an overflow during a calculation.*	
Integrity	Execute Unauthorized Code or Commands	

Scope	Impact	Likelihood
	The results of the math are not as expected. This could cause issues where a value would not be properly calculated and provide an incorrect answer.	
Confidentiality Availability Access Control	Read Application Data Modify Application Data *This weakness can sometimes trigger buffer overflows which can be used to execute arbitrary code. This is usually outside the scope of a product's implicit security policy.*	

Potential Mitigations

Phase: Implementation

Phase: Patching and Maintenance

The developer or maintainer can move to a more accurate representation of real numbers. In extreme cases, the programmer can move to representations such as ratios of BigInts which can represent real numbers to extremely fine precision. The programmer can also use the concept of an Unum real. The memory and CPU tradeoffs of this change must be examined. Since floating point reals are used in many products and many locations, they are implemented in hardware and most format changes will cause the calculations to be moved into software resulting in slower products.

Demonstrative Examples

Example 1:

Muller's Recurrence is a series that is supposed to converge to the number 5. When running this series with the following code, different implementations of real numbers fail at specific iterations:

Example Language: Rust *(Bad)*

```
fn rec_float(y: f64, z: f64) -> f64
{
  108.0 - ((815.0 - 1500.0 / z) / y);
}
fn float_calc(turns: usize) -> f64
{
  let mut x: Vec<f64> = vec![4.0, 4.25];
  (2..turns + 1).for_each(|number|
  {
    x.push(rec_float(x[number - 1], x[number - 2]));
  });
  x[turns]
}
```

The chart below shows values for different data structures in the rust language when Muller's recurrence is executed to 80 iterations. The data structure f64 is a 64 bit float. The data structures I<number>F<number> are fixed representations 128 bits in length that use the first number as the size of the integer and the second size as the size of the fraction (e.g. I16F112 uses 16 bits for the integer and 112 bits for the fraction). The data structure of Ratio comes in three different implementations: i32 uses a ratio of 32 bit signed integers, i64 uses a ratio of 64 bit signed integers and BigInt uses a ratio of signed integer with up to 2^32 digits of base 256. Notice how even with 112 bits of fractions or ratios of 64bit unsigned integers, this math still does not converge to an expected value of 5.

Example Language: Rust *(Good)*

```
Use num_rational::BigRational;
fn rec_big(y: BigRational, z: BigRational) -> BigRational
{
```

```
    BigRational::from_integer(BigInt::from(108))
      - ((BigRational::from_integer(BigInt::from(815))
      - BigRational::from_integer(BigInt::from(1500)) / z)
      / y)
}
fn big_calc(turns: usize) -> BigRational
{
    let mut x: Vec<BigRational> = vec![BigRational::from_float(4.0).unwrap(), BigRational::from_float(4.25).unwrap(),];
    (2..turns + 1).for_each(|number|
    {
      x.push(rec_big(x[number - 1].clone(), x[number - 2].clone()));
    });
    x[turns].clone()
}
```

Example 2:

On February 25, 1991, during the eve of the Iraqi invasion of Saudi Arabia, a Scud missile fired from Iraqi positions hit a US Army barracks in Dhahran, Saudi Arabia. It miscalculated time and killed 28 people [REF-1190].

Example 3:

Sleipner A, an offshore drilling platform in the North Sea, was incorrectly constructed with an underestimate of 50% of strength in a critical cluster of buoyancy cells needed for construction. This led to a leak in buoyancy cells during lowering, causing a seismic event of 3.0 on the Richter Scale and about $700M loss [REF-1281].

Observed Examples

Reference	Description
CVE-2018-16069	Chain: series of floating-point precision errors (CWE-1339) in a web browser rendering engine causes out-of-bounds read (CWE-125), giving access to cross-origin data *https://www.cve.org/CVERecord?id=CVE-2018-16069*
CVE-2017-7619	Chain: rounding error in floating-point calculations (CWE-1339) in image processor leads to infinite loop (CWE-835) *https://www.cve.org/CVERecord?id=CVE-2017-7619*
CVE-2021-29529	Chain: machine-learning product can have a heap-based buffer overflow (CWE-122) when some integer-oriented bounds are calculated by using ceiling() and floor() on floating point values (CWE-1339) *https://www.cve.org/CVERecord?id=CVE-2021-29529*
CVE-2008-2108	Chain: insufficient precision (CWE-1339) in random-number generator causes some zero bits to be reliably generated, reducing the amount of entropy (CWE-331) *https://www.cve.org/CVERecord?id=CVE-2008-2108*
CVE-2006-6499	Chain: web browser crashes due to infinite loop - "bad looping logic [that relies on] floating point math [CWE-1339] to exit the loop [CWE-835]" *https://www.cve.org/CVERecord?id=CVE-2006-6499*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1408	Comprehensive Categorization: Incorrect Calculation	1400	2534

References

[REF-1186]"Is COBOL holding you hostage with Math?". 2018 July 8. < https://medium.com/the-technical-archaeologist/is-cobol-holding-you-hostage-with-math-5498c0eb428b >.

[REF-1187]"Intermediate results and arithmetic precision". 2021 June 0. < https://www.ibm.com/docs/en/cobol-zos/6.2?topic=appendixes-intermediate-results-arithmetic-precision >.

[REF-1188]"8.1.2. Arbitrary Precision Numbers". 2021 June 4. < https://www.postgresql.org/docs/8.3/datatype-numeric.html#DATATYPE-NUMERIC-DECIMAL >.

[REF-1189]"Muller's Recurrence". 2017 November 1. < https://scipython.com/blog/mullers-recurrence/ >.

[REF-1190]"An Improvement To Floating Point Numbers". 2015 October 2. < https://hackaday.com/2015/10/22/an-improvement-to-floating-point-numbers/ >.

[REF-1191]"HIGH PERFORMANCE COMPUTING: ARE WE JUST GETTING WRONG ANSWERS FASTER?". 1999 June 3. < https://www3.nd.edu/~markst/cast-award-speech.pdf >.

CWE-1341: Multiple Releases of Same Resource or Handle

Weakness ID : 1341
Structure : Simple
Abstraction : Base

Description

The product attempts to close or release a resource or handle more than once, without any successful open between the close operations.

Extended Description

Code typically requires "opening" handles or references to resources such as memory, files, devices, socket connections, services, etc. When the code is finished with using the resource, it is typically expected to "close" or "release" the resource, which indicates to the environment (such as the OS) that the resource can be re-assigned or reused by unrelated processes or actors - or in some cases, within the same process. API functions or other abstractions are often used to perform this release, such as free() or delete() within C/C++, or file-handle close() operations that are used in many languages.

Unfortunately, the implementation or design of such APIs might expect the developer to be responsible for ensuring that such APIs are only called once per release of the resource. If the developer attempts to release the same resource/handle more than once, then the API's expectations are not met, resulting in undefined and/or insecure behavior. This could lead to consequences such as memory corruption, data corruption, execution path corruption, or other consequences.

Note that while the implementation for most (if not all) resource reservation allocations involve a unique identifier/pointer/symbolic reference, then if this identifier is reused, checking the identifier for resource closure may result in a false state of openness and closing of the wrong resource. For this reason, reuse of identifiers is discouraged.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf		675	Multiple Operations on Resource in Single-Operation Context	1487

Nature	Type	ID	Name	Page
ParentOf	V	415	Double Free	1008
CanPrecede	C	672	Operation on a Resource after Expiration or Release	1479

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name	Page
MemberOf	C	399	Resource Management Errors	2324

Applicable Platforms

Language : Java *(Prevalence = Undetermined)*

Language : Rust *(Prevalence = Undetermined)*

Language : Not Language-Specific *(Prevalence = Undetermined)*

Language : C *(Prevalence = Undetermined)*

Language : C++ *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Availability Integrity	DoS: Crash, Exit, or Restart	Medium

Detection Methods

Automated Static Analysis

For commonly-used APIs and resource types, automated tools often have signatures that can spot this issue.

Automated Dynamic Analysis

Some compiler instrumentation tools such as AddressSanitizer (ASan) can indirectly detect some instances of this weakness.

Potential Mitigations

Phase: Implementation

Change the code's logic so that the resource is only closed once. This might require simplifying or refactoring. This fix can be simple to do in small code blocks, but more difficult when multiple closes are buried within complex conditionals.

Phase: Implementation

Strategy = Refactoring

It can be effective to implement a flag that is (1) set when the resource is opened, (2) cleared when it is closed, and (3) checked before closing. This approach can be useful when there are disparate cases in which closes must be performed. However, flag-tracking can increase code complexity and requires diligent compliance by the programmer.

Phase: Implementation

Strategy = Refactoring

When closing a resource, set the resource's associated variable to NULL or equivalent value for the given language. Some APIs will ignore this null value without causing errors. For other APIs, this can lead to application crashes or exceptions, which may still be preferable to corrupting an unintended resource such as memory or data.

Effectiveness = Defense in Depth

Demonstrative Examples

Example 1:

This example attempts to close a file twice. In some cases, the C library fclose() function will catch the error and return an error code. In other implementations, a double-free (CWE-415) occurs, causing the program to fault. Note that the examples presented here are simplistic, and double fclose() calls will frequently be spread around a program, making them more difficult to find during code reviews.

Example Language: C *(Bad)*

```
char b[2000];
FILE *f = fopen("dbl_cls.c", "r");
if (f)
{
  b[0] = 0;
  fread(b, 1, sizeof(b) - 1, f);
  printf("%s\n'", b);
  int r1 = fclose(f);
  printf("\n-----------------\n1 close done '%d'\n", r1);
  int r2 = fclose(f); // Double close
  printf("2 close done '%d'\n", r2);
}
```

There are multiple possible fixes. This fix only has one call to fclose(), which is typically the preferred handling of this problem - but this simplistic method is not always possible.

Example Language: C *(Good)*

```
char b[2000];
FILE *f = fopen("dbl_cls.c", "r");
if (f)
{
  b[0] = 0;
  fread(b, 1, sizeof(b) - 1, f);
  printf("%s\n'", b);
  int r = fclose(f);
  printf("\n-----------------\n1 close done '%d'\n", r);
}
```

This fix uses a flag to call fclose() only once. Note that this flag is explicit. The variable "f" could also have been used as it will be either NULL if the file is not able to be opened or a valid pointer if the file was successfully opened. If "f" is replacing "f_flg" then "f" would need to be set to NULL after the first fclose() call so the second fclose call would never be executed.

Example Language: C *(Good)*

```
char b[2000];
int f_flg = 0;
FILE *f = fopen("dbl_cls.c", "r");
if (f)
{
  f_flg = 1;
  b[0] = 0;
  fread(b, 1, sizeof(b) - 1, f);
  printf("%s\n'", b);
  if (f_flg)
  {
    int r1 = fclose(f);
    f_flg = 0;
    printf("\n-----------------\n1 close done '%d'\n", r1);
  }
  if (f_flg)
```

```
  {
    int r2 = fclose(f); // Double close
    f_flg = 0;
    printf("2 close done '%d'\n", r2);
  }
}
```

Example 2:

The following code shows a simple example of a double free vulnerability.

Example Language: C *(Bad)*

```
char* ptr = (char*)malloc (SIZE);
...
if (abrt) {
  free(ptr);
}
...
free(ptr);
```

Double free vulnerabilities have two common (and sometimes overlapping) causes:

- Error conditions and other exceptional circumstances
- Confusion over which part of the program is responsible for freeing the memory

Although some double free vulnerabilities are not much more complicated than this example, most are spread out across hundreds of lines of code or even different files. Programmers seem particularly susceptible to freeing global variables more than once.

Observed Examples

Reference	Description
CVE-2019-13351	file descriptor double close can cause the wrong file to be associated with a file descriptor. *https://www.cve.org/CVERecord?id=CVE-2019-13351*
CVE-2006-5051	Chain: Signal handler contains too much functionality (CWE-828), introducing a race condition that leads to a double free (CWE-415). *https://www.cve.org/CVERecord?id=CVE-2006-5051*
CVE-2004-0772	Double free resultant from certain error conditions. *https://www.cve.org/CVERecord?id=CVE-2004-0772*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1412	Comprehensive Categorization: Poor Coding Practices	1400	2538

Notes

Terminology

The terms related to "release" may vary depending on the type of resource, programming language, specification, or framework. "Close" has been used synonymously for the release of resources like file descriptors and file handles. "Return" is sometimes used instead of Release. "Free" is typically used when releasing memory or buffers back into the system for reuse.

References

[REF-1198]"close - Perldoc Browser". < https://perldoc.perl.org/functions/close >.

[REF-1199]"io - Core tools for working with streams — Python 3.9.7 documentation". 2021 September 2. < https://docs.python.org/3.9/library/io.html#io.IOBase.close >.

[REF-1200]"FileOutputStream (Java Platform SE 7)". 2020. < https://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html >.

[REF-1201]"FileOutputStream (Java SE 11 & JDK 11)". 2021. < https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/io/FileOutputStream.html >.

CWE-1342: Information Exposure through Microarchitectural State after Transient Execution

Weakness ID : 1342
Structure : Simple
Abstraction : Base

Description

The processor does not properly clear microarchitectural state after incorrect microcode assists or speculative execution, resulting in transient execution.

Extended Description

In many processor architectures an exception, mis-speculation, or microcode assist results in a flush operation to clear results that are no longer required. This action prevents these results from influencing architectural state that is intended to be visible from software. However, traces of this transient execution may remain in microarchitectural buffers, resulting in a change in microarchitectural state that can expose sensitive information to an attacker using side-channel analysis. For example, Load Value Injection (LVI) [REF-1202] can exploit direct injection of erroneous values into intermediate load and store buffers.

Several conditions may need to be fulfilled for a successful attack:

1. incorrect transient execution that results in remanence of sensitive information;
2. attacker has the ability to provoke microarchitectural exceptions;
3. operations and structures in victim code that can be exploited must be identified.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	B	226	Sensitive Information in Resource Not Removed Before Reuse	562

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name	Page
ChildOf	B	226	Sensitive Information in Resource Not Removed Before Reuse	562

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Workstation *(Prevalence = Undetermined)*

Architecture : x86 *(Prevalence = Undetermined)*

Architecture : ARM *(Prevalence = Undetermined)*

Architecture : Other *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Technology : System on Chip *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality Integrity	Modify Memory Read Memory Execute Unauthorized Code or Commands	Medium

Potential Mitigations

Phase: Architecture and Design

Phase: Requirements

Hardware ensures that no illegal data flows from faulting micro-ops exists at the microarchitectural level.

Effectiveness = High

Being implemented in silicon it is expected to fully address the known weaknesses with limited performance impact.

Phase: Build and Compilation

Include instructions that explicitly remove traces of unneeded computations from software interactions with microarchitectural elements e.g. lfence, sfence, mfence, clflush.

Effectiveness = High

This effectively forces the processor to complete each memory access before moving on to the next operation. This may have a large performance impact.

Demonstrative Examples

Example 1:

Faulting loads in a victim domain may trigger incorrect transient forwarding, which leaves secret-dependent traces in the microarchitectural state. Consider this example from [REF-1203].

Consider the code gadget:

Example Language: C *(Bad)*

```
void call_victim(size_t untrusted_arg) {
  *arg_copy = untrusted_arg;
  array[**trusted_ptr * 4096];
}
```

A processor with this weakness will store the value of untrusted_arg (which may be provided by an attacker) to the stack, which is trusted memory. Additionally, this store operation will save this value in some microarchitectural buffer, e.g. the store queue.

In this code gadget, trusted_ptr is dereferenced while the attacker forces a page fault. The faulting load causes the processor to mis-speculate by forwarding untrusted_arg as the (speculative) load result. The processor then uses untrusted_arg for the pointer dereference. After the fault has been handled and the load has been re-issued with the correct argument, secret-dependent information stored at the address of trusted_ptr remains in microarchitectural state and can be extracted by an attacker using a code gadget.

Observed Examples

Reference	Description
CVE-2020-0551	Load value injection in some processors utilizing speculative execution may allow an authenticated user to enable information disclosure via a side-channel with local access. *https://www.cve.org/CVERecord?id=CVE-2020-0551*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1201	Core and Compute Issues	1194	2471
MemberOf	C	1416	Comprehensive Categorization: Resource Lifecycle Management	1400	2545

Notes

Relationship

CWE-1342 differs from CWE-1303, which is related to misprediction and biasing microarchitectural components, while CWE-1342 addresses illegal data flows and retention. For example, Spectre is an instance of CWE-1303 biasing branch prediction to steer the transient execution indirectly.

Maintenance

As of CWE 4.9, members of the CWE Hardware SIG are closely analyzing this entry and others to improve CWE's coverage of transient execution weaknesses, which include issues related to Spectre, Meltdown, and other attacks. Additional investigation may include other weaknesses related to microarchitectural state. As a result, this entry might change significantly in CWE 4.10.

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
696	Load Value Injection

References

[REF-1202]Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yuval Yarom, Berk Sunar, Daniel Gruss, and Frank Piessens. "LVI - Hijacking Transient Execution with Load Value Injection". 2020. < https://lviattack.eu/ >.

[REF-1203]Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yuval Yarom, Berk Sunar, Daniel Gruss, and Frank Piessens. "LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection". 2020 January 9. < https://lviattack.eu/lvi.pdf >.

[REF-1204]"Hijacking Transient Execution through Microarchitectural Load Value Injection". 2020 May 8. < https://www.youtube.com/watch?v=99kVz-YGi6Y >.

[REF-1205]Stephan van Schaik, Marina Minkin, Andrew Kwong, Daniel Genkin, Yuval Yarom. "CacheOut: Leaking Data on Intel CPUs via Cache Evictions". 2020 December 8. < https://cacheoutattack.com/files/CacheOut.pdf >.

CWE-1351: Improper Handling of Hardware Behavior in Exceptionally Cold Environments

Weakness ID : 1351
Structure : Simple

Abstraction : Base

Description

A hardware device, or the firmware running on it, is missing or has incorrect protection features to maintain goals of security primitives when the device is cooled below standard operating temperatures.

Extended Description

The hardware designer may improperly anticipate hardware behavior when exposed to exceptionally cold conditions. As a result they may introduce a weakness by not accounting for the modified behavior of critical components when in extreme environments.

An example of a change in behavior is that power loss won't clear/reset any volatile state when cooled below standard operating temperatures. This may result in a weakness when the starting state of the volatile memory is being relied upon for a security decision. For example, a Physical Unclonable Function (PUF) may be supplied as a security primitive to improve confidentiality, authenticity, and integrity guarantees. However, when the PUF is paired with DRAM, SRAM, or another temperature sensitive entropy source, the system designer may introduce weakness by failing to account for the chosen entropy source's behavior at exceptionally low temperatures. In the case of DRAM and SRAM, when power is cycled at low temperatures, the device will not contain the bitwise biasing caused by inconsistencies in manufacturing and will instead contain the data from previous boot. Should the PUF primitive be used in a cryptographic construction which does not account for full adversary control of PUF seed data, weakness would arise.

This weakness does not cover "Cold Boot Attacks" wherein RAM or other external storage is super cooled and read externally by an attacker.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf		1384	Improper Handling of Physical or Environmental Conditions	2257

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Embedded *(Prevalence = Undetermined)*

Architecture : Microcomputer *(Prevalence = Undetermined)*

Technology : System on Chip *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Integrity Authentication	Varies by Context Unexpected State *Consequences of this weakness are highly contextual.*	Low

Potential Mitigations

Phase: Architecture and Design

The system should account for security primitive behavior when cooled outside standard temperatures.

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1205	Security Primitives and Cryptography Issues	1194	2473
MemberOf	C	1365	ICS Communications: Unreliability	1358	2502
MemberOf	C	1388	Physical Access Issues and Concerns	1194	2518
MemberOf	C	1405	Comprehensive Categorization: Improper Check or Handling of Exceptional Conditions	1400	2531

Related Attack Patterns

CAPEC-ID	Attack Pattern Name
624	Hardware Fault Injection
625	Mobile Device Fault Injection

References

[REF-1181]Nikolaos Athanasios Anagnostopoulos, Tolga Arul, Markus Rosenstihl, André Schaller, Sebastian Gabmeyer and Stefan Katzenbeisser. "Low-Temperature Data Remnanence Attacks Against Intrinsic SRAM PUFs". 2018 October 5. < https://ieeexplore.ieee.org/abstract/document/8491873/ >.

[REF-1182]Yuan Cao, Yunyi Guo, Benyu Liu, Wei Ge, Min Zhu and Chip-Hong Chang. "A Fully Digital Physical Unclonable Function Based Temperature Sensor for Secure Remote Sensing". 2018 October 1. < https://ieeexplore.ieee.org/abstract/document/8487347/ >.

[REF-1183] Urbi Chatterjee, Soumi Chatterjee, Debdeep Mukhopadhyay and Rajat Subhra Chakraborty. "Machine Learning Assisted PUF Calibration for Trustworthy Proof of Sensor Data in IoT". 2020 June. < https://dl.acm.org/doi/abs/10.1145/3393628 >.2023-04-07.

CWE-1357: Reliance on Insufficiently Trustworthy Component

Weakness ID : 1357
Structure : Simple
Abstraction : Class

Description

The product is built from multiple separate components, but it uses a component that is not sufficiently trusted to meet expectations for security, reliability, updateability, and maintainability.

Extended Description

Many modern hardware and software products are built by combining multiple smaller components together into one larger entity, often during the design or architecture phase. For example, a hardware component might be built by a separate supplier, or the product might use an open-source software library from a third party.

Regardless of the source, each component should be sufficiently trusted to ensure correct, secure operation of the product. If a component is not trustworthy, it can produce significant risks for the overall product, such as vulnerabilities that cannot be patched fast enough (if at all); hidden functionality such as malware; inability to update or replace the component if needed for security purposes; hardware components built from parts that do not meet specifications in ways that can lead to weaknesses; etc. Note that a component might not be trustworthy even if it is owned by the product vendor, such as a software component whose source code is lost and was built by developers who left the company, or a component that was developed by a separate company that was acquired and brought into the product's own company.

Note that there can be disagreement as to whether a component is sufficiently trustworthy, since trust is ultimately subjective. Different stakeholders (e.g., customers, vendors, governments) have various threat models and ways to assess trust, and design/architecture choices might make tradeoffs between security, reliability, safety, privacy, cost, and other characteristics.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	P	710	Improper Adherence to Coding Standards	1549
ParentOf	B	1104	Use of Unmaintained Third Party Components	1944
ParentOf	B	1329	Reliance on Component That is Not Updateable	2219

Weakness Ordinalities

Indirect :

Applicable Platforms

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Technology : ICS/OT *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Other	Reduce Maintainability	

Potential Mitigations

Phase: Requirements

Phase: Architecture and Design

Phase: Implementation

For each component, ensure that its supply chain is well-controlled with sub-tier suppliers using best practices. For third-party software components such as libraries, ensure that they are developed and actively maintained by reputable vendors.

Phase: Architecture and Design

Phase: Implementation

Phase: Integration

Phase: Manufacturing

Maintain a Bill of Materials for all components and sub-components of the product. For software, maintain a Software Bill of Materials (SBOM). According to [REF-1247], "An SBOM is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships."

Phase: Operation

Phase: Patching and Maintenance

Continue to monitor changes in each of the product's components, especially when the changes indicate new vulnerabilities, end-of-life (EOL) plans, supplier practices that affect trustworthiness, etc.

Observed Examples

Reference	Description
CVE-2020-9054	Chain: network-attached storage (NAS) device has a critical OS command injection (CWE-78) vulnerability that is actively exploited to place IoT devices into a botnet, but some products are "end-of-support" and cannot be patched (CWE-1277). [REF-1097] *https://www.cve.org/CVERecord?id=CVE-2020-9054*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1208	Cross-Cutting Problems	1194	2474
MemberOf	C	1367	ICS Dependencies (& Architecture): External Physical Systems	1358	2504
MemberOf	C	1368	ICS Dependencies (& Architecture): External Digital Systems	1358	2505
MemberOf	C	1370	ICS Supply Chain: Common Mode Frailties	1358	2507
MemberOf	C	1412	Comprehensive Categorization: Poor Coding Practices	1400	2538

Notes

Maintenance

As of CWE 4.10, the name and description for this entry has undergone significant change and is still under public discussion, especially by members of the HW SIG.

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
ISA/IEC 62443	Part 2-4		Req SP.03.02 RE(1)
ISA/IEC 62443	Part 2-4		Req SP.03.02 RE(2)
ISA/IEC 62443	Part 3-3		Req SR 1.13
ISA/IEC 62443	Part 4-2		Req EDR 3.12
ISA/IEC 62443	Part 4-2		Req HDR 3.12
ISA/IEC 62443	Part 4-2		Req NDR 3.12
ISA/IEC 62443	Part 4-2		Req EDR 3.13
ISA/IEC 62443	Part 4-2		Req HDR 3.13
ISA/IEC 62443	Part 4-2		Req NDR 3.13
ISA/IEC 62443	Part 4-2		Req CR-7.8
ISA/IEC 62443	Part 4-1		Req SM-6
ISA/IEC 62443	Part 4-1		Req SM-9
ISA/IEC 62443	Part 4-1		Req SM-10

References

[REF-1212]"A06:2021 - Vulnerable and Outdated Components". 2021 September 4. OWASP. < https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/ >.

[REF-1246]National Telecommunications and Information Administration. "SOFTWARE BILL OF MATERIALS". < https://ntia.gov/page/software-bill-materials >.2023-04-07.

[REF-1247]NTIA Multistakeholder Process on Software Component Transparency Framing Working Group. "Framing Software Component Transparency: Establishing a Common Software

Bill of Materials (SBOM)". 2021 October 1. < https://www.ntia.gov/files/ntia/publications/ntia_sbom_framing_2nd_edition_20211021.pdf >.

[REF-1097]Brian Krebs. "Zyxel Flaw Powers New Mirai IoT Botnet Strain". 2020 March 0. < https://krebsonsecurity.com/2020/03/zyxel-flaw-powers-new-mirai-iot-botnet-strain/ >.

CWE-1384: Improper Handling of Physical or Environmental Conditions

Weakness ID : 1384
Structure : Simple
Abstraction : Class

Description

The product does not properly handle unexpected physical or environmental conditions that occur naturally or are artificially induced.

Extended Description

Hardware products are typically only guaranteed to behave correctly within certain physical limits or environmental conditions. Such products cannot necessarily control the physical or external conditions to which they are subjected. However, the inability to handle such conditions can undermine a product's security. For example, an unexpected physical or environmental condition may cause the flipping of a bit that is used for an authentication decision. This unexpected condition could occur naturally or be induced artificially by an adversary.

Physical or environmental conditions of concern are:

- Atmospheric characteristics: extreme temperature ranges, etc.
- Interference: electromagnetic interference (EMI), radio frequency interference (RFI), etc.
- Assorted light sources: white light, ultra-violet light (UV), lasers, infrared (IR), etc.
- Power variances: under-voltages, over-voltages, under-current, over-current, etc.
- Clock variances: glitching, overclocking, clock stretching, etc.
- Component aging and degradation
- Materials manipulation: focused ion beams (FIB), etc.
- Exposure to radiation: x-rays, cosmic radiation, etc.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	P	703	Improper Check or Handling of Exceptional Conditions	1535
ParentOf	B	1247	Improper Protection Against Voltage and Clock Glitches	2044
ParentOf	B	1261	Improper Handling of Single Event Upsets	2079
ParentOf	B	1332	Improper Handling of Faults that Lead to Instruction Skips	2227
ParentOf	B	1351	Improper Handling of Hardware Behavior in Exceptionally Cold Environments	2252

Applicable Platforms

Technology : System on Chip *(Prevalence = Undetermined)*

Technology : ICS/OT *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality Integrity Availability	Varies by Context Unexpected State *Consequences of this weakness are highly dependent on the role of affected components within the larger product.*	

Potential Mitigations

Phase: Requirements

In requirements, be specific about expectations for how the product will perform when it exceeds physical and environmental boundary conditions, e.g., by shutting down.

Phase: Architecture and Design

Phase: Implementation

Where possible, include independent components that can detect excess environmental conditions and have the capability to shut down the product.

Phase: Architecture and Design

Phase: Implementation

Where possible, use shielding or other materials that can increase the adversary's workload and reduce the likelihood of being able to successfully trigger a security-related failure.

Observed Examples

Reference	Description
CVE-2019-17391	Lack of anti-glitch protections allows an attacker to launch a physical attack to bypass the secure boot and read protected eFuses. *https://www.cve.org/CVERecord?id=CVE-2019-17391*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1365	ICS Communications: Unreliability	1358	2502
MemberOf	C	1367	ICS Dependencies (& Architecture): External Physical Systems	1358	2504
MemberOf	C	1388	Physical Access Issues and Concerns	1194	2518
MemberOf	C	1405	Comprehensive Categorization: Improper Check or Handling of Exceptional Conditions	1400	2531

References

[REF-1248]Securing Energy Infrastructure Executive Task Force (SEI ETF). "Categories of Security Vulnerabilities in ICS". 2022 March 9. < https://inl.gov/wp-content/uploads/2022/03/SEI-ETF-NCSV-TPT-Categories-of-Security-Vulnerabilities-ICS-v1_03-09-22.pdf >.

[REF-1255]Sergei P. Skorobogatov. "Semi-invasive attacks - A new approach to hardware security analysis". 2005 April. < https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.pdf >.

[REF-1285]Texas Instruments. "Physical Security Attacks Against Silicon Devices". 2022 January 1. < https://www.ti.com/lit/an/swra739/swra739.pdf?ts=1644234570420 >.

[REF-1286]Lennert Wouters, Benedikt Gierlichs and Bart Preneel. "On The Susceptibility of Texas Instruments SimpleLink Platform Microcontrollers to Non-Invasive Physical Attacks". 2022 March 4. < https://eprint.iacr.org/2022/328.pdf >.

CWE-1385: Missing Origin Validation in WebSockets

Weakness ID : 1385
Structure : Simple
Abstraction : Variant

Description

The product uses a WebSocket, but it does not properly verify that the source of data or communication is valid.

Extended Description

WebSockets provide a bi-directional low latency communication (near real-time) between a client and a server. WebSockets are different than HTTP in that the connections are long-lived, as the channel will remain open until the client or the server is ready to send the message, whereas in HTTP, once the response occurs (which typically happens immediately), the transaction completes.

A WebSocket can leverage the existing HTTP protocol over ports 80 and 443, but it is not limited to HTTP. WebSockets can make cross-origin requests that are not restricted by browser-based protection mechanisms such as the Same Origin Policy (SOP) or Cross-Origin Resource Sharing (CORS). Without explicit origin validation, this makes CSRF attacks more powerful.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	346	Origin Validation Error	853

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Technology : Web Server *(Prevalence = Undetermined)*

Alternate Terms

Cross-Site WebSocket hijacking (CSWSH) : this term is used for attacks that exploit this weakness

Common Consequences

Scope	Impact	Likelihood
Confidentiality Integrity Availability Non-Repudiation Access Control	Varies by Context Gain Privileges or Assume Identity Bypass Protection Mechanism Read Application Data Modify Application Data DoS: Crash, Exit, or Restart *The consequences will vary depending on the nature of the functionality that is vulnerable to CSRF. An attacker could effectively perform any operations as the victim. If the victim is an administrator or privileged user, the consequences may include obtaining complete control over the web application - deleting or stealing data, uninstalling the product, or using it to launch other attacks against all of the product's users. Because the attacker has*	

Scope	Impact	Likelihood
	the identity of the victim, the scope of the CSRF is limited only by the victim's privileges.	

Potential Mitigations

Phase: Implementation

Enable CORS-like access restrictions by verifying the 'Origin' header during the WebSocket handshake.

Phase: Implementation

Use a randomized CSRF token to verify requests.

Phase: Implementation

Use TLS to securely communicate using 'wss' (WebSocket Secure) instead of 'ws'.

Phase: Architecture and Design

Phase: Implementation

Require user authentication prior to the WebSocket connection being established. For example, the WS library in Node has a 'verifyClient' function.

Phase: Implementation

Leverage rate limiting to prevent against DoS. Use of the leaky bucket algorithm can help with this.

Effectiveness = Defense in Depth

Phase: Implementation

Use a library that provides restriction of the payload size. For example, WS library for Node includes 'maxPayloadoption' that can be set.

Effectiveness = Defense in Depth

Phase: Implementation

Treat data/input as untrusted in both directions and apply the same data/input sanitization as XSS, SQLi, etc.

Observed Examples

Reference	Description
CVE-2020-25095	web console for SIEM product does not check Origin header, allowing Cross Site WebSocket Hijacking (CSWH) *https://www.cve.org/CVERecord?id=CVE-2020-25095*
CVE-2018-6651	Chain: gaming client attempts to validate the Origin header, but only uses a substring, allowing Cross-Site WebSocket hijacking by forcing requests from an origin whose hostname is a substring of the valid origin. *https://www.cve.org/CVERecord?id=CVE-2018-6651*
CVE-2018-14730	WebSocket server does not check the origin of requests, allowing attackers to steal developer's code using a ws://127.0.0.1:3123/ connection. *https://www.cve.org/CVERecord?id=CVE-2018-14730*
CVE-2018-14731	WebSocket server does not check the origin of requests, allowing attackers to steal developer's code using a ws://127.0.0.1/ connection to a randomized port number. *https://www.cve.org/CVERecord?id=CVE-2018-14731*
CVE-2018-14732	WebSocket server does not check the origin of requests, allowing attackers to steal developer's code using a ws://127.0.0.1:8080/ connection. *https://www.cve.org/CVERecord?id=CVE-2018-14732*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1411	Comprehensive Categorization: Insufficient Verification of Data Authenticity	1400	2538

References

[REF-1257]Christian Schneider. "Cross-Site WebSocket Hijacking (CSWSH)". 2013 September 1. < https://christian-schneider.net/CrossSiteWebSocketHijacking.html >.

[REF-1251]Drew Branch. "WebSockets not Bound by SOP and CORS? Does this mean...". 2018 June 6. < https://blog.securityevaluators.com/websockets-not-bound-by-cors-does-this-mean-2e7819374acc >.

[REF-1252]Mehul Mohan. "How to secure your WebSocket connections". 2018 November 2. < https://www.freecodecamp.org/news/how-to-secure-your-websocket-connections-d0be0996c556/ >.

[REF-1256]Vickie Li. "Cross-Site WebSocket Hijacking (CSWSH)". 2019 November 7. < https://medium.com/swlh/hacking-websocket-25d3cba6a4b9 >.

[REF-1253]PortSwigger. "Testing for WebSockets security vulnerabilities". < https://portswigger.net/web-security/websockets >.2023-04-07.

CWE-1386: Insecure Operation on Windows Junction / Mount Point

Weakness ID : 1386
Structure : Simple
Abstraction : Base

Description

The product opens a file or directory, but it does not properly prevent the name from being associated with a junction or mount point to a destination that is outside of the intended control sphere.

Extended Description

Depending on the intended action being performed, this could allow an attacker to cause the product to read, write, delete, or otherwise operate on unauthorized files.

In Windows, NTFS5 allows for file system objects called reparse points. Applications can create a hard link from one directory to another directory, called a junction point. They can also create a mapping from a directory to a drive letter, called a mount point. If a file is used by a privileged program, but it can be replaced with a hard link to a sensitive file (e.g., AUTOEXEC.BAT), an attacker could excalate privileges. When the process opens the file, the attacker can assume the privileges of that process, tricking the privileged process to read, modify, or delete the sensitive file, preventing the program from accurately processing data. Note that one can also point to registries and semaphores.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	B	59	Improper Link Resolution Before File Access ('Link Following')	111

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Windows *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality	Read Files or Directories *Read arbitrary files by replacing a user-controlled folder with a mount point and additional hard links.*	
Integrity	Modify Files or Directories *Modify an arbitrary file by replacing the rollback files in installer directories, as they can have the installer execute those rollbacks.*	
Availability	Modify Files or Directories *Even if there is no control of contents, an arbitrary file delete or overwrite (when running as SYSTEM or admin) can be used for a permanent system denial-of-service, e.g. by deleting a startup configuration file that prevents the service from starting.*	

Potential Mitigations

Phase: Architecture and Design

Strategy = Separation of Privilege

When designing software that will have different rights than the executer, the software should check that files that it is interacting with are not improper hard links or mount points. One way to do this in Windows is to use the functionality embedded in the following command: "dir /al /s /b" or, in PowerShell, use LinkType as a filter. In addition, some software uses authentication via signing to ensure that the file is the correct one to use. Make checks atomic with the file action, otherwise a TOCTOU weakness (CWE-367) can be introduced.

Observed Examples

Reference	Description
CVE-2021-26426	Privileged service allows attackers to delete unauthorized files using a directory junction, leading to arbitrary code execution as SYSTEM. *https://www.cve.org/CVERecord?id=CVE-2021-26426*
CVE-2020-0863	By creating a mount point and hard links, an attacker can abuse a service to allow users arbitrary file read permissions. *https://www.cve.org/CVERecord?id=CVE-2020-0863*
CVE-2019-1161	Chain: race condition (CWE-362) in anti-malware product allows deletion of files by creating a junction (CWE-1386) and using hard links during the time window in which a temporary file is created and deleted. *https://www.cve.org/CVERecord?id=CVE-2019-1161*
CVE-2014-0568	Escape from sandbox for document reader by using a mountpoint [REF-1264] *https://www.cve.org/CVERecord?id=CVE-2014-0568*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1416	Comprehensive Categorization: Resource Lifecycle Management	1400	2545

Notes

Terminology

Symbolic links, hard links, junctions, and mount points can be confusing terminology, as there are differences in how they operate between UNIX-based systems and Windows, and there are interactions between them.

Maintenance

This entry is still under development and will continue to see updates and content improvements.

References

[REF-1262]Eran Shimony. "Follow the Link: Exploiting Symbolic Links with Ease". 2019 October 3. < https://www.cyberark.com/resources/threat-research-blog/follow-the-link-exploiting-symbolic-links-with-ease >.

[REF-1264]James Forshaw. "Windows 10^H^H Symbolic Link Mitigations". 2015 August 5. < https://googleprojectzero.blogspot.com/2015/08/windows-10hh-symbolic-link-mitigations.html >.

[REF-1265]"Symbolic testing tools". < https://github.com/googleprojectzero/symboliclink-testing-tools >.

[REF-1266]Shubham Dubey. "Understanding and Exploiting Symbolic links in Windows - Symlink Attack EOP". 2020 April 6. < https://nixhacker.com/understanding-and-exploiting-symbolic-link-in-windows/ >.

[REF-1267]Simon Zuckerbraun. "Abusing Arbitrary File Deletes to Escalate Privilege and Other Great Tricks". 2022 March 7. < https://www.zerodayinitiative.com/blog/2022/3/16/abusing-arbitrary-file-deletes-to-escalate-privilege-and-other-great-tricks >.

[REF-1271]Clément Lavoillotte. "Abusing privileged file operations". 2019 March 0. < https://troopers.de/troopers19/agenda/7af9hw/ >.

CWE-1389: Incorrect Parsing of Numbers with Different Radices

Weakness ID : 1389
Structure : Simple
Abstraction : Base

Description

The product parses numeric input assuming base 10 (decimal) values, but it does not account for inputs that use a different base number (radix).

Extended Description

Frequently, a numeric input that begins with "0" is treated as octal, or "0x" causes it to be treated as hexadecimal, e.g. by the inet_addr() function. For example, "023" (octal) is 35 decimal, or "0x31" is 49 decimal. Other bases may be used as well. If the developer assumes decimal-only inputs, the code could produce incorrect numbers when the inputs are parsed using a different base. This can result in unexpected and/or dangerous behavior. For example, a "0127.0.0.1" IP address is parsed as octal due to the leading "0", whose numeric value would be the same as 87.0.0.1 (decimal), where the developer likely expected to use 127.0.0.1.

The consequences vary depending on the surrounding code in which this weakness occurs, but they can include bypassing network-based access control using unexpected IP addresses or netmasks, or causing apparently-symbolic identifiers to be processed as if they are numbers. In web applications, this can enable bypassing of SSRF restrictions.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf		704	Incorrect Type Conversion or Cast	1538

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name	Page
MemberOf	C	189	Numeric Errors	2312

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality	Read Application Data *An attacker may use an unexpected numerical base to access private application resources.*	Unknown
Integrity	Bypass Protection Mechanism Alter Execution Logic *An attacker may use an unexpected numerical base to bypass or manipulate access control mechanisms.*	Unknown

Potential Mitigations

Phase: Implementation

Strategy = Enforcement by Conversion

If only decimal-based values are expected in the application, conditional checks should be created in a way that prevent octal or hexadecimal strings from being checked. This can be achieved by converting any numerical string to an explicit base-10 integer prior to the conditional check, to prevent octal or hex values from ever being checked against the condition.

Phase: Implementation

Strategy = Input Validation

If various numerical bases do need to be supported, check for leading values indicating the non-decimal base you wish to support (such as 0x for hex) and convert the numeric strings to integers of the respective base. Reject any other alternative-base string that is not intentionally supported by the application.

Phase: Implementation

Strategy = Input Validation

If regular expressions are used to validate IP addresses, ensure that they are bounded using ^ and $ to prevent base-prepended IP addresses from being matched.

Demonstrative Examples

Example 1:

The below demonstrative example uses an IP validator that splits up an IP address by octet, tests to ensure each octet can be casted into an integer, and then returns the original IP address if no exceptions are raised. This validated IP address is then tested using the "ping" command.

Example Language: Python *(Bad)*

```
import subprocess
def validate_ip(ip: str):
  split_ip = ip.split('.')
  if len(split_ip) > 4 or len(split_ip) == 0:
    raise ValueError("Invalid IP length")
  for octet in split_ip:
    try:
      int(octet, 10)
    except ValueError as e:
      raise ValueError(f"Cannot convert IP octet to int - {e}")
  # Returns original IP after ensuring no exceptions are raised
  return ip
def run_ping(ip: str):
  validated = validate_ip(ip)
  # The ping command treats zero-prepended IP addresses as octal
  result = subprocess.call(["ping", validated])
  print(result)
```

If run_ping() were to be called with one or more zero-prepended octets, validate_ip() will succeed as zero-prepended numerical strings can be interpreted as decimal by a cast ("012" would cast to 12). However, as the original IP with the prepended zeroes is returned rather than the casted IP, it will be used in the call to the ping command. Ping DOES check and support octal-based IP octets, so the IP reached via ping may be different than the IP assumed by the validator. For example, ping would considered "0127.0.0.1" the same as "87.0.0.1".

Example 2:

This code uses a regular expression to validate an IP string prior to using it in a call to the "ping" command.

Example Language: Python *(Bad)*

```
import subprocess
import re
def validate_ip_regex(ip: str):
  ip_validator = re.compile(r"((25[0-5]|(2[0-4]|1\d|[1-9]|)\d)\.?\b){4}")
  if ip_validator.match(ip):
    return ip
  else:
    raise ValueError("IP address does not match valid pattern.")
def run_ping_regex(ip: str):
  validated = validate_ip_regex(ip)
  # The ping command treats zero-prepended IP addresses as octal
  result = subprocess.call(["ping", validated])
  print(result)
```

Since the regular expression does not have anchors (CWE-777), i.e. is unbounded without ^ or $ characters, then prepending a 0 or 0x to the beginning of the IP address will still result in a matched regex pattern. Since the ping command supports octal and hex prepended IP addresses, it will use the unexpectedly valid IP address (CWE-1389). For example, "0x63.63.63.63" would be considered equivalent to "99.63.63.63". As a result, the attacker could potentially ping systems that the attacker cannot reach directly.

Example 3:

Consider the following scenario, inspired by CWE team member Kelly Todd.

Kelly wants to set up monitoring systems for his two cats, who pose very different threats. One cat, Night, tweets embarrassing or critical comments about his owner in ways that could cause reputational damage, so Night's blog needs to be monitored regularly. The other cat, Taki, likes to distract Kelly and his coworkers during business meetings with cute meows, so Kelly monitors Taki's location using a different web site.

Suppose /etc/hosts provides the site info as follows:

Example Language: Other *(Bad)*

```
taki.example.com 10.1.0.7
night.example.com 010.1.0.8
```

The entry for night.example.com has a typo "010" in the first octet. When using ping to ensure the servers are up, the leading 0 causes the IP address to be converted using octal. So when Kelly's script attempts to access night.example.com, it inadvertently scans 8.1.0.8 instead of 10.1.0.8 (since "010" in octal is 8 in decimal), and Night is free to send new Tweets without being immediately detected.

Observed Examples

Reference	Description
CVE-2021-29662	Chain: Use of zero-prepended IP addresses in Perl-based IP validation module can lead to an access control bypass. *https://www.cve.org/CVERecord?id=CVE-2021-29662*
CVE-2021-28918	Chain: Use of zero-prepended IP addresses in a product that manages IP blocks can lead to an SSRF. *https://www.cve.org/CVERecord?id=CVE-2021-28918*
CVE-2021-29921	Chain: Use of zero-prepended IP addresses in a Python standard library package can lead to an SSRF. *https://www.cve.org/CVERecord?id=CVE-2021-29921*
CVE-2021-29923	Chain: Use of zero-prepended IP addresses in the net Golang library can lead to an access control bypass. *https://www.cve.org/CVERecord?id=CVE-2021-29923*
CVE-2021-29424	Chain: Use of zero-prepended IP addresses in Perl netmask module allows bypass of IP-based access control. *https://www.cve.org/CVERecord?id=CVE-2021-29424*
CVE-2016-4029	Chain: incorrect validation of intended decimal-based IP address format (CWE-1286) enables parsing of octal or hexadecimal formats (CWE-1389), allowing bypass of an SSRF protection mechanism (CWE-918). *https://www.cve.org/CVERecord?id=CVE-2016-4029*
CVE-2020-13776	Mishandling of hex-valued usernames leads to unexpected decimal conversion and privilege escalation in the systemd Linux suite. *https://www.cve.org/CVERecord?id=CVE-2020-13776*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1416	Comprehensive Categorization: Resource Lifecycle Management	1400	2545

References

[REF-1284]Sick Codes. "Universal "netmask" npm package, used by 270,000+ projects, vulnerable to octal input data". 2021 March 8. < https://sick.codes/universal-netmask-npm-package-used-

by-270000-projects-vulnerable-to-octal-input-data-server-side-request-forgery-remote-file-inclusion-local-file-inclusion-and-more-cve-2021-28918/ >.

CWE-1390: Weak Authentication

Weakness ID : 1390
Structure : Simple
Abstraction : Class

Description

The product uses an authentication mechanism to restrict access to specific users or identities, but the mechanism does not sufficiently prove that the claimed identity is correct.

Extended Description

Attackers may be able to bypass weak authentication faster and/or with less effort than expected.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	287	Improper Authentication	692
ParentOf	B	262	Not Using Password Aging	633
ParentOf	B	263	Password Aging with Long Expiration	636
ParentOf	B	289	Authentication Bypass by Alternate Name	703
ParentOf	B	290	Authentication Bypass by Spoofing	705
ParentOf	B	294	Authentication Bypass by Capture-replay	712
ParentOf	B	301	Reflection Attack in an Authentication Protocol	733
ParentOf	B	302	Authentication Bypass by Assumed-Immutable Data	735
ParentOf	B	303	Incorrect Implementation of Authentication Algorithm	737
ParentOf	B	305	Authentication Bypass by Primary Weakness	740
ParentOf	B	307	Improper Restriction of Excessive Authentication Attempts	747
ParentOf	B	308	Use of Single-factor Authentication	752
ParentOf	B	309	Use of Password System for Primary Authentication	754
ParentOf	C	522	Insufficiently Protected Credentials	1225
ParentOf	V	593	Authentication Bypass: OpenSSL CTX Object Modified after SSL Objects are Created	1331
ParentOf	B	603	Use of Client-Side Authentication	1354
ParentOf	B	620	Unverified Password Change	1383
ParentOf	B	640	Weak Password Recovery Mechanism for Forgotten Password	1409
ParentOf	B	804	Guessable CAPTCHA	1701
ParentOf	B	836	Use of Password Hash Instead of Password for Authentication	1761
ParentOf	C	1391	Use of Weak Credentials	2269

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Technology : ICS/OT *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Integrity Confidentiality Availability Access Control	Read Application Data Gain Privileges or Assume Identity Execute Unauthorized Code or Commands *This weakness can lead to the exposure of resources or functionality to unintended actors, possibly providing attackers with sensitive information or even execute arbitrary code.*	

Demonstrative Examples

Example 1:

In 2022, the OT:ICEFALL study examined products by 10 different Operational Technology (OT) vendors. The researchers reported 56 vulnerabilities and said that the products were "insecure by design" [REF-1283]. If exploited, these vulnerabilities often allowed adversaries to change how the products operated, ranging from denial of service to changing the code that the products executed. Since these products were often used in industries such as power, electrical, water, and others, there could even be safety implications.

Multiple OT products used weak authentication.

Observed Examples

Reference	Description
CVE-2022-30034	Chain: Web UI for a Python RPC framework does not use regex anchors to validate user login emails (CWE-777), potentially allowing bypass of OAuth (CWE-1390). *https://www.cve.org/CVERecord?id=CVE-2022-30034*
CVE-2022-35248	Chat application skips validation when Central Authentication Service (CAS) is enabled, effectively removing the second factor from two-factor authentication *https://www.cve.org/CVERecord?id=CVE-2022-35248*
CVE-2021-3116	Chain: Python-based HTTP Proxy server uses the wrong boolean operators (CWE-480) causing an incorrect comparison (CWE-697) that identifies an authN failure if all three conditions are met instead of only one, allowing bypass of the proxy authentication (CWE-1390) *https://www.cve.org/CVERecord?id=CVE-2021-3116*
CVE-2022-29965	Distributed Control System (DCS) uses a deterministic algorithm to generate utility passwords *https://www.cve.org/CVERecord?id=CVE-2022-29965*
CVE-2022-29959	Initialization file contains credentials that can be decoded using a "simple string transformation" *https://www.cve.org/CVERecord?id=CVE-2022-29959*
CVE-2020-8994	UART interface for AI speaker uses empty password for root shell *https://www.cve.org/CVERecord?id=CVE-2020-8994*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1396	Comprehensive Categorization: Access Control	1400	2519

References

[REF-1283]Forescout Vedere Labs. "OT:ICEFALL: The legacy of "insecure by design" and its implications for certifications and risk management". 2022 June 0. < https://www.forescout.com/resources/ot-icefall-report/ >.

CWE-1391: Use of Weak Credentials

Weakness ID : 1391
Structure : Simple
Abstraction : Class

Description

The product uses weak credentials (such as a default key or hard-coded password) that can be calculated, derived, reused, or guessed by an attacker.

Extended Description

By design, authentication protocols try to ensure that attackers must perform brute force attacks if they do not know the credentials such as a key or password. However, when these credentials are easily predictable or even fixed (as with default or hard-coded passwords and keys), then the attacker can defeat the mechanism without relying on brute force.

Credentials may be weak for different reasons, such as:

- Hard-coded (i.e., static and unchangeable by the administrator)
- Default (i.e., the same static value across different deployments/installations, but able to be changed by the administrator)
- Predictable (i.e., generated in a way that produces unique credentials across deployments/installations, but can still be guessed with reasonable efficiency)

Even if a new, unique credential is intended to be generated for each product installation, if the generation is predictable, then that may also simplify guessing attacks.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	1390	Weak Authentication	2267
ParentOf	B	521	Weak Password Requirements	1223
ParentOf	B	798	Use of Hard-coded Credentials	1690
ParentOf	B	1392	Use of Default Credentials	2271

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : ICS/OT *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Demonstrative Examples

Example 1:

In 2022, the OT:ICEFALL study examined products by 10 different Operational Technology (OT) vendors. The researchers reported 56 vulnerabilities and said that the products were "insecure by design" [REF-1283]. If exploited, these vulnerabilities often allowed adversaries to change how the products operated, ranging from denial of service to changing the code that the products executed. Since these products were often used in industries such as power, electrical, water, and others, there could even be safety implications.

Multiple OT products used weak credentials.

Observed Examples

Reference	Description
[REF-1374]	Chain: JavaScript-based cryptocurrency library can fall back to the insecure Math.random() function instead of reporting a failure (CWE-392), thus reducing the entropy (CWE-332) and leading to generation of non-unique cryptographic keys for Bitcoin wallets (CWE-1391) *https://www.unciphered.com/blog/randstorm-you-cant-patch-a-house-of-cards*
CVE-2022-30270	Remote Terminal Unit (RTU) uses default credentials for some SSH accounts *https://www.cve.org/CVERecord?id=CVE-2022-30270*
CVE-2022-29965	Distributed Control System (DCS) uses a deterministic algorithm to generate utility passwords *https://www.cve.org/CVERecord?id=CVE-2022-29965*
CVE-2022-30271	Remote Terminal Unit (RTU) uses a hard-coded SSH private key that is likely to be used in typical deployments *https://www.cve.org/CVERecord?id=CVE-2022-30271*
CVE-2021-38759	microcontroller board has default password, allowing admin access *https://www.cve.org/CVERecord?id=CVE-2021-38759*
CVE-2021-41192	data visualization/sharing package uses default secret keys or cookie values if they are not specified in environment variables *https://www.cve.org/CVERecord?id=CVE-2021-41192*
CVE-2020-8994	UART interface for AI speaker uses empty password for root shell *https://www.cve.org/CVERecord?id=CVE-2020-8994*
CVE-2020-27020	password manager does not generate cryptographically strong passwords, allowing prediction of passwords using guessable details such as time of generation *https://www.cve.org/CVERecord?id=CVE-2020-27020*
CVE-2020-8632	password generator for cloud application has small length value, making it easier for brute-force guessing *https://www.cve.org/CVERecord?id=CVE-2020-8632*
CVE-2020-5365	network-attached storage (NAS) system has predictable default passwords for a diagnostics/support account *https://www.cve.org/CVERecord?id=CVE-2020-5365*
CVE-2020-5248	IT asset management app has a default encryption key that is the same across installations *https://www.cve.org/CVERecord?id=CVE-2020-5248*
CVE-2012-3503	Installation script has a hard-coded secret token value, allowing attackers to bypass authentication *https://www.cve.org/CVERecord?id=CVE-2012-3503*
CVE-2010-2306	Intrusion Detection System (IDS) uses the same static, private SSL keys for multiple devices and installations, allowing decryption of SSL traffic *https://www.cve.org/CVERecord?id=CVE-2010-2306*
CVE-2001-0618	Residential gateway uses the last 5 digits of the 'Network Name' or SSID as the default WEP key, which allows attackers to get the key by sniffing the SSID, which is sent in the clear *https://www.cve.org/CVERecord?id=CVE-2001-0618*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1396	Comprehensive Categorization: Access Control	1400	2519

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
ISA/IEC 62443	Part 2-4		Req SP.09.02 RE(1)
ISA/IEC 62443	Part 4-1		Req SR-3 b)
ISA/IEC 62443	Part 4-1		Req SI-2 b)
ISA/IEC 62443	Part 4-1		Req SI-2 d)
ISA/IEC 62443	Part 4-1		Req SG-3 d)
ISA/IEC 62443	Part 4-1		Req SG-6 b)
ISA/IEC 62443	Part 4-2		Req CR 1.1
ISA/IEC 62443	Part 4-2		Req CR 1.2
ISA/IEC 62443	Part 4-2		Req CR 1.5
ISA/IEC 62443	Part 4-2		Req CR 1.7
ISA/IEC 62443	Part 4-2		Req CR 1.8
ISA/IEC 62443	Part 4-2		Req CR 1.9
ISA/IEC 62443	Part 4-2		Req CR 1.14
ISA/IEC 62443	Part 4-2		Req CR 2.1
ISA/IEC 62443	Part 4-2		Req CR 4.3
ISA/IEC 62443	Part 4-2		Req CR 7.5

References

[REF-1303]Kelly Jackson Higgins. "Researchers Out Default Passwords Packaged With ICS/ SCADA Wares". 2016 January 4. < https://www.darkreading.com/endpoint/researchers-out-default-passwords-packaged-with-ics-scada-wares >.2022-10-11.

[REF-1304]ICS-CERT. "ICS Alert (ICS-ALERT-13-164-01): Medical Devices Hard-Coded Passwords". 2013 June 3. < https://www.cisa.gov/news-events/ics-alerts/ics-alert-13-164-01 >.2023-04-07.

[REF-1283]Forescout Vedere Labs. "OT:ICEFALL: The legacy of "insecure by design" and its implications for certifications and risk management". 2022 June 0. < https://www.forescout.com/ resources/ot-icefall-report/ >.

[REF-1374]Unciphered. "Randstorm: You Can't Patch a House of Cards". 2023 November 4. < https://www.unciphered.com/blog/randstorm-you-cant-patch-a-house-of-cards >.2023-11-15.

CWE-1392: Use of Default Credentials

Weakness ID : 1392
Structure : Simple
Abstraction : Base

Description

The product uses default credentials (such as passwords or cryptographic keys) for potentially critical functionality.

Extended Description

It is common practice for products to be designed to use default keys, passwords, or other mechanisms for authentication. The rationale is to simplify the manufacturing process or the system administrator's task of installation and deployment into an enterprise. However, if admins do not change the defaults, it is easier for attackers to bypass authentication quickly across multiple organizations.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	1391	Use of Weak Credentials	2269
ParentOf	B	1393	Use of Default Password	2273
ParentOf	B	1394	Use of Default Cryptographic Key	2275

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name	Page
MemberOf	C	255	Credentials Management Errors	2315

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : ICS/OT *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Authentication	Gain Privileges or Assume Identity	

Potential Mitigations

Phase: Requirements

Prohibit use of default, hard-coded, or other values that do not vary for each installation of the product - especially for separate organizations.

Effectiveness = High

Phase: Architecture and Design

Force the administrator to change the credential upon installation.

Effectiveness = High

Phase: Installation

Phase: Operation

The product administrator could change the defaults upon installation or during operation.

Effectiveness = Moderate

Demonstrative Examples

Example 1:

In 2022, the OT:ICEFALL study examined products by 10 different Operational Technology (OT) vendors. The researchers reported 56 vulnerabilities and said that the products were "insecure by design" [REF-1283]. If exploited, these vulnerabilities often allowed adversaries to change how the products operated, ranging from denial of service to changing the code that the products executed.

Since these products were often used in industries such as power, electrical, water, and others, there could even be safety implications.

At least one OT product used default credentials.

Observed Examples

Reference	Description
CVE-2022-30270	Remote Terminal Unit (RTU) uses default credentials for some SSH accounts *https://www.cve.org/CVERecord?id=CVE-2022-30270*
CVE-2021-41192	data visualization/sharing package uses default secret keys or cookie values if they are not specified in environment variables *https://www.cve.org/CVERecord?id=CVE-2021-41192*
CVE-2021-38759	microcontroller board has default password *https://www.cve.org/CVERecord?id=CVE-2021-38759*
CVE-2010-2306	Intrusion Detection System (IDS) uses the same static, private SSL keys for multiple devices and installations, allowing decryption of SSL traffic *https://www.cve.org/CVERecord?id=CVE-2010-2306*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1396	Comprehensive Categorization: Access Control	1400	2519

References

[REF-1283]Forescout Vedere Labs. "OT:ICEFALL: The legacy of "insecure by design" and its implications for certifications and risk management". 2022 June 0. < https://www.forescout.com/resources/ot-icefall-report/ >.

CWE-1393: Use of Default Password

Weakness ID : 1393
Structure : Simple
Abstraction : Base

Description

The product uses default passwords for potentially critical functionality.

Extended Description

It is common practice for products to be designed to use default passwords for authentication. The rationale is to simplify the manufacturing process or the system administrator's task of installation and deployment into an enterprise. However, if admins do not change the defaults, then it makes it easier for attackers to quickly bypass authentication across multiple organizations. There are many lists of default passwords and default-password scanning tools that are easily available from the World Wide Web.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	B	1392	Use of Default Credentials	2271

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Technology : ICS/OT *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Authentication	Gain Privileges or Assume Identity	

Potential Mitigations

Phase: Requirements

Prohibit use of default, hard-coded, or other values that do not vary for each installation of the product - especially for separate organizations.

Effectiveness = High

Phase: Documentation

Ensure that product documentation clearly emphasizes the presence of default passwords and provides steps for the administrator to change them.

Effectiveness = Limited

Phase: Architecture and Design

Force the administrator to change the credential upon installation.

Effectiveness = High

Phase: Installation

Phase: Operation

The product administrator could change the defaults upon installation or during operation.

Effectiveness = Moderate

Demonstrative Examples

Example 1:

In 2022, the OT:ICEFALL study examined products by 10 different Operational Technology (OT) vendors. The researchers reported 56 vulnerabilities and said that the products were "insecure by design" [REF-1283]. If exploited, these vulnerabilities often allowed adversaries to change how the products operated, ranging from denial of service to changing the code that the products executed. Since these products were often used in industries such as power, electrical, water, and others, there could even be safety implications.

Multiple OT products used default credentials.

Observed Examples

Reference	Description
CVE-2022-30270	Remote Terminal Unit (RTU) uses default credentials for some SSH accounts *https://www.cve.org/CVERecord?id=CVE-2022-30270*
CVE-2022-2336	OPC Unified Architecture (OPC UA) industrial automation product has a default password *https://www.cve.org/CVERecord?id=CVE-2022-2336*

Reference	Description
CVE-2021-38759	microcontroller board has default password *https://www.cve.org/CVERecord?id=CVE-2021-38759*
CVE-2021-44480	children's smart watch has default passwords allowing attackers to send SMS commands and listen to the device's surroundings *https://www.cve.org/CVERecord?id=CVE-2021-44480*
CVE-2020-11624	surveillance camera has default password for the admin account *https://www.cve.org/CVERecord?id=CVE-2020-11624*
CVE-2018-15719	medical dental records product installs a MySQL database with a blank default password *https://www.cve.org/CVERecord?id=CVE-2018-15719*
CVE-2014-9736	healthcare system for archiving patient images has default passwords for key management and storage databases *https://www.cve.org/CVERecord?id=CVE-2014-9736*
CVE-2000-1209	database product installs admin account with default null password, allowing privileges, as exploited by various worms *https://www.cve.org/CVERecord?id=CVE-2000-1209*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1364	ICS Communications: Zone Boundary Failures	1358	2501
MemberOf	C	1366	ICS Communications: Frail Security in Protocols	1358	2503
MemberOf	C	1368	ICS Dependencies (& Architecture): External Digital Systems	1358	2505
MemberOf	C	1376	ICS Engineering (Construction/Deployment): Security Gaps in Commissioning	1358	2512
MemberOf	C	1396	Comprehensive Categorization: Access Control	1400	2519

References

[REF-1283]Forescout Vedere Labs. "OT:ICEFALL: The legacy of "insecure by design" and its implications for certifications and risk management". 2022 June 0. < https://www.forescout.com/resources/ot-icefall-report/ >.

[REF-1303]Kelly Jackson Higgins. "Researchers Out Default Passwords Packaged With ICS/SCADA Wares". 2016 January 4. < https://www.darkreading.com/endpoint/researchers-out-default-passwords-packaged-with-ics-scada-wares >.2022-10-11.

CWE-1394: Use of Default Cryptographic Key

Weakness ID : 1394
Structure : Simple
Abstraction : Base

Description

The product uses a default cryptographic key for potentially critical functionality.

Extended Description

It is common practice for products to be designed to use default keys. The rationale is to simplify the manufacturing process or the system administrator's task of installation and deployment into an enterprise. However, if admins do not change the defaults, it is easier for attackers to bypass authentication quickly across multiple organizations.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	B	1392	Use of Default Credentials	2271

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Authentication	Gain Privileges or Assume Identity	

Potential Mitigations

Phase: Requirements

Prohibit use of default, hard-coded, or other values that do not vary for each installation of the product - especially for separate organizations.

Effectiveness = High

Phase: Architecture and Design

Force the administrator to change the credential upon installation.

Effectiveness = High

Phase: Installation

Phase: Operation

The product administrator could change the defaults upon installation or during operation.

Effectiveness = Moderate

Observed Examples

Reference	Description
CVE-2018-3825	cloud cluster management product has a default master encryption key *https://www.cve.org/CVERecord?id=CVE-2018-3825*
CVE-2016-1561	backup storage product has a default SSH public key in the authorized_keys file, allowing root access *https://www.cve.org/CVERecord?id=CVE-2016-1561*
CVE-2010-2306	Intrusion Detection System (IDS) uses the same static, private SSL keys for multiple devices and installations, allowing decryption of SSL traffic *https://www.cve.org/CVERecord?id=CVE-2010-2306*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1396	Comprehensive Categorization: Access Control	1400	2519

CWE-1395: Dependency on Vulnerable Third-Party Component

Weakness ID : 1395
Structure : Simple
Abstraction : Class

Description

The product has a dependency on a third-party component that contains one or more known vulnerabilities.

Extended Description

Many products are large enough or complex enough that part of their functionality uses libraries, modules, or other intellectual property developed by third parties who are not the product creator. For example, even an entire operating system might be from a third-party supplier in some hardware products. Whether open or closed source, these components may contain publicly known vulnerabilities that could be exploited by adversaries to compromise the product.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	657	Violation of Secure Design Principles	1446

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality Integrity Availability	Varies by Context *The consequences vary widely, depending on the vulnerabilities that exist in the component; how those vulnerabilities can be "reached" by adversaries, as the exploitation paths and attack surface will vary depending on how the component is used; and the criticality of the privilege levels and features for which the product relies on the component.*	

Detection Methods

Automated Analysis

For software, use Software Composition Analysis (SCA) tools, which automatically analyze products to identify third-party dependencies. Often, SCA tools can be used to link with known vulnerabilities in the dependencies that they detect. There are commercial and open-source alternatives, such as OWASP Dependency-Check [REF-1312]. Many languages or frameworks have package managers with similar capabilities, such as npm audit for JavaScript, pip-audit for Python, govulncheck for Go, and many others. Dynamic methods can detect loading of third-party components.

Effectiveness = High

Software Composition Analysis (SCA) tools face a number of technical challenges that can lead to false positives and false negatives. Dynamic methods have other technical challenges.

Potential Mitigations

Phase: Requirements

Phase: Policy

In some industries such as healthcare [REF-1320] [REF-1322] or technologies such as the cloud [REF-1321], it might be unclear about who is responsible for applying patches for third-party vulnerabilities: the vendor, the operator/customer, or a separate service. Clarifying roles and responsibilities can be important to minimize confusion or unnecessary delay when third-party vulnerabilities are disclosed.

Phase: Requirements

Require a Bill of Materials for all components and sub-components of the product. For software, require a Software Bill of Materials (SBOM) [REF-1247] [REF-1311].

Phase: Architecture and Design

Phase: Implementation

Phase: Integration

Phase: Manufacturing

Maintain a Bill of Materials for all components and sub-components of the product. For software, maintain a Software Bill of Materials (SBOM). According to [REF-1247], "An SBOM is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships."

Phase: Operation

Phase: Patching and Maintenance

Actively monitor when a third-party component vendor announces vulnerability patches; fix the third-party component as soon as possible; and make it easy for operators/customers to obtain and apply the patch.

Phase: Operation

Phase: Patching and Maintenance

Continuously monitor changes in each of the product's components, especially when the changes indicate new vulnerabilities, end-of-life (EOL) plans, etc.

Demonstrative Examples

Example 1:

The "SweynTooth" vulnerabilities in Bluetooth Low Energy (BLE) software development kits (SDK) were found to affect multiple Bluetooth System-on-Chip (SoC) manufacturers. These SoCs were used by many products such as medical devices, Smart Home devices, wearables, and other IoT devices. [REF-1314] [REF-1315]

Example 2:

log4j, a Java-based logging framework, is used in a large number of products, with estimates in the range of 3 billion affected devices [REF-1317]. When the "log4shell" (CVE-2021-44228) vulnerability was initially announced, it was actively exploited for remote code execution, requiring urgent mitigation in many organizations. However, it was unclear how many products were affected, as Log4j would sometimes be part of a long sequence of transitive dependencies. [REF-1316]

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1418	Comprehensive Categorization: Violation of Secure Design Principles	1400	2549

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
ISA/IEC 62443	Part 4-2		Req CR 2.4
ISA/IEC 62443	Part 4-2		Req CR 6.2
ISA/IEC 62443	Part 4-2		Req CR 7.2
ISA/IEC 62443	Part 4-1		Req SM-9
ISA/IEC 62443	Part 4-1		Req SM-10
ISA/IEC 62443	Part 4-1		Req SR-2
ISA/IEC 62443	Part 4-1		Req DM-1
ISA/IEC 62443	Part 4-1		Req DM-3
ISA/IEC 62443	Part 4-1		Req DM-4
ISA/IEC 62443	Part 4-1		Req SVV-1
ISA/IEC 62443	Part 4-1		Req SVV-3

References

[REF-1313]Jeff Williams, Arshan Dabirsiaghi. "The Unfortunate Reality of Insecure Libraries". 2014. < https://owasp.org/www-project-dependency-check/ >.2023-01-25.

[REF-1212]"A06:2021 - Vulnerable and Outdated Components". 2021 September 4. OWASP. < https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/ >.

[REF-1247]NTIA Multistakeholder Process on Software Component Transparency Framing Working Group. "Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM)". 2021 October 1. < https://www.ntia.gov/files/ntia/publications/ntia_sbom_framing_2nd_edition_20211021.pdf >.

[REF-1311]Amélie Koran, Wendy Nather, Stewart Scott, Sara Ann Brackett. "The Cases for Using the SBOMs We Build". 2022 November. < https://www.atlanticcouncil.org/wp-content/uploads/2022/11/AC_SBOM_IB_v2-002.pdf >.2023-01-25.

[REF-1312]OWASP. "OWASP Dependency-Check". < https://owasp.org/www-project-dependency-check/ >.2023-01-25.

[REF-1314]ICS-CERT. "ICS Alert (ICS-ALERT-20-063-01): SweynTooth Vulnerabilities". 2020 March 4. < https://www.cisa.gov/news-events/ics-alerts/ics-alert-20-063-01 >.2023-04-07.

[REF-1315]Matheus E. Garbelini, Sudipta Chattopadhyay, Chundong Wang, Singapore University of Technology and Design. "Unleashing Mayhem over Bluetooth Low Energy". 2020 March 4. < https://asset-group.github.io/disclosures/sweyntooth/ >.2023-01-25.

[REF-1316]CISA. "Alert (AA21-356A): Mitigating Log4Shell and Other Log4j-Related Vulnerabilities". 2021 December 2. < https://www.cisa.gov/news-events/cybersecurity-advisories/aa21-356a >.2023-04-07.

[REF-1317]SC Media. "What Log4Shell taught us about application security, and how to respond now". 2022 July 5. < https://www.scmagazine.com/resource/application-security/what-log4shell-taught-us-about-appsec-and-how-to-respond >.2023-01-26.

[REF-1320]Ali Youssef. "A Framework for a Medical Device Security Program at a Healthcare Delivery Organization". 2022 August 8. < https://array.aami.org/content/news/framework-medical-device-security-program-healthcare-delivery-organization >.2023-04-07.

[REF-1321]Cloud Security Alliance. "Shared Responsibility Model Explained". 2020 August 6. < https://cloudsecurityalliance.org/blog/2020/08/26/shared-responsibility-model-explained/ >.2023-01-28.

[REF-1322]Melissa Chase, Steven Christey Coley, Julie Connolly, Ronnie Daldos, Margie Zuk. "Medical Device Cybersecurity Regional Incident Preparedness and Response Playbook". 2022 November 4. < https://www.mitre.org/news-insights/publication/medical-device-cybersecurity-regional-incident-preparedness-and-response >.2023-01-28.

CWE-1419: Incorrect Initialization of Resource

Weakness ID : 1419
Structure : Simple
Abstraction : Class

Description

The product attempts to initialize a resource but does not correctly do so, which might leave the resource in an unexpected, incorrect, or insecure state when it is accessed.

Extended Description

This can have security implications when the associated resource is expected to have certain properties or values. Examples include a variable that determines whether a user has been authenticated or not, or a register or fuse value that determines the security state of the product.

For software, this weakness can frequently occur when implicit initialization is used, meaning the resource is not explicitly set to a specific value. For example, in C, memory is not necessarily cleared when it is allocated on the stack, and many scripting languages use a default empty, null value, or zero value when a variable is not explicitly initialized.

For hardware, this weakness frequently appears with reset values and fuses. After a product reset, hardware may initialize registers incorrectly. During different phases of a product lifecycle, fuses may be set to incorrect values. Even if fuses are set to correct values, the lines to the fuse could be broken or there might be hardware on the fuse line that alters the fuse value to be incorrect.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	665	Improper Initialization	1456
ParentOf	B	454	External Initialization of Trusted Variables or Data Stores	1085
ParentOf	B	1051	Initialization with Hard-Coded Network Resource Configuration Data	1886
ParentOf	B	1052	Excessive Use of Hard-Coded Literals in Initialization	1887
ParentOf	B	1188	Initialization of a Resource with an Insecure Default	1974
ParentOf	B	1221	Incorrect Register Defaults or Module Parameters	1996

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality	Read Memory Read Application Data Unexpected State	Unknown
Authorization Integrity	Gain Privileges or Assume Identity	
Other	Varies by Context *The technical impact can vary widely based on how the resource is used in the product, and whether its contents affect security decisions.*	

Potential Mitigations

Phase: Implementation

Choose the safest-possible initialization for security-related resources.

Phase: Implementation

Ensure that each resource (whether variable, memory buffer, register, etc.) is fully initialized.

Phase: Implementation

Pay close attention to complex conditionals or reset sources that affect initialization, since some paths might not perform the initialization.

Phase: Architecture and Design

Ensure that the design and architecture clearly identify what the initialization should be, and that the initialization does not have security implications.

Demonstrative Examples

Example 1:

Consider example design module system verilog code shown below. The register_example module is an example parameterized module that defines two parameters, REGISTER_WIDTH and REGISTER_DEFAULT. Register_example module defines a Secure_mode setting, which when set makes the register content read-only and not modifiable by software writes. register_top module instantiates two registers, Insecure_Device_ID_1 and Insecure_Device_ID_2. Generally, registers containing device identifier values are required to be read only to prevent any possibility of software modifying these values.

Example Language: Verilog *(Bad)*

```
// Parameterized Register module example
// Secure_mode : REGISTER_DEFAULT[0] : When set to 1 register is read only and not writable//
module register_example
#(
parameter REGISTER_WIDTH = 8, // Parameter defines width of register, default 8 bits
parameter [REGISTER_WIDTH-1:0] REGISTER_DEFAULT = 2**REGISTER_WIDTH -2 // Default value of register
computed from Width. Sets all bits to 1s except bit 0 (Secure _mode)
)
(
input [REGISTER_WIDTH-1:0] Data_in,
input Clk,
input resetn,
```

```
input write,
output reg [REGISTER_WIDTH-1:0] Data_out
);
reg Secure_mode;
always @(posedge Clk or negedge resetn)
   if (~resetn)
   begin
      Data_out <= REGISTER_DEFAULT; // Register content set to Default at reset
      Secure_mode <= REGISTER_DEFAULT[0]; // Register Secure_mode set at reset
   end
   else if (write & ~Secure_mode)
   begin
      Data_out <= Data_in;
   end
endmodule
module register_top
(
input Clk,
input resetn,
input write,
input [31:0] Data_in,
output reg [31:0] Secure_reg,
output reg [31:0] Insecure_reg
);
register_example #(
   .REGISTER_WIDTH (32),
   .REGISTER_DEFAULT (1224) // Incorrect Default value used bit 0 is 0.
) Insecure_Device_ID_1 (
   .Data_in (Data_in),
   .Data_out (Secure_reg),
   .Clk (Clk),
   .resetn (resetn),
   .write (write)
);
register_example #(
   .REGISTER_WIDTH (32) // Default not defined 2^32-2 value will be used as default.
) Insecure_Device_ID_2 (
   .Data_in (Data_in),
   .Data_out (Insecure_reg),
   .Clk (Clk),
   .resetn (resetn),
   .write (write)
);
endmodule
```

These example instantiations show how, in a hardware design, it would be possible to instantiate the register module with insecure defaults and parameters.

In the example design, both registers will be software writable since Secure_mode is defined as zero.

Example Language: Verilog *(Good)*

```
register_example #(
   .REGISTER_WIDTH (32),
   .REGISTER_DEFAULT (1225) // Correct default value set, to enable Secure_mode
) Secure_Device_ID_example (
   .Data_in (Data_in),
   .Data_out (Secure_reg),
   .Clk (Clk),
   .resetn (resetn),
   .write (write)
);
```

Example 2:

This code attempts to login a user using credentials from a POST request:

Example Language: PHP *(Bad)*

```
// $user and $pass automatically set from POST request
if (login_user($user,$pass)) {
  $authorized = true;
}
...
if ($authorized) {
  generatePage();
}
```

Because the $authorized variable is never initialized, PHP will automatically set $authorized to any value included in the POST request if register_globals is enabled. An attacker can send a POST request with an unexpected third value 'authorized' set to 'true' and gain authorized status without supplying valid credentials.

Here is a fixed version:

Example Language: PHP *(Good)*

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$authorized = false;
if (login_user($user,$pass)) {
  $authorized = true;
}
...
```

This code avoids the issue by initializing the $authorized variable to false and explicitly retrieving the login credentials from the $_POST variable. Regardless, register_globals should never be enabled and is disabled by default in current versions of PHP.

Observed Examples

Reference	Description
CVE-2020-27211	Chain: microcontroller system-on-chip uses a register value stored in flash to set product protection state on the memory bus and does not contain protection against fault injection (CWE-1319) which leads to an incorrect initialization of the memory bus (CWE-1419) causing the product to be in an unprotected state. *https://www.cve.org/CVERecord?id=CVE-2020-27211*
CVE-2023-25815	chain: a change in an underlying package causes the gettext function to use implicit initialization with a hard-coded path (CWE-1419) under the user-writable C:\ drive, introducing an untrusted search path element (CWE-427) that enables spoofing of messages. *https://www.cve.org/CVERecord?id=CVE-2023-25815*
CVE-2022-43468	WordPress module sets internal variables based on external inputs, allowing false reporting of the number of views *https://www.cve.org/CVERecord?id=CVE-2022-43468*
CVE-2022-36349	insecure default variable initialization in BIOS firmware for a hardware board allows DoS *https://www.cve.org/CVERecord?id=CVE-2022-36349*
CVE-2015-7763	distributed filesystem only initializes part of the variable-length padding for a packet, allowing attackers to read sensitive information from previously-sent packets in the same memory location *https://www.cve.org/CVERecord?id=CVE-2015-7763*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1416	Comprehensive Categorization: Resource Lifecycle Management	1400	2545

CWE-1420: Exposure of Sensitive Information during Transient Execution

Weakness ID : 1420
Structure : Simple
Abstraction : Base

Description

A processor event or prediction may allow incorrect operations (or correct operations with incorrect data) to execute transiently, potentially exposing data over a covert channel.

Extended Description

When operations execute but do not commit to the processor's architectural state, this is commonly referred to as transient execution. This behavior can occur when the processor mis-predicts an outcome (such as a branch target), or when a processor event (such as an exception or microcode assist, etc.) is handled after younger operations have already executed. Operations that execute transiently may exhibit observable discrepancies (CWE-203) in covert channels [REF-1400] such as data caches. Observable discrepancies of this kind can be detected and analyzed using timing or power analysis techniques, which may allow an attacker to infer information about the operations that executed transiently. For example, the attacker may be able to infer confidential data that was accessed or used by those operations.

Transient execution weaknesses may be exploited using one of two methods. In the first method, the attacker generates a code sequence that exposes data through a covert channel when it is executed transiently (the attacker must also be able to trigger transient execution). Some transient execution weaknesses can only expose data that is accessible within the attacker's processor context. For example, an attacker executing code in a software sandbox may be able to use a transient execution weakness to expose data within the same address space, but outside of the attacker's sandbox. Other transient execution weaknesses can expose data that is architecturally inaccessible, that is, data protected by hardware-enforced boundaries such as page tables or privilege rings. These weaknesses are the subject of CWE-1421.

In the second exploitation method, the attacker first identifies a code sequence in a victim program that, when executed transiently, can expose data that is architecturally accessible within the victim's processor context. For instance, the attacker may search the victim program for code sequences that resemble a bounds-check bypass sequence (see Demonstrative Example 1). If the attacker can trigger a mis-prediction of the conditional branch and influence the index of the out-of-bounds array access, then the attacker may be able to infer the value of out-of-bounds data by monitoring observable discrepancies in a covert channel.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	C	669	Incorrect Resource Transfer Between Spheres	1471
ParentOf	B	1421	Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution	2290
ParentOf	B	1422	Exposure of Sensitive Information caused by Incorrect Data Forwarding during Transient Execution	2297
ParentOf	B	1423	Exposure of Sensitive Information caused by Shared Microarchitectural Predictor State that Influences Transient Execution	2302

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name	Page
ParentOf	B	1421	Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution	2290
ParentOf	B	1422	Exposure of Sensitive Information caused by Incorrect Data Forwarding during Transient Execution	2297
ParentOf	B	1423	Exposure of Sensitive Information caused by Shared Microarchitectural Predictor State that Influences Transient Execution	2302

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality	Read Memory	Medium

Detection Methods

Manual Analysis

This weakness can be detected in hardware by manually inspecting processor specifications. Features that exhibit this weakness may include microarchitectural predictors, access control checks that occur out-of-order, or any other features that can allow operations to execute without committing to architectural state. Academic researchers have demonstrated that new hardware weaknesses can be discovered by exhaustively analyzing a processor's machine clear (or nuke) conditions ([REF-1427]).

Effectiveness = Moderate

Hardware designers can also scrutinize aspects of the instruction set architecture that have undefined behavior; these can become a focal point when applying other detection methods. Manual analysis may not reveal all weaknesses in a processor specification and should be combined with other detection methods to improve coverage.

Fuzzing

Academic researchers have demonstrated that this weakness can be detected in hardware using software fuzzing tools that treat the underlying hardware as a black box ([REF-1428]).

Effectiveness = Opportunistic

Fuzzing may not reveal all weaknesses in a processor specification and should be combined with other detection methods to improve coverage.

Fuzzing

Academic researchers have demonstrated that this weakness can be detected in software using software fuzzing tools ([REF-1429]).

Effectiveness = Opportunistic

At the time of this writing, publicly available software fuzzing tools can only detect a subset of transient execution weaknesses in software (for example, [REF-1429] can only detect instances of Spectre v1) and may produce false positives.

Automated Static Analysis

A variety of automated static analysis tools can identify potentially exploitable code sequences in software. These tools may perform the analysis on source code, on binary code, or on an intermediate code representation (for example, during compilation).

Effectiveness = Limited

At the time of this writing, publicly available software static analysis tools can only detect a subset of transient execution weaknesses in software and may produce false positives.

Automated Analysis

Software vendors can release tools that detect presence of known weaknesses on a processor. For example, some of these tools can attempt to transiently execute a vulnerable code sequence and detect whether code successfully leaks data in a manner consistent with the weakness under test. Alternatively, some hardware vendors provide enumeration for the presence of a weakness (or lack of a weakness). These enumeration bits can be checked and reported by system software. For example, Linux supports these checks for many commodity processors: $ cat /proc/cpuinfo | grep bugs | head -n 1 bugs : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa itlb_multihit srbds mmio_stale_data retbleed

Effectiveness = High

This method can be useful for detecting whether a processor is affected by known weaknesses, but it may not be useful for detecting unknown weaknesses.

Potential Mitigations

Phase: Architecture and Design

The hardware designer can attempt to prevent transient execution from causing observable discrepancies in specific covert channels.

Effectiveness = Limited

This technique has many pitfalls. For example, InvisiSpec was an early attempt to mitigate this weakness by blocking "micro-architectural covert and side channels through the multiprocessor data cache hierarchy due to speculative loads" [REF-1417]. Commodity processors and SoCs have many covert and side channels that exist outside of the data cache hierarchy. Even when some of these channels are blocked, others (such as execution ports [REF-1418]) may allow an attacker to infer confidential data. Mitigation strategies that attempt to prevent transient execution from causing observable discrepancies also have other pitfalls, for example, see [REF-1419].

Phase: Requirements

Processor designers may expose instructions or other architectural features that allow software to mitigate the effects of transient execution, but without disabling predictors. These features may also help to limit opportunities for data exposure.

Effectiveness = Moderate

Instructions or features that constrain transient execution or suppress its side effects may impact performance.

Phase: Requirements

Processor designers may expose registers (for example, control registers or model-specific registers) that allow privileged and/or user software to disable specific predictors or other hardware features that can cause confidential data to be exposed during transient execution.

Effectiveness = Limited

Disabling specific predictors or other hardware features may result in significant performance overhead.

Phase: Requirements

Processor designers, system software vendors, or other agents may choose to restrict the ability of unprivileged software to access to high-resolution timers that are commonly used to monitor covert channels.

Effectiveness = Defense in Depth

Specific software algorithms can be used by an attacker to compensate for a lack of a high-resolution time source [REF-1420].

Phase: Build and Compilation

Isolate sandboxes or managed runtimes in separate address spaces (separate processes). For examples, see [REF-1421].

Effectiveness = High

Phase: Build and Compilation

Include serialization instructions (for example, LFENCE) that prevent processor events or mis-predictions prior to the serialization instruction from causing transient execution after the serialization instruction. For some weaknesses, a serialization instruction can also prevent a processor event or a mis-prediction from occurring after the serialization instruction (for example, CVE-2018-3639 can allow a processor to predict that a load will not depend on an older store; a serialization instruction between the store and the load may allow the store to update memory and prevent the prediction from happening at all).

Effectiveness = Moderate

When used to comprehensively mitigate a transient execution weakness (for example, by inserting an LFENCE after every instruction in a program), serialization instructions can introduce significant performance overhead. On the other hand, when used to mitigate only a relatively small number of high-risk code sequences, serialization instructions may have a low or negligible impact on performance.

Phase: Build and Compilation

Use control-flow integrity (CFI) techniques to constrain the behavior of instructions that redirect the instruction pointer, such as indirect branch instructions.

Effectiveness = Moderate

Some CFI techniques may not be able to constrain transient execution, even though they are effective at constraining architectural execution. Or they may be able to provide some additional protection against a transient execution weakness, but without comprehensively mitigating the weakness. For example, Clang-CFI provides strong architectural CFI properties and can make some transient execution weaknesses more difficult to exploit [REF-1398].

Phase: Build and Compilation

If the weakness is exposed by a single instruction (or a small set of instructions), then the compiler (or JIT, etc.) can be configured to prevent the affected instruction(s) from being generated, and instead generate an alternate sequence of instructions that is not affected by the weakness. One prominent example of this mitigation is retpoline ([REF-1414]).

Effectiveness = Limited

This technique may only be effective for software that is compiled with this mitigation. For some transient execution weaknesses, this technique may not be sufficient to protect software that is compiled without the affected instruction(s). For example, see CWE-1421.

Phase: Build and Compilation

Use software techniques that can mitigate the consequences of transient execution. For example, address masking can be used in some circumstances to prevent out-of-bounds transient reads.

Effectiveness = Limited

Address masking and related software mitigation techniques have been used to harden specific code sequences that could potentially be exploited via transient execution. For example, the Linux kernel makes limited use of manually inserted address masks to mitigate bounds-check bypass [REF-1390]. Compiler-based techniques have also been used to automatically harden software [REF-1425].

Phase: Build and Compilation

Use software techniques (including the use of serialization instructions) that are intended to reduce the number of instructions that can be executed transiently after a processor event or misprediction.

Effectiveness = Incidental

Some transient execution weaknesses can be exploited even if a single instruction is executed transiently after a processor event or mis-prediction. This mitigation strategy has many other pitfalls that prevent it from eliminating this weakness entirely. For example, see [REF-1389].

Phase: Documentation

If a hardware feature can allow incorrect operations (or correct operations with incorrect data) to execute transiently, the hardware designer may opt to disclose this behavior in architecture documentation. This documentation can inform users about potential consequences and effective mitigations.

Effectiveness = High

Demonstrative Examples

Example 1:

Secure programs perform bounds checking before accessing an array if the source of the array index is provided by an untrusted source such as user input. In the code below, data from array1 will not be accessed if x is out of bounds. The following code snippet is from [REF-1415]:

Example Language: C *(Bad)*

```
if (x < array1_size)
  y = array2[array1[x] * 4096];
```

However, if this code executes on a processor that performs conditional branch prediction the outcome of the if statement could be mis-predicted and the access on the next line will occur with a value of x that can point to an out-of-bounds location (within the program's memory).

Even though the processor does not commit the architectural effects of the mis-predicted branch, the memory accesses alter data cache state, which is not rolled back after the branch is resolved. The cache state can reveal array1[x] thereby providing a mechanism to recover the data value located at address array1 + x.

Example 2:

Some managed runtimes or just-in-time (JIT) compilers may overwrite recently executed code with new code. When the instruction pointer enters the new code, the processor may inadvertently execute the stale code that had been overwritten. This can happen, for instance, when the processor issues a store that overwrites a sequence of code, but the processor fetches and executes the (stale) code before the store updates memory. Similar to the first example, the processor does not commit the stale code's architectural effects, though microarchitectural side effects can persist. Hence, confidential information accessed or used by the stale code may be inferred via an observable discrepancy in a covert channel. This vulnerability is described in more detail in [REF-1427].

Observed Examples

Reference	Description
CVE-2017-5753	Microarchitectural conditional branch predictors may allow operations to execute transiently after a misprediction, potentially exposing data over a covert channel. *https://www.cve.org/CVERecord?id=CVE-2017-5753*
CVE-2021-0089	A machine clear triggered by self-modifying code may allow incorrect operations to execute transiently, potentially exposing data over a covert channel. *https://www.cve.org/CVERecord?id=CVE-2021-0089*
CVE-2022-0002	Microarchitectural indirect branch predictors may allow incorrect operations to execute transiently after a misprediction, potentially exposing data over a covert channel. *https://www.cve.org/CVERecord?id=CVE-2022-0002*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1198	Privilege Separation and Access Control Issues	1194	2470
MemberOf	C	1201	Core and Compute Issues	1194	2471
MemberOf	C	1202	Memory and Storage Issues	1194	2472
MemberOf	C	1416	Comprehensive Categorization: Resource Lifecycle Management	1400	2545

References

[REF-1389]Alyssa Milburn, Ke Sun and Henrique Kawakami. "You Cannot Always Win the Race: Analyzing the LFENCE/JMP Mitigation for Branch Target Injection". 2022 March 8. < https://arxiv.org/abs/2203.04277 >.2024-02-22.

[REF-1417]Mengjia Yan, Jiho Choi, Dimitrios Skarlatos, Adam Morrison, Christopher W. Fletcher and Josep Torrella. "InvisiSpec: making speculative execution invisible in the cache hierarchy.". 2019 May. < http://iacoma.cs.uiuc.edu/iacoma-papers/micro18.pdf >.2024-02-14.

[REF-1418]Alejandro Cabrera Aldaya, Billy Bob Brumley, Sohaib ul Hassan, Cesar Pereida García and Nicola Tuveri. "Port Contention for Fun and Profit". 2019 May. < https://eprint.iacr.org/2018/1060.pdf >.2024-02-14.

[REF-1419]Mohammad Behnia, Prateek Sahu, Riccardo Paccagnella, Jiyong Yu, Zirui Zhao, Xiang Zou, Thomas Unterluggauer, Josep Torrellas, Carlos Rozas, Adam Morrison, Frank Mckeen, Fangfei Liu, Ron Gabor, Christopher W. Fletcher, Abhishek Basak and Alaa Alameldeen. "Speculative Interference Attacks: Breaking Invisible Speculation Schemes". 2021 April. < https://arxiv.org/abs/2007.11818 >.2024-02-14.

[REF-1420]Ross Mcilroy, Jaroslav Sevcik, Tobias Tebbi, Ben L. Titzer and Toon Verwaest. "Spectre is here to stay: An analysis of side-channels and speculative execution". 2019 February 4. < https://arxiv.org/pdf/1902.05178.pdf >.2024-02-14.

[REF-1421]Intel Corporation. "Managed Runtime Speculative Execution Side Channel Mitigations". 2018 January 3. < https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/technical-documentation/runtime-speculative-side-channel-mitigations.html >.2024-02-14.

[REF-1398]The Clang Team. "Control Flow Integrity". < https://clang.llvm.org/docs/ControlFlowIntegrity.html >.2024-02-13.

[REF-1414]Intel Corporation. "Retpoline: A Branch Target Injection Mitigation". 2022 August 2. < https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/technical-documentation/retpoline-branch-target-injection-mitigation.html >.2023-02-13.

[REF-1390]The kernel development community. "Speculation". 2020 August 6. < https://docs.kernel.org/6.6/staging/speculation.html >.2024-02-04.

[REF-1425]Chandler Carruth. "Speculative Load Hardening". < https://llvm.org/docs/SpeculativeLoadHardening.html >.2024-02-14.

[REF-1427]Hany Ragab, Enrico Barberis, Herbert Bos and Cristiano Giuffrida. "Rage Against the Machine Clear: A Systematic Analysis of Machine Clears and Their Implications for Transient Execution Attacks". 2021 August. < https://www.usenix.org/system/files/sec21-ragab.pdf >.2024-02-14.

[REF-1428]Oleksii Oleksenko, Marco Guarnieri, Boris Köpf and Mark Silberstein. "Hide and Seek with Spectres: Efficient discovery of speculative information leaks with random testing". 2023 January 8. < https://arxiv.org/pdf/2301.07642.pdf >.2024-02-14.

[REF-1429]Oleksii Oleksenko, Bohdan Trach, Mark Silberstein and Christof Fetzer. "SpecFuzz: Bringing Spectre-type vulnerabilities to the surface". 2020 August. < https://www.usenix.org/system/files/sec20-oleksenko.pdf >.2024-02-14.

[REF-1415]Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz and Yuval Yarom. "Spectre Attacks: Exploiting Speculative Execution". 2019 May. < https://spectreattack.com/spectre.pdf >.2024-02-14.

[REF-1400]Intel Corporation. "Refined Speculative Execution Terminology". 2022 March 1. < https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/best-practices/refined-speculative-execution-terminology.html >.2024-02-13.

CWE-1421: Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution

Weakness ID : 1421
Structure : Simple
Abstraction : Base

Description

A processor event may allow transient operations to access architecturally restricted data (for example, in another address space) in a shared microarchitectural structure (for example, a CPU cache), potentially exposing the data over a covert channel.

Extended Description

Many commodity processors have Instruction Set Architecture (ISA) features that protect software components from one another. These features can include memory segmentation, virtual memory, privilege rings, trusted execution environments, and virtual machines, among others. For example, virtual memory provides each process with its own address space, which prevents processes from accessing each other's private data. Many of these features can be used to form hardware-enforced security boundaries between software components.

Many commodity processors also share microarchitectural resources that cache (temporarily store) data, which may be confidential. These resources may be shared across processor contexts, including across SMT threads, privilege rings, or others.

When transient operations allow access to ISA-protected data in a shared microarchitectural resource, this might violate users' expectations of the ISA feature that is bypassed. For example, if transient operations can access a victim's private data in a shared microarchitectural resource, then the operations' microarchitectural side effects may correspond to the accessed data. If an attacker can trigger these transient operations and observe their side effects through a covert channel [REF-1400], then the attacker may be able to infer the victim's private data. Private data could include sensitive program data, OS/VMM data, page table data (such as memory addresses), system configuration data (see Demonstrative Example 3), or any other data that the attacker does not have the required privileges to access.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	B	1420	Exposure of Sensitive Information during Transient Execution	2284

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name	Page
ChildOf	B	1420	Exposure of Sensitive Information during Transient Execution	2284

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality	Read Memory *<<put the information here>>*	Medium

Detection Methods

Manual Analysis

This weakness can be detected in hardware by manually inspecting processor specifications. Features that exhibit this weakness may include microarchitectural predictors, access control checks that occur out-of-order, or any other features that can allow operations to execute without committing to architectural state. Academic researchers have demonstrated that new hardware weaknesses can be discovered by examining publicly available patent filings, for example [REF-1405] and [REF-1406]. Hardware designers can also scrutinize aspects of the instruction set architecture that have undefined behavior; these can become a focal point when applying other detection methods.

Effectiveness = Moderate

Manual analysis may not reveal all weaknesses in a processor specification and should be combined with other detection methods to improve coverage.

Automated Analysis

This weakness can be detected (pre-discovery) in hardware by employing static or dynamic taint analysis methods [REF-1401]. These methods can label data in one context (for example, kernel data) and perform information flow analysis (or a simulation, etc.) to determine whether tainted data can appear in another context (for example, user mode). Alternatively, stale or invalid data in shared microarchitectural resources can be marked as tainted, and the taint analysis framework can identify when transient operations encounter tainted data.

Effectiveness = Moderate

Automated static or dynamic taint analysis may not reveal all weaknesses in a processor specification and should be combined with other detection methods to improve coverage.

Automated Analysis

Software vendors can release tools that detect presence of known weaknesses (post-discovery) on a processor. For example, some of these tools can attempt to transiently execute a vulnerable code sequence and detect whether code successfully leaks data in a manner consistent with the weakness under test. Alternatively, some hardware vendors provide enumeration for the presence of a weakness (or lack of a weakness). These enumeration bits can be checked and reported by system software. For example, Linux supports these checks for many commodity processors: $ cat /proc/cpuinfo | grep bugs | head -n 1 bugs : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa itlb_multihit srbds mmio_stale_data retbleed

Effectiveness = High

This method can be useful for detecting whether a processor if affected by known weaknesses, but it may not be useful for detecting unknown weaknesses.

Fuzzing

Academic researchers have demonstrated that this weakness can be detected in hardware using software fuzzing tools that treat the underlying hardware as a black box ([REF-1406], [REF-1430])

Effectiveness = Opportunistic

Fuzzing may not reveal all weaknesses in a processor specification and should be combined with other detection methods to improve coverage.

Potential Mitigations

Phase: Architecture and Design

Hardware designers may choose to engineer the processor's pipeline to prevent architecturally restricted data from being used by operations that can execute transiently.

Effectiveness = High

Phase: Architecture and Design

Hardware designers may choose not to share microarchitectural resources that can contain sensitive data, such as fill buffers and store buffers.

Effectiveness = Moderate

This can be highly effective at preventing this weakness from being exposed across different SMT threads or different processor cores. It is generally less practical to isolate these resources between different contexts (for example, user and kernel) that may execute on the same SMT thread or processor core.

Phase: Architecture and Design

Hardware designers may choose to sanitize specific microarchitectural state (for example, store buffers) when the processor transitions to a different context, such as whenever a system call is invoked. Alternatively, the hardware may expose instruction(s) that allow software to sanitize microarchitectural state according to the user or system administrator's threat model. These mitigation approaches are similar to those that address CWE-226; however, sanitizing microarchitectural state may not be the optimal or best way to mitigate this weakness on every processor design.

Effectiveness = Moderate

Sanitizing shared state on context transitions may not be practical for all processors, especially when the amount of shared state affected by the weakness is relatively large. Additionally, this technique may not be practical unless there is a synchronous transition between two processor contexts that would allow the affected resource to be sanitized. For example, this technique alone may not suffice to mitigate asynchronous access to a resource that is shared by two SMT threads.

Phase: Architecture and Design

The hardware designer can attempt to prevent transient execution from causing observable discrepancies in specific covert channels.

Effectiveness = Limited

This technique has many pitfalls. For example, InvisiSpec was an early attempt to mitigate this weakness by blocking "micro-architectural covert and side channels through the multiprocessor data cache hierarchy due to speculative loads" [REF-1417]. Commodity processors and SoCs have many covert and side channels that exist outside of the data cache hierarchy. Even when some of these channels are blocked, others (such as execution ports [REF-1418]) may allow an attacker to infer confidential data. Mitigation strategies that attempt to prevent transient execution from causing observable discrepancies also have other pitfalls, for example, see [REF-1419].

Phase: Architecture and Design

Software architects may design software to enforce strong isolation between different contexts. For example, kernel page table isolation (KPTI) mitigates the Meltdown vulnerability [REF-1401] by separating user-mode page tables from kernel-mode page tables, which prevents user-mode processes from using Meltdown to transiently access kernel memory [REF-1404].

Effectiveness = Limited

Isolating different contexts across a process boundary (or another kind of architectural boundary) may only be effective for some weaknesses.

Phase: Build and Compilation

If the weakness is exposed by a single instruction (or a small set of instructions), then the compiler (or JIT, etc.) can be configured to prevent the affected instruction(s) from being generated, and instead generate an alternate sequence of instructions that is not affected by the weakness.

Effectiveness = Limited

This technique may only be fully effective if it is applied to all software that runs on the system. Also, relatively few observed examples of this weakness have exposed data through only a single instruction.

Phase: Build and Compilation

Use software techniques (including the use of serialization instructions) that are intended to reduce the number of instructions that can be executed transiently after a processor event or misprediction.

Effectiveness = Incidental

Some transient execution weaknesses can be exploited even if a single instruction is executed transiently after a processor event or mis-prediction. This mitigation strategy has many other pitfalls that prevent it from eliminating this weakness entirely. For example, see [REF-1389].

Phase: Implementation

System software can mitigate this weakness by invoking state-sanitizing operations when switching from one context to another, according to the hardware vendor's recommendations.

Effectiveness = Limited

This technique may not be able to mitigate weaknesses that arise from resource sharing across SMT threads.

Phase: System Configuration

Some systems may allow the user to disable (for example, in the BIOS) sharing of the affected resource.

Effectiveness = Limited

Disabling resource sharing (for example, by disabling SMT) may result in significant performance overhead.

Phase: System Configuration

Some systems may allow the user to disable (for example, in the BIOS) microarchitectural features that allow transient access to architecturally restricted data.

Effectiveness = Limited

Disabling microarchitectural features such as predictors may result in significant performance overhead.

Phase: Patching and Maintenance

The hardware vendor may provide a patch to sanitize the affected shared microarchitectural state when the processor transitions to a different context.

Effectiveness = Moderate

This technique may not be able to mitigate weaknesses that arise from resource sharing across SMT threads.

Phase: Patching and Maintenance

This kind of patch may not be feasible or implementable for all processors or all weaknesses.

Effectiveness = Limited

Phase: Requirements

Processor designers, system software vendors, or other agents may choose to restrict the ability of unprivileged software to access to high-resolution timers that are commonly used to monitor covert channels.

Effectiveness = Defense in Depth

Specific software algorithms can be used by an attacker to compensate for a lack of a high-resolution time source [REF-1420].

Demonstrative Examples

Example 1:

Some processors may perform access control checks in parallel with memory read/write operations. For example, when a user-mode program attempts to read data from memory, the processor may also need to check whether the memory address is mapped into user space or kernel space. If the processor performs the access concurrently with the check, then the access may be able to transiently read kernel data before the check completes. This race condition is demonstrated in the following code snippet from [REF-1408], with additional annotations:

Example Language: x86 Assembly *(Bad)*

```
1 ; rcx = kernel address, rbx = probe array
2 xor rax, rax # set rax to 0
3 retry:
4 mov al, byte [rcx] # attempt to read kernel memory
5 shl rax, 0xc # multiply result by page size (4KB)
6 jz retry # if the result is zero, try again
7 mov rbx, qword [rbx + rax] # transmit result over a cache covert channel
```

Vulnerable processors may return kernel data from a shared microarchitectural resource in line 4, for example, from the processor's L1 data cache. Since this vulnerability involves a race condition, the mov in line 4 may not always return kernel data (that is, whenever the check "wins" the race), in which case this demonstration code re-attempts the access in line 6. The accessed data is multiplied by 4KB, a common page size, to make it easier to observe via a cache covert channel after the transmission in line 7. The use of cache covert channels to observe the side effects of transient execution has been described in [REF-1408].

Example 2:

Many commodity processors share microarchitectural fill buffers between sibling hardware threads on simultaneous multithreaded (SMT) processors. Fill buffers can serve as temporary storage for data that passes to and from the processor's caches. Microarchitectural Fill Buffer Data Sampling (MFBDS) is a vulnerability that can allow a hardware thread to access its sibling's private data in a shared fill buffer. The access may be prohibited by the processor's ISA, but MFBDS can allow the access to occur during transient execution, in particular during a faulting operation or an operation that triggers a microcode assist.

More information on MFBDS can be found in [REF-1405] and [REF-1409].

Example 3:

Some processors may allow access to system registers (for example, system coprocessor registers or model-specific registers) during transient execution. This scenario is depicted in the code snippet below. Under ordinary operating circumstances, code in exception level 0 (EL0) is not permitted to access registers that are restricted to EL1, such as TTBR0_EL1. However, on some processors an earlier mis-prediction can cause the MRS instruction to transiently read the value in an EL1 register. In this example, a conditional branch (line 2) can be mis-predicted as "not taken" while waiting for a slow load (line 1). This allows MRS (line 3) to transiently read the value in the TTBR0_EL1 register. The subsequent memory access (line 6) can allow the restricted register's value to become observable, for example, over a cache covert channel.

Code snippet is from [REF-1410]. See also [REF-1411].

Example Language: x86 Assembly *(Bad)*

```
1 LDR X1, [X2] ; arranged to miss in the cache
2 CBZ X1, over ; This will be taken
3 MRS X3, TTBR0_EL1;
4 LSL X3, X3, #imm
5 AND X3, X3, #0xFC0
6 LDR X5, [X6,X3] ; X6 is an EL0 base address
7 over
```

Observed Examples

Reference	Description
CVE-2017-5715	A fault may allow transient user-mode operations to access kernel data cached in the L1D, potentially exposing the data over a covert channel. *https://www.cve.org/CVERecord?id=CVE-2017-5715*
CVE-2018-3615	A fault may allow transient non-enclave operations to access SGX enclave data cached in the L1D, potentially exposing the data over a covert channel. *https://www.cve.org/CVERecord?id=CVE-2018-3615*
CVE-2019-1135	A TSX Asynchronous Abort may allow transient operations to access architecturally restricted data, potentially exposing the data over a covert channel. *https://www.cve.org/CVERecord?id=CVE-2019-1135*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1416	Comprehensive Categorization: Resource Lifecycle Management	1400	2545

References

[REF-1404]The kernel development community. "Page Table Isolation (PTI)". 2023 January 0. < https://kernel.org/doc/html/next/x86/pti.html >.2024-02-13.

[REF-1405]Stephan van Schaik, Alyssa Milburn, Sebastian Österlund, Pietro Frigo, Giorgi Maisuradze, Kaveh Razavi, Herbert Bos and Cristiano Giuffrida. "RIDL: Rogue In-Flight Data Load". 2019 May 9. < https://mdsattacks.com/files/ridl.pdf >.2024-02-13.

[REF-1406]Daniel Moghimi. "Downfall: Exploiting Speculative Data Gathering". 2023 August 9. < https://www.usenix.org/system/files/usenixsecurity23-moghimi.pdf >.2024-02-13.

[REF-1401]Neta Bar Kama and Roope Kaivola. "Hardware Security Leak Detection by Symbolic Simulation". 2021 November. < https://ieeexplore.ieee.org/document/9617727 >.2024-02-13.

[REF-1408]Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom and Mike Hamburg. "Meltdown: Reading Kernel Memory from User Space". 2020 May 1. < https://meltdownattack.com/meltdown.pdf >.2024-02-13.

[REF-1409]Intel Corporation. "Microarchitectural Data Sampling". 2021 March 1. < https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/technical-documentation/intel-analysis-microarchitectural-data-sampling.html >.2024-02-13.

[REF-1410]ARM. "Cache Speculation Side-channels". 2018 January. < https://armkeil.blob.core.windows.net/developer/Files/pdf/Cache_Speculation_Side-channels.pdf >.2024-02-22.

[REF-1411]Intel Corporation. "Rogue System Register Read/CVE-2018-3640/INTEL-SA-00115". 2018 May 1. < https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/rogue-system-register-read.html >.2024-02-13.

[REF-1400]Intel Corporation. "Refined Speculative Execution Terminology". 2022 March 1. < https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/best-practices/refined-speculative-execution-terminology.html >.2024-02-13.

[REF-1389]Alyssa Milburn, Ke Sun and Henrique Kawakami. "You Cannot Always Win the Race: Analyzing the LFENCE/JMP Mitigation for Branch Target Injection". 2022 March 8. < https://arxiv.org/abs/2203.04277 >.2024-02-22.

[REF-1430]Daniel Moghimi, Moritz Lipp, Berk Sunar and Michael Schwarz. "Medusa: Microarchitectural: Data Leakage via Automated Attack Synthesis". 2020 August. < https://www.usenix.org/conference/usenixsecurity20/presentation/moghimi-medusa >.2024-02-27.

[REF-1417]Mengjia Yan, Jiho Choi, Dimitrios Skarlatos, Adam Morrison, Christopher W. Fletcher and Josep Torrella. "InvisiSpec: making speculative execution invisible in the cache hierarchy.". 2019 May. < http://iacoma.cs.uiuc.edu/iacoma-papers/micro18.pdf >.2024-02-14.

[REF-1418]Alejandro Cabrera Aldaya, Billy Bob Brumley, Sohaib ul Hassan, Cesar Pereida García and Nicola Tuveri. "Port Contention for Fun and Profit". 2019 May. < https://eprint.iacr.org/2018/1060.pdf >.2024-02-14.

[REF-1419]Mohammad Behnia, Prateek Sahu, Riccardo Paccagnella, Jiyong Yu, Zirui Zhao, Xiang Zou, Thomas Unterluggauer, Josep Torrellas, Carlos Rozas, Adam Morrison, Frank Mckeen, Fangfei Liu, Ron Gabor, Christopher W. Fletcher, Abhishek Basak and Alaa Alameldeen. "Speculative Interference Attacks: Breaking Invisible Speculation Schemes". 2021 April. < https://arxiv.org/abs/2007.11818 >.2024-02-14.

[REF-1420]Ross Mcilroy, Jaroslav Sevcik, Tobias Tebbi, Ben L. Titzer and Toon Verwaest. "Spectre is here to stay: An analysis of side-channels and speculative execution". 2019 February 4. < https://arxiv.org/pdf/1902.05178.pdf >.2024-02-14.

CWE-1422: Exposure of Sensitive Information caused by Incorrect Data Forwarding during Transient Execution

Weakness ID : 1422
Structure : Simple
Abstraction : Base

Description

A processor event or prediction may allow incorrect or stale data to be forwarded to transient operations, potentially exposing data over a covert channel.

Extended Description

Software may use a variety of techniques to preserve the confidentiality of private data that is accessible within the current processor context. For example, the memory safety and type safety properties of some high-level programming languages help to prevent software written in those languages from exposing private data. As a second example, software sandboxes may co-locate multiple users' software within a single process. The processor's Instruction Set Architecture (ISA) may permit one user's software to access another user's data (because the software shares the same address space), but the sandbox prevents these accesses by using software techniques such as bounds checking.

If incorrect or stale data can be forwarded (for example, from a cache) to transient operations, then the operations' microarchitectural side effects may correspond to the data. If an attacker can trigger these transient operations and observe their side effects through a covert channel, then the attacker may be able to infer the data. For example, an attacker process may induce transient execution in a victim process that causes the victim to inadvertently access and then expose its private data via a covert channel. In the software sandbox example, an attacker sandbox may induce transient execution in its own code, allowing it to transiently access and expose data in a victim sandbox that shares the same address space.

Consequently, weaknesses that arise from incorrect/stale data forwarding might violate users' expectations of software-based memory safety and isolation techniques. If the data forwarding behavior is not properly documented by the hardware vendor, this might violate the software vendor's expectation of how the hardware should behave.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	B	1420	Exposure of Sensitive Information during Transient Execution	2284

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name	Page
ChildOf	B	1420	Exposure of Sensitive Information during Transient Execution	2284

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Not Technology-Specific *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality	Read Memory	Medium

Detection Methods

Automated Static Analysis

A variety of automated static analysis tools can identify potentially exploitable code sequences in software. These tools may perform the analysis on source code, on binary code, or on an intermediate code representation (for example, during compilation).

Effectiveness = Moderate

Automated static analysis may not reveal all weaknesses in a processor specification and should be combined with other detection methods to improve coverage.

Manual Analysis

This weakness can be detected in hardware by manually inspecting processor specifications. Features that exhibit this weakness may include microarchitectural predictors, access control checks that occur out-of-order, or any other features that can allow operations to execute without committing to architectural state.Hardware designers can also scrutinize aspects of the instruction set architecture that have undefined behavior; these can become a focal point when applying other detection methods.

Effectiveness = Moderate

Manual analysis may not reveal all weaknesses in a processor specification and should be combined with other detection methods to improve coverage.

Automated Analysis

Software vendors can release tools that detect presence of known weaknesses on a processor. For example, some of these tools can attempt to transiently execute a vulnerable code sequence and detect whether code successfully leaks data in a manner consistent with the weakness under test. Alternatively, some hardware vendors provide enumeration for the presence of a weakness (or lack of a weakness). These enumeration bits can be checked and reported by system software. For example, Linux supports these checks for many commodity processors: $ cat /proc/cpuinfo | grep bugs | head -n 1 bugs : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa itlb_multihit srbds mmio_stale_data retbleed

Effectiveness = High

This method can be useful for detecting whether a processor if affected by known weaknesses, but it may not be useful for detecting unknown weaknesses.

Potential Mitigations

Phase: Architecture and Design

The hardware designer can attempt to prevent transient execution from causing observable discrepancies in specific covert channels.

Effectiveness = Limited

Instructions or features that constrain transient execution or suppress its side effects may impact performance.

Phase: Requirements

Processor designers, system software vendors, or other agents may choose to restrict the ability of unprivileged software to access to high-resolution timers that are commonly used to monitor covert channels.

Effectiveness = Defense in Depth

Disabling specific predictors or other hardware features may result in significant performance overhead.

Phase: Requirements

Processor designers may expose instructions or other architectural features that allow software to mitigate the effects of transient execution, but without disabling predictors. These features may also help to limit opportunities for data exposure.

Effectiveness = Moderate

Instructions or features that constrain transient execution or suppress its side effects may impact performance.

Phase: Requirements

Processor designers may expose registers (for example, control registers or model-specific registers) that allow privileged and/or user software to disable specific predictors or other hardware features that can cause confidential data to be exposed during transient execution.

Effectiveness = Limited

Disabling specific predictors or other hardware features may result in significant performance overhead.

Phase: Build and Compilation

Use software techniques (including the use of serialization instructions) that are intended to reduce the number of instructions that can be executed transiently after a processor event or misprediction.

Effectiveness = Incidental

Some transient execution weaknesses can be exploited even if a single instruction is executed transiently after a processor event or mis-prediction. This mitigation strategy has many other pitfalls that prevent it from eliminating this weakness entirely. For example, see [REF-1389].

Phase: Build and Compilation

Isolate sandboxes or managed runtimes in separate address spaces (separate processes).

Effectiveness = High

Process isolation is also an effective strategy to mitigate many other kinds of weaknesses.

Phase: Build and Compilation

Include serialization instructions (for example, LFENCE) that prevent processor events or mis-predictions prior to the serialization instruction from causing transient execution after the serialization instruction. For some weaknesses, a serialization instruction can also prevent a processor event or a mis-prediction from occurring after the serialization instruction (for example, CVE-2018-3639 can allow a processor to predict that a load will not depend on an older store; a serialization instruction between the store and the load may allow the store to update memory and prevent the mis-prediction from happening at all).

Effectiveness = Moderate

When used to comprehensively mitigate a transient execution weakness, serialization instructions can introduce significant performance overhead.

Phase: Build and Compilation

Use software techniques that can mitigate the consequences of transient execution. For example, address masking can be used in some circumstances to prevent out-of-bounds transient reads.

Effectiveness = Limited

Address masking and related software mitigation techniques have been used to harden specific code sequences that could potentially be exploited via transient execution. For example, the Linux kernel makes limited use of this technique to mitigate bounds-check bypass [REF-1390].

Phase: Build and Compilation

If the weakness is exposed by a single instruction (or a small set of instructions), then the compiler (or JIT, etc.) can be configured to prevent the affected instruction(s) from being generated, and instead generate an alternate sequence of instructions that is not affected by the weakness.

Effectiveness = Limited

This technique is only effective for software that is compiled with this mitigation.

Phase: Documentation

If a hardware feature can allow incorrect or stale data to be forwarded to transient operations, the hardware designer may opt to disclose this behavior in architecture documentation. This documentation can inform users about potential consequences and effective mitigations.

Effectiveness = High

Demonstrative Examples

Example 1:

Faulting loads in a victim domain may trigger incorrect transient forwarding, which leaves secret-dependent traces in the microarchitectural state. Consider this code sequence example from [REF-1391].

Example Language: C *(Bad)*

```
void call_victim(size_t untrusted_arg) {
  *arg_copy = untrusted_arg;
  array[**trusted_ptr * 4096];
}
```

A processor with this weakness will store the value of untrusted_arg (which may be provided by an attacker) to the stack, which is trusted memory. Additionally, this store operation will save this value in some microarchitectural buffer, for example, the store buffer.

In this code sequence, trusted_ptr is dereferenced while the attacker forces a page fault. The faulting load causes the processor to mis-speculate by forwarding untrusted_arg as the (transient) load result. The processor then uses untrusted_arg for the pointer dereference. After the fault has been handled and the load has been re-issued with the correct argument, secret-dependent information stored at the address of trusted_ptr remains in microarchitectural state and can be extracted by an attacker using a vulnerable code sequence.

Example 2:

Some processors try to predict when a store will forward data to a subsequent load, even when the address of the store or the load is not yet known. For example, on Intel processors this feature is called a Fast Store Forwarding Predictor [REF-1392], and on AMD processors the feature is called Predictive Store Forwarding [REF-1393]. A misprediction can cause incorrect or stale data to be forwarded from a store to a load, as illustrated in the following code snippet from [REF-1393]:

Example Language: C *(Bad)*

```
void fn(int idx) {
  unsigned char v;
  idx_array[0] = 4096;
  v = array[idx_array[idx] * (idx)];
}
```

In this example, assume that the parameter idx can only be 0 or 1, and assume that idx_array initially contains all 0s. Observe that the assignment to v in line 4 will be array[0], regardless of whether idx=0 or idx=1. Now suppose that an attacker repeatedly invokes fn with idx=0 to train the store forwarding predictor to predict that the store in line 3 will forward the data 4096 to the load idx_array[idx] in line 4. Then, when the attacker invokes fn with idx=1 the predictor may cause idx_array[idx] to transiently produce the incorrect value 4096, and therefore v will transiently be assigned the value array[4096], which otherwise would not have been accessible in line 4.

Although this toy example is benign (it doesn't transmit array[4096] over a covert channel), an attacker may be able to use similar techniques to craft and train malicious code sequences to, for example, read data beyond a software sandbox boundary.

Observed Examples

Reference	Description
CVE-2020-0551	A fault, microcode assist, or abort may allow transient load operations to forward malicious stale data to dependent operations executed by a victim, causing the victim to unintentionally access and potentially expose its own data over a covert channel. *https://www.cve.org/CVERecord?id=CVE-2020-0551*
CVE-2020-8698	A fast store forwarding predictor may allow store operations to forward incorrect data to transient load operations, potentially exposing data over a covert channel. *https://www.cve.org/CVERecord?id=CVE-2020-8698*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1416	Comprehensive Categorization: Resource Lifecycle Management	1400	2545

References

[REF-1389]Alyssa Milburn, Ke Sun and Henrique Kawakami. "You Cannot Always Win the Race: Analyzing the LFENCE/JMP Mitigation for Branch Target Injection". 2022 March 8. < https://arxiv.org/abs/2203.04277 >.2024-02-22.

[REF-1390]The kernel development community. "Speculation". 2020 August 6. < https://docs.kernel.org/6.6/staging/speculation.html >.2024-02-04.

[REF-1391]Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yuval Yarom, Berk Sunar, Daniel Gruss and Frank Piessens. "LVI : Hijacking Transient Execution through Microarchitectural Load Value Injection". 2020 January 9. < https://lviattack.eu/lvi.pdf >.2024-02-04.

[REF-1392]Intel Corporation. "Fast Store Forwarding Predictor". 2022 February 8. < https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/technical-documentation/fast-store-forwarding-predictor.html >.2024-02-04.

[REF-1393]AMD. "Security Analysis Of AMD Predictive Store Forwarding". 2021 March. < https://www.amd.com/system/files/documents/security-analysis-predictive-store-forwarding.pdf >.2024-02-04.

CWE-1423: Exposure of Sensitive Information caused by Shared Microarchitectural Predictor State that Influences Transient Execution

Weakness ID : 1423
Structure : Simple
Abstraction : Base

Description

Shared microarchitectural predictor state may allow code to influence transient execution across a hardware boundary, potentially exposing data that is accessible beyond the boundary over a covert channel.

Extended Description

Many commodity processors have Instruction Set Architecture (ISA) features that protect software components from one another. These features can include memory segmentation, virtual memory, privilege rings, trusted execution environments, and virtual machines, among others. For example, virtual memory provides each process with its own address space, which prevents processes from accessing each other's private data. Many of these features can be used to form hardware-enforced security boundaries between software components.

When separate software components (for example, two processes) share microarchitectural predictor state across a hardware boundary, code in one component may be able to influence microarchitectural predictor behavior in another component. If the predictor can cause transient execution, the shared predictor state may allow an attacker to influence transient execution in a victim, and in a manner that could allow the attacker to infer private data from the victim by monitoring observable discrepancies (CWE-203) in a covert channel [REF-1400].

Predictor state may be shared when the processor transitions from one component to another (for example, when a process makes a system call to enter the kernel). Many commodity processors have features which prevent microarchitectural predictions that occur before a boundary from influencing predictions that occur after the boundary.

Predictor state may also be shared between hardware threads, for example, sibling hardware threads on a processor that supports simultaneous multithreading (SMT). This sharing may be benign if the hardware threads are simultaneously executing in the same software component, or it could expose a weakness if one sibling is a malicious software component, and the other sibling is a victim software component. Processors that share microarchitectural predictors between hardware threads may have features which prevent microarchitectural predictions that occur on one hardware thread from influencing predictions that occur on another hardware thread.

Features that restrict predictor state sharing across transitions or between hardware threads may be always-on, on by default, or may require opt-in from software.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOr and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that may want to be explored.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name	Page
ChildOf	B	1420	Exposure of Sensitive Information during Transient Execution	2284

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name	Page
ChildOf	B	1420	Exposure of Sensitive Information during Transient Execution	2284

Applicable Platforms

Language : Not Language-Specific *(Prevalence = Undetermined)*

Operating_System : Not OS-Specific *(Prevalence = Undetermined)*

Architecture : Not Architecture-Specific *(Prevalence = Undetermined)*

Technology : Microcontroller Hardware *(Prevalence = Undetermined)*

Technology : Processor Hardware *(Prevalence = Undetermined)*

Technology : Memory Hardware *(Prevalence = Undetermined)*

Technology : System on Chip *(Prevalence = Undetermined)*

Common Consequences

Scope	Impact	Likelihood
Confidentiality	Read Memory	Medium

Detection Methods

Manual Analysis

This weakness can be detected in hardware by manually inspecting processor specifications. Features that exhibit this weakness may have microarchitectural predictor state that is shared between hardware threads, execution contexts (for example, user and kernel), or other components that may host mutually distrusting software (or firmware, etc.).

Effectiveness = Moderate

Manual analysis may not reveal all weaknesses in a processor specification and should be combined with other detection methods to improve coverage.

Automated Analysis

Software vendors can release tools that detect presence of known weaknesses on a processor. For example, some of these tools can attempt to transiently execute a vulnerable code sequence and detect whether code successfully leaks data in a manner consistent with the weakness under test. Alternatively, some hardware vendors provide enumeration for the presence of a weakness (or lack of a weakness). These enumeration bits can be checked and reported by system software. For example, Linux supports these checks for many commodity processors: $ cat /proc/cpuinfo | grep bugs | head -n 1 bugs : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa itlb_multihit srbds mmio_stale_data retbleed

Effectiveness = High

This method can be useful for detecting whether a processor if affected by known weaknesses, but it may not be useful for detecting unknown weaknesses

Automated Analysis

This weakness can be detected in hardware by employing static or dynamic taint analysis methods [REF-1401]. These methods can label each predictor entry (or prediction history, etc.) according to the processor context that created it. Taint analysis or information flow analysis can then be applied to detect when predictor state created in one context can influence predictions made in another context.

Effectiveness = Moderate

Automated static or dynamic taint analysis may not reveal all weaknesses in a processor specification and should be combined with other detection methods to improve coverage.

Potential Mitigations

Phase: Architecture and Design

The hardware designer can attempt to prevent transient execution from causing observable discrepancies in specific covert channels.

Phase: Architecture and Design

Hardware designers may choose to use microarchitectural bits to tag predictor entries. For example, each predictor entry may be tagged with a kernel-mode bit which, when set, indicates that the predictor entry was created in kernel mode. The processor can use this bit to enforce that predictions in the current mode must have been trained in the current mode. This can prevent malicious cross-mode training, such as when user-mode software attempts to create predictor entries that influence transient execution in the kernel. Predictor entry tags can also be used to associate each predictor entry with the SMT thread that created it, and thus the processor can enforce that each predictor entry can only be used by the SMT thread that created it. This can prevent an SMT thread from using predictor entries crafted by a malicious sibling SMT thread.

Effectiveness = Moderate

Tagging can be highly effective for predictor state that is comprised of discrete elements, such as an array of recently visited branch targets. Predictor state can also have different representations that are not conducive to tagging. For example, some processors keep a compressed digest of branch history which does not contain discrete elements that can be individually tagged.

Phase: Architecture and Design

Hardware designers may choose to sanitize microarchitectural predictor state (for example, branch prediction history) when the processor transitions to a different context, for example, whenever a system call is invoked. Alternatively, the hardware may expose instruction(s) that allow software to sanitize predictor state according to the user's threat model. For example, this can allow operating system software to sanitize predictor state when performing a context switch from one process to another.

Effectiveness = Moderate

This technique may not be able to mitigate weaknesses that arise from predictor state that is shared across SMT threads. Sanitizing predictor state on context switches may also negatively impact performance, either by removing predictor entries that could be reused when returning to the previous context, or by slowing down the context switch itself.

Phase: Implementation

System software can mitigate this weakness by invoking predictor-state-sanitizing operations (for example, the indirect branch prediction barrier on Intel x86) when switching from one context to another, according to the hardware vendor's recommendations.

Effectiveness = Moderate

This technique may not be able to mitigate weaknesses that arise from predictor state shared across SMT threads. Sanitizing predictor state may also negatively impact performance in some circumstances.

Phase: Build and Compilation

If the weakness is exposed by a single instruction (or a small set of instructions), then the compiler (or JIT, etc.) can be configured to prevent the affected instruction(s) from being generated. One prominent example of this mitigation is retpoline ([REF-1414]).

Effectiveness = Limited

This technique is only effective for software that is compiled with this mitigation. Additionally, an alternate instruction sequence may mitigate the weakness on some processors but not others, even when the processors share the same ISA. For example, retpoline has been documented as effective on some x86 processors, but not fully effective on other x86 processors.

Phase: Build and Compilation

Use control-flow integrity (CFI) techniques to constrain the behavior of instructions that redirect the instruction pointer, such as indirect branch instructions.

Effectiveness = Moderate

Some CFI techniques may not be able to constrain transient execution, even though they are effective at constraining architectural execution. Or they may be able to provide some additional protection against a transient execution weakness, but without comprehensively mitigating the weakness. For example, Clang-CFI provides strong architectural CFI properties and can make some transient execution weaknesses more difficult to exploit [REF-1398].

Phase: Build and Compilation

Use software techniques (including the use of serialization instructions) that are intended to reduce the number of instructions that can be executed transiently after a processor event or misprediction.

Effectiveness = Incidental

Some transient execution weaknesses can be exploited even if a single instruction is executed transiently after a processor event or mis-prediction. This mitigation strategy has many other pitfalls that prevent it from eliminating this weakness entirely. For example, see [REF-1389].

Phase: System Configuration

Some systems may allow the user to disable predictor sharing. For example, this could be a BIOS configuration, or a model-specific register (MSR) that can be configured by the operating system or virtual machine monitor.

Effectiveness = Moderate

Disabling predictor sharing can negatively impact performance for some workloads that benefit from shared predictor state.

Phase: Patching and Maintenance

The hardware vendor may provide a patch to, for example, sanitize predictor state when the processor transitions to a different context, or to prevent predictor entries from being shared across SMT threads. A patch may also introduce new ISA that allows software to toggle a mitigation.

Effectiveness = Moderate

This mitigation may only be fully effective if the patch prevents predictor sharing across all contexts that are affected by the weakness. Additionally, sanitizing predictor state and/or preventing shared predictor state can negatively impact performance in some circumstances.

Phase: Documentation

If a hardware feature can allow microarchitectural predictor state to be shared between contexts, SMT threads, or other architecturally defined boundaries, the hardware designer may opt to disclose this behavior in architecture documentation. This documentation can inform users about potential consequences and effective mitigations.

Effectiveness = High

Phase: Requirements

Processor designers, system software vendors, or other agents may choose to restrict the ability of unprivileged software to access to high-resolution timers that are commonly used to monitor covert channels.

Demonstrative Examples

Example 1:

Branch Target Injection (BTI) is a vulnerability that can allow an SMT hardware thread to maliciously train the indirect branch predictor state that is shared with its sibling hardware thread. A cross-thread BTI attack requires the attacker to find a vulnerable code sequence within the victim software. For example, the authors of [REF-1415] identified the following code sequence in the Windows library ntdll.dll:

Example Language: x86 Assembly *(Bad)*

```
  adc edi,dword ptr [ebx+edx+13BE13BDh]
  adc dl,byte ptr [edi]
  ...
indirect_branch_site:
  jmp dword ptr [rsi] # at this point attacker knows edx, controls edi and ebx
```

To successfully exploit this code sequence to disclose the victim's private data, the attacker must also be able to find an indirect branch site within the victim, where the attacker controls the values in edi and ebx, and the attacker knows the value in edx as shown above at the indirect branch site.

A proof-of-concept cross-thread BTI attack might proceed as follows:

1. The attacker thread and victim thread must be co-scheduled on the same physical processor core.
2. The attacker thread must train the shared branch predictor so that when the victim thread reaches indirect_branch_site, the jmp instruction will be predicted to target example_code_sequence instead of the correct architectural target. The training procedure may vary by processor, and the attacker may need to reverse-engineer the branch predictor to identify a suitable training algorithm.
3. This step assumes that the attacker can control some values in the victim program, specifically the values in edi and ebx at indirect_branch_site. When the victim reaches indirect_branch_site the processor will (mis)predict example_code_sequence as the target and (transiently) execute the adc instructions. If the attacker chooses ebx so that `ebx = m

- 0x13BE13BD - edx, then the first adc will load 32 bits from address m in the victim's address space and add *m (the data loaded from) to the attacker-controlled base address in edi. The second adc instruction accesses a location in memory whose address corresponds to *m`.

4. The adversary uses a covert channel analysis technique such as Flush+Reload ([REF-1416]) to infer the value of the victim's private data *m.

Example 2:

BTI can also allow software in one execution context to maliciously train branch predictor entries that can be used in another context. For example, on some processors user-mode software may be able to train predictor entries that can also be used after transitioning into kernel mode, such as after invoking a system call. This vulnerability does not necessarily require SMT and may instead be performed in synchronous steps, though it does require the attacker to find an exploitable code sequence in the victim's code, for example, in the kernel.

Observed Examples

Reference	Description
CVE-2017-5754	(Branch Target Injection, BTI, Spectre v2). Shared microarchitectural indirect branch predictor state may allow code to influence transient execution across a process, VM, or privilege boundary, potentially exposing data that is accessible beyond the boundary. *https://www.cve.org/CVERecord?id=CVE-2017-5754*
CVE-2022-0001	(Branch History Injection, BHI, Spectre-BHB). Shared branch history state may allow user-mode code to influence transient execution in the kernel, potentially exposing kernel data over a covert channel. *https://www.cve.org/CVERecord?id=CVE-2022-0001*
CVE-2021-33149	(RSB underflow, Retbleed). Shared return stack buffer state may allow code that executes before a prediction barrier to influence transient execution after the prediction barrier, potentially exposing data that is accessible beyond the barrier over a covert channel. *https://www.cve.org/CVERecord?id=CVE-2021-33149*

MemberOf Relationships

This MemberOf relationships table shows additional CWE Catgeories and Views that reference this weakness as a member. This information is often useful in understanding where a weakness fits within the context of external information sources.

Nature	Type	ID	Name	V	Page
MemberOf	C	1416	Comprehensive Categorization: Resource Lifecycle Management	1400	2545

References

[REF-1414]Intel Corporation. "Retpoline: A Branch Target Injection Mitigation". 2022 August 2. < https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/technical-documentation/retpoline-branch-target-injection-mitigation.html >.2023-02-13.

[REF-1415]Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz and Yuval Yarom. "Spectre Attacks: Exploiting Speculative Execution". 2019 May. < https://spectreattack.com/spectre.pdf >.2024-02-14.

[REF-1416]Yuval Yarom and Katrina Falkner. "Flush+Reload: A High Resolution, Low Noise, L3 Cache Side-Channel Attack". 2014. < https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-yarom.pdf >.2023-02-13.

[REF-1398]The Clang Team. "Control Flow Integrity". < https://clang.llvm.org/docs/ControlFlowIntegrity.html >.2024-02-13.

[REF-1389]Alyssa Milburn, Ke Sun and Henrique Kawakami. "You Cannot Always Win the Race: Analyzing the LFENCE/JMP Mitigation for Branch Target Injection". 2022 March 8. < https://arxiv.org/abs/2203.04277 >.2024-02-22.

[REF-1400]Intel Corporation. "Refined Speculative Execution Terminology". 2022 March 1. < https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/best-practices/refined-speculative-execution-terminology.html >.2024-02-13.

[REF-1401]Neta Bar Kama and Roope Kaivola. "Hardware Security Leak Detection by Symbolic Simulation". 2021 November. < https://ieeexplore.ieee.org/document/9617727 >.2024-02-13.

Categories

Category-2: 7PK - Environment

Category ID : 2

Summary

This category represents one of the phyla in the Seven Pernicious Kingdoms vulnerability classification. It includes weaknesses that are typically introduced during unexpected environmental conditions. According to the authors of the Seven Pernicious Kingdoms, "This section includes everything that is outside of the source code but is still critical to the security of the product that is being created. Because the issues covered by this kingdom are not directly related to source code, we separated it from the rest of the kingdoms."

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	700	Seven Pernicious Kingdoms	700	2557
MemberOf	C	933	OWASP Top Ten 2013 Category A5 - Security Misconfiguration	928	2391
MemberOf	C	1349	OWASP Top Ten 2021 Category A05:2021 - Security Misconfiguration	1344	2493
HasMember	V	5	J2EE Misconfiguration: Data Transmission Without Encryption	700	1
HasMember	V	6	J2EE Misconfiguration: Insufficient Session-ID Length	700	2
HasMember	V	7	J2EE Misconfiguration: Missing Custom Error Page	700	4
HasMember	V	8	J2EE Misconfiguration: Entity Bean Declared Remote	700	6
HasMember	V	9	J2EE Misconfiguration: Weak Access Permissions for EJB Methods	700	8
HasMember	V	11	ASP.NET Misconfiguration: Creating Debug Binary	700	9
HasMember	V	12	ASP.NET Misconfiguration: Missing Custom Error Page	700	11
HasMember	V	13	ASP.NET Misconfiguration: Password in Configuration File	700	13
HasMember	V	14	Compiler Removal of Code to Clear Buffers	700	14

References

[REF-6]Katrina Tsipenyuk, Brian Chess and Gary McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". NIST Workshop on Software Security Assurance Tools Techniques and Metrics. 2005 November 7. NIST. < https://samate.nist.gov/SSATTM_Content/

papers/Seven%20Pernicious%20Kingdoms%20-%20Taxonomy%20of%20Sw%20Security %20Errors%20-%20Tsipenyuk%20-%20Chess%20-%20McGraw.pdf >.

Category-16: Configuration

Category ID : 16

Summary

Weaknesses in this category are typically introduced during the configuration of the software.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	635	Weaknesses Originally Used by NVD from 2008 to 2016	635	2552
MemberOf	C	933	OWASP Top Ten 2013 Category A5 - Security Misconfiguration	928	2391
MemberOf	C	1032	OWASP Top Ten 2017 Category A6 - Security Misconfiguration	1026	2438
MemberOf	C	1349	OWASP Top Ten 2021 Category A05:2021 - Security Misconfiguration	1344	2493

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
WASC	14		Server Misconfiguration
WASC	15		Application Misconfiguration

Notes

Maintenance

Further discussion about this category was held over the CWE Research mailing list in early 2020. No definitive action has been decided.

Maintenance

This entry is a Category, but various sources map to it anyway, despite CWE guidance that Categories should not be mapped. In this case, there are no clear CWE Weaknesses that can be utilized. "Inappropriate Configuration" sounds more like a Weakness in CWE's style, but it still does not indicate actual behavior of the product. Further research is still required, however, as a "configuration weakness" might be Primary to many other CWEs, i.e., it might be better described in terms of chaining relationships.

References

[REF-1287]MITRE. "Supplemental Details - 2022 CWE Top 25". 2022 June 8. < https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25_supplemental.html#problematicMappingDetails >.

Category-19: Data Processing Errors

Category ID : 19

Summary

Weaknesses in this category are typically found in functionality that processes data. Data processing is the manipulation of input to retrieve or save information.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	130	Improper Handling of Length Parameter Inconsistency	699	351
HasMember	B	166	Improper Handling of Missing Special Element	699	423
HasMember	B	167	Improper Handling of Additional Special Element	699	425
HasMember	B	168	Improper Handling of Inconsistent Special Elements	699	426
HasMember	B	178	Improper Handling of Case Sensitivity	699	445
HasMember	B	182	Collapse of Data into Unsafe Value	699	455
HasMember	B	186	Overly Restrictive Regular Expression	699	466
HasMember	B	229	Improper Handling of Values	699	570
HasMember	B	233	Improper Handling of Parameters	699	574
HasMember	B	237	Improper Handling of Structural Elements	699	580
HasMember	B	241	Improper Handling of Unexpected Data Type	699	584
HasMember	B	409	Improper Handling of Highly Compressed Data (Data Amplification)	699	996
HasMember	B	472	External Control of Assumed-Immutable Web Parameter	699	1123
HasMember	B	601	URL Redirection to Untrusted Site ('Open Redirect')	699	1345
HasMember	B	611	Improper Restriction of XML External Entity Reference	699	1367
HasMember	B	624	Executable Regular Expression Error	699	1390
HasMember	B	625	Permissive Regular Expression	699	1392
HasMember	B	776	Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')	699	1633
HasMember	B	1024	Comparison of Incompatible Types	699	1867

Category-133: String Errors

Category ID : 133

Summary

Weaknesses in this category are related to the creation and modification of strings.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	134	Use of Externally-Controlled Format String	699	365
HasMember	B	135	Incorrect Calculation of Multi-Byte String Length	699	370
HasMember	B	480	Use of Incorrect Operator	699	1150

Category-136: Type Errors

Category ID : 136

Summary

Weaknesses in this category are caused by improper data type transformation or improper handling of multiple data types.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	681	Incorrect Conversion between Numeric Types	699	1495
HasMember	B	843	Access of Resource Using Incompatible Type ('Type Confusion')	699	1776
HasMember	B	1287	Improper Validation of Specified Type of Input	699	2138

Category-137: Data Neutralization Issues

Category ID : 137

Summary

Weaknesses in this category are related to the creation or neutralization of data using an incorrect format.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	76	Improper Neutralization of Equivalent Special Elements	699	144
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	699	151
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	699	163
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	699	194
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	699	201
HasMember	B	90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	699	212
HasMember	B	91	XML Injection (aka Blind XPath Injection)	699	215
HasMember	B	93	Improper Neutralization of CRLF Sequences ('CRLF Injection')	699	217
HasMember	B	94	Improper Control of Generation of Code ('Code Injection')	699	219
HasMember	B	117	Improper Output Neutralization for Logs	699	288
HasMember	B	140	Improper Neutralization of Delimiters	699	376
HasMember	B	170	Improper Null Termination	699	428
HasMember	B	463	Deletion of Data Structure Sentinel	699	1105
HasMember	B	464	Addition of Data Structure Sentinel	699	1107
HasMember	B	641	Improper Restriction of Names for Files and Other Resources	699	1412
HasMember	B	694	Use of Multiple Resources with Duplicate Identifier	699	1523
HasMember	B	791	Incomplete Filtering of Special Elements	699	1680
HasMember	B	838	Inappropriate Encoding for Output Context	699	1764
HasMember	B	917	Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')	699	1818
HasMember	B	1236	Improper Neutralization of Formula Elements in a CSV File	699	2019

Category-189: Numeric Errors

Category ID : 189

Summary

Weaknesses in this category are related to improper calculation or conversion of numbers.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	635	Weaknesses Originally Used by NVD from 2008 to 2016	635	2552
MemberOf	V	699	Software Development	699	2555
MemberOf	C	1182	SEI CERT Perl Coding Standard - Guidelines 04. Integers (INT)	1178	2466
HasMember	B	128	Wrap-around Error	699	339
HasMember	B	190	Integer Overflow or Wraparound	699	472
HasMember	B	191	Integer Underflow (Wrap or Wraparound)	699	480
HasMember	B	193	Off-by-one Error	699	486
HasMember	B	369	Divide By Zero	699	913
HasMember	B	681	Incorrect Conversion between Numeric Types	699	1495
HasMember	B	839	Numeric Range Comparison Without Minimum Check	699	1767
HasMember	B	1335	Incorrect Bitwise Shift of Integer	699	2235
HasMember	B	1339	Insufficient Precision or Accuracy of a Real Number	699	2242
HasMember	B	1389	Incorrect Parsing of Numbers with Different Radices	699	2263

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
SEI CERT Perl Coding Standard	INT01-PL	CWE More Abstract	Use small integers when precise computation is required

References

[REF-1287]MITRE. "Supplemental Details - 2022 CWE Top 25". 2022 June 8. < https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25_supplemental.html#problematicMappingDetails >.

Category-199: Information Management Errors

Category ID : 199

Summary

Weaknesses in this category are related to improper handling of sensitive information.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	201	Insertion of Sensitive Information Into Sent Data	699	514
HasMember	B	204	Observable Response Discrepancy	699	523
HasMember	B	205	Observable Behavioral Discrepancy	699	526
HasMember	B	208	Observable Timing Discrepancy	699	529
HasMember	B	209	Generation of Error Message Containing Sensitive Information	699	533

Nature	Type	ID	Name	V	Page
HasMember	B	212	Improper Removal of Sensitive Information Before Storage or Transfer	699	544
HasMember	B	213	Exposure of Sensitive Information Due to Incompatible Policies	699	547
HasMember	B	214	Invocation of Process Using Visible Sensitive Information	699	549
HasMember	B	215	Insertion of Sensitive Information Into Debugging Code	699	551
HasMember	B	312	Cleartext Storage of Sensitive Information	699	764
HasMember	B	319	Cleartext Transmission of Sensitive Information	699	779
HasMember	B	359	Exposure of Private Personal Information to an Unauthorized Actor	699	882
HasMember	B	497	Exposure of Sensitive System Information to an Unauthorized Control Sphere	699	1193
HasMember	B	524	Use of Cache Containing Sensitive Information	699	1232
HasMember	B	538	Insertion of Sensitive Information into Externally-Accessible File or Directory	699	1248
HasMember	B	921	Storage of Sensitive Data in a Mechanism without Access Control	699	1824
HasMember	B	1230	Exposure of Sensitive Information Through Metadata	699	2006

Category-227: 7PK - API Abuse

Category ID : 227

Summary

This category represents one of the phyla in the Seven Pernicious Kingdoms vulnerability classification. It includes weaknesses that involve the software using an API in a manner contrary to its intended use. According to the authors of the Seven Pernicious Kingdoms, "An API is a contract between a caller and a callee. The most common forms of API misuse occurs when the caller does not honor its end of this contract. For example, if a program does not call chdir() after calling chroot(), it violates the contract that specifies how to change the active root directory in a secure fashion. Another good example of library abuse is expecting the callee to return trustworthy DNS information to the caller. In this case, the caller misuses the callee API by making certain assumptions about its behavior (that the return value can be used for authentication purposes). One can also violate the caller-callee contract from the other side. For example, if a coder subclasses SecureRandom and returns a non-random value, the contract is violated."

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	700	Seven Pernicious Kingdoms	700	2557
MemberOf	C	1001	SFP Secondary Cluster: Use of an Improper API	888	2420
HasMember	B	242	Use of Inherently Dangerous Function	700	586
HasMember	V	243	Creation of chroot Jail Without Changing Working Directory	700	589
HasMember	V	244	Improper Clearing of Heap Memory Before Release ('Heap Inspection')	700	591
HasMember	V	245	J2EE Bad Practices: Direct Management of Connections	700	592
HasMember	V	246	J2EE Bad Practices: Direct Use of Sockets	700	594
HasMember	B	248	Uncaught Exception	700	596
HasMember	B	250	Execution with Unnecessary Privileges	700	599

Nature	Type	ID	Name	V	Page
HasMember	C	251	Often Misused: String Management	700	2314
HasMember	B	252	Unchecked Return Value	700	606
HasMember	V	558	Use of getlogin() in Multithreaded Application	700	1272

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
CERT C Secure Coding	WIN30-C	CWE More Abstract	Properly pair allocation and deallocation functions

References

[REF-6]Katrina Tsipenyuk, Brian Chess and Gary McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". NIST Workshop on Software Security Assurance Tools Techniques and Metrics. 2005 November 7. NIST. < https://samate.nist.gov/SSATTM_Content/papers/Seven%20Pernicious%20Kingdoms%20-%20Taxonomy%20of%20Sw%20Security%20Errors%20-%20Tsipenyuk%20-%20Chess%20-%20McGraw.pdf >.

Category-251: Often Misused: String Management

Category ID : 251

Summary

Functions that manipulate strings encourage buffer overflows.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	227	7PK - API Abuse	700	2313
MemberOf	C	974	SFP Secondary Cluster: Incorrect Buffer Length Computation	888	2406

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Often Misused: Strings
Software Fault Patterns	SFP10		Incorrect Buffer Length Computation

References

[REF-6]Katrina Tsipenyuk, Brian Chess and Gary McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". NIST Workshop on Software Security Assurance Tools Techniques and Metrics. 2005 November 7. NIST. < https://samate.nist.gov/SSATTM_Content/papers/Seven%20Pernicious%20Kingdoms%20-%20Taxonomy%20of%20Sw%20Security%20Errors%20-%20Tsipenyuk%20-%20Chess%20-%20McGraw.pdf >.

Category-254: 7PK - Security Features

Category ID : 254

Summary

Software security is not security software. Here we're concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	700	Seven Pernicious Kingdoms	700	2557
HasMember	B	256	Plaintext Storage of a Password	700	615
HasMember	V	258	Empty Password in Configuration File	700	621
HasMember	V	259	Use of Hard-coded Password	700	623
HasMember	B	260	Password in Configuration File	700	629
HasMember	B	261	Weak Encoding for Password	700	631
HasMember	B	272	Least Privilege Violation	700	656
HasMember	P	284	Improper Access Control	700	680
HasMember	C	285	Improper Authorization	700	684
HasMember	C	330	Use of Insufficiently Random Values	700	814
HasMember	B	359	Exposure of Private Personal Information to an Unauthorized Actor	700	882
HasMember	B	798	Use of Hard-coded Credentials	700	1690

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Security Features

References

[REF-6]Katrina Tsipenyuk, Brian Chess and Gary McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". NIST Workshop on Software Security Assurance Tools Techniques and Metrics. 2005 November 7. NIST. < https://samate.nist.gov/SSATTM_Content/papers/Seven%20Pernicious%20Kingdoms%20-%20Taxonomy%20of%20Sw%20Security%20Errors%20-%20Tsipenyuk%20-%20Chess%20-%20McGraw.pdf >.

Category-255: Credentials Management Errors

Category ID : 255

Summary

Weaknesses in this category are related to the management of credentials.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	635	Weaknesses Originally Used by NVD from 2008 to 2016	635	2552
MemberOf	V	699	Software Development	699	2555
MemberOf	C	724	OWASP Top Ten 2004 Category A3 - Broken Authentication and Session Management	711	2335
MemberOf	C	1353	OWASP Top Ten 2021 Category A07:2021 - Identification and Authentication Failures	1344	2494
HasMember	B	256	Plaintext Storage of a Password	699	615
HasMember	B	257	Storing Passwords in a Recoverable Format	699	618
HasMember	B	260	Password in Configuration File	699	629
HasMember	B	261	Weak Encoding for Password	699	631
HasMember	B	262	Not Using Password Aging	699	633
HasMember	B	263	Password Aging with Long Expiration	699	636
HasMember	B	324	Use of a Key Past its Expiration Date	699	792
HasMember	B	521	Weak Password Requirements	699	1223
HasMember	B	523	Unprotected Transport of Credentials	699	1230
HasMember	B	549	Missing Password Field Masking	699	1262

Nature	Type	ID	Name	V	Page
HasMember	B	620	Unverified Password Change	699	1383
HasMember	B	640	Weak Password Recovery Mechanism for Forgotten Password	699	1409
HasMember	B	798	Use of Hard-coded Credentials	699	1690
HasMember	B	916	Use of Password Hash With Insufficient Computational Effort	699	1813
HasMember	B	1392	Use of Default Credentials	699	2271

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
OWASP Top Ten 2004	A3	CWE More Specific	Broken Authentication and Session Management

References

[REF-1287]MITRE. "Supplemental Details - 2022 CWE Top 25". 2022 June 8. < https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25_supplemental.html#problematicMappingDetails >.

Category-264: Permissions, Privileges, and Access Controls

Category ID : 264

Summary

Weaknesses in this category are related to the management of permissions, privileges, and other security features that are used to perform access control.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	635	Weaknesses Originally Used by NVD from 2008 to 2016	635	2552
MemberOf	C	1345	OWASP Top Ten 2021 Category A01:2021 - Broken Access Control	1344	2487

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Permissions, Privileges, and ACLs

Notes

Maintenance

This entry heavily overlaps other categories and has been marked obsolete.

References

[REF-7]Michael Howard and David LeBlanc. "Writing Secure Code". 2nd Edition. 2002 December 4. Microsoft Press. < https://www.microsoftpressstore.com/store/writing-secure-code-9780735617223 >.

[REF-1287]MITRE. "Supplemental Details - 2022 CWE Top 25". 2022 June 8. < https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25_supplemental.html#problematicMappingDetails >.

Category-265: Privilege Issues

Category ID : 265

Summary

Weaknesses in this category occur with improper handling, assignment, or management of privileges. A privilege is a property of an agent, such as a user. It lets the agent do things that are not ordinarily allowed. For example, there are privileges which allow an agent to perform maintenance functions such as restart a computer.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	V	243	Creation of chroot Jail Without Changing Working Directory	699	589
HasMember	B	250	Execution with Unnecessary Privileges	699	599
HasMember	B	266	Incorrect Privilege Assignment	699	638
HasMember	B	267	Privilege Defined With Unsafe Actions	699	641
HasMember	B	268	Privilege Chaining	699	644
HasMember	B	270	Privilege Context Switching Error	699	651
HasMember	B	272	Least Privilege Violation	699	656
HasMember	B	273	Improper Check for Dropped Privileges	699	660
HasMember	B	274	Improper Handling of Insufficient Privileges	699	663
HasMember	B	280	Improper Handling of Insufficient Permissions or Privileges	699	672
HasMember	B	501	Trust Boundary Violation	699	1203
HasMember	V	580	clone() Method Without super.clone()	699	1311
HasMember	B	648	Incorrect Use of Privileged APIs	699	1428

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Privilege / sandbox errors

Notes

Relationship

This can strongly overlap authorization errors.

Theoretical

A sandbox could be regarded as an explicitly defined sphere of control, in that the sandbox only defines a limited set of behaviors, which can only access a limited set of resources.

Theoretical

It could be argued that any privilege problem occurs within the context of a sandbox.

Research Gap

Many of the following concepts require deeper study. Most privilege problems are not classified at such a low level of detail, and terminology is very sparse. Certain classes of software, such as web browsers and software bug trackers, provide a rich set of examples for further research. Operating systems have matured to the point that these kinds of weaknesses are rare, but finer-grained models for privileges, capabilities, or roles might introduce subtler issues.

Category-275: Permission Issues

Category ID : 275

Summary

Weaknesses in this category are related to improper assignment or handling of permissions.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
MemberOf	C	723	OWASP Top Ten 2004 Category A2 - Broken Access Control	711	2335
MemberOf	C	731	OWASP Top Ten 2004 Category A10 - Insecure Configuration Management	711	2339
MemberOf	C	1345	OWASP Top Ten 2021 Category A01:2021 - Broken Access Control	1344	2487
HasMember	B	276	Incorrect Default Permissions	699	665
HasMember	V	277	Insecure Inherited Permissions	699	668
HasMember	V	278	Insecure Preserved Inherited Permissions	699	669
HasMember	V	279	Incorrect Execution-Assigned Permissions	699	671
HasMember	B	280	Improper Handling of Insufficient Permissions or Privileges	699	672
HasMember	B	281	Improper Preservation of Permissions	699	674
HasMember	V	618	Exposed Unsafe ActiveX Method	699	1380
HasMember	B	766	Critical Data Element Declared Public	699	1607
HasMember	B	767	Access to Critical Private Variable via Public Method	699	1610

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Permission errors
OWASP Top Ten 2004	A2	CWE More Specific	Broken Access Control
OWASP Top Ten 2004	A10	CWE More Specific	Insecure Configuration Management

Notes

Terminology

Permissions are associated with a resource and specify which actors are allowed to access that resource and what they are allowed to do with that access (e.g., read it, modify it). Privileges are associated with an actor and define which behaviors or actions an actor is allowed to perform.

References

[REF-44]Michael Howard, David LeBlanc and John Viega. "24 Deadly Sins of Software Security". McGraw-Hill. 2010.

Category-310: Cryptographic Issues

Category ID : 310

Summary

Weaknesses in this category are related to the design and implementation of data confidentiality and integrity. Frequently these deal with the use of encoding techniques, encryption libraries, and hashing algorithms. The weaknesses in this category could lead to a degradation of the quality data if they are not addressed.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	635	Weaknesses Originally Used by NVD from 2008 to 2016	635	2552

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
MemberOf	C	1346	OWASP Top Ten 2021 Category A02:2021 - Cryptographic Failures	1344	2488
HasMember	B	261	Weak Encoding for Password	699	631
HasMember	B	324	Use of a Key Past its Expiration Date	699	792
HasMember	B	325	Missing Cryptographic Step	699	794
HasMember	B	328	Use of Weak Hash	699	806
HasMember	B	331	Insufficient Entropy	699	821
HasMember	B	334	Small Space of Random Values	699	827
HasMember	B	335	Incorrect Usage of Seeds in Pseudo-Random Number Generator (PRNG)	699	829
HasMember	B	338	Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)	699	837
HasMember	B	347	Improper Verification of Cryptographic Signature	699	857
HasMember	B	916	Use of Password Hash With Insufficient Computational Effort	699	1813
HasMember	B	1204	Generation of Weak Initialization Vector (IV)	699	1987
HasMember	B	1240	Use of a Cryptographic Primitive with a Risky Implementation	699	2025

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Cryptographic Issues

References

[REF-7]Michael Howard and David LeBlanc. "Writing Secure Code". 2nd Edition. 2002 December 4. Microsoft Press. < https://www.microsoftpressstore.com/store/writing-secure-code-9780735617223 >.

[REF-1287]MITRE. "Supplemental Details - 2022 CWE Top 25". 2022 June 8. < https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25_supplemental.html#problematicMappingDetails >.

Category-320: Key Management Errors

Category ID : 320

Summary

Weaknesses in this category are related to errors in the management of cryptographic keys.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
MemberOf	C	934	OWASP Top Ten 2013 Category A6 - Sensitive Data Exposure	928	2391
MemberOf	C	1029	OWASP Top Ten 2017 Category A3 - Sensitive Data Exposure	1026	2436
HasMember	B	322	Key Exchange without Entity Authentication	699	788
HasMember	B	323	Reusing a Nonce, Key Pair in Encryption	699	790
HasMember	B	324	Use of a Key Past its Expiration Date	699	792
HasMember	B	798	Use of Hard-coded Credentials	699	1690

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Key Management Errors

Notes

Maintenance

This entry heavily overlaps other categories and has been marked obsolete.

Category-355: User Interface Security Issues

Category ID : 355

Summary

Weaknesses in this category are related to or introduced in the User Interface (UI).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	356	Product UI does not Warn User of Unsafe Actions	699	879
HasMember	B	357	Insufficient UI Warning of Dangerous Operations	699	880
HasMember	B	447	Unimplemented or Unsupported Feature in UI	699	1075
HasMember	B	448	Obsolete Feature in UI	699	1076
HasMember	B	449	The UI Performs the Wrong Action	699	1077
HasMember	B	549	Missing Password Field Masking	699	1262
HasMember	B	1007	Insufficient Visual Distinction of Homoglyphs Presented to User	699	1857
HasMember	B	1021	Improper Restriction of Rendered UI Layers or Frames	699	1860

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			(UI) User Interface Errors

Notes

Research Gap

User interface errors that are relevant to security have not been studied at a high level.

Category-361: 7PK - Time and State

Category ID : 361

Summary

This category represents one of the phyla in the Seven Pernicious Kingdoms vulnerability classification. It includes weaknesses related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads. According to the authors of the Seven Pernicious Kingdoms, "Distributed computation is about time and state. That is, in order for more than one component to communicate, state must be shared, and all that takes time. Most programmers anthropomorphize their work. They think about one thread of control carrying out the entire program in the same way they would if they had to do the job themselves. Modern computers, however, switch between tasks very quickly, and in multi-core, multi-CPU, or distributed systems, two events may take place at exactly the same time. Defects rush to fill the gap between the programmer's model of how a program executes and what happens in reality. These defects are related to unexpected interactions between threads, processes, time, and information. These interactions happen through shared state: semaphores, variables, the file system, and, basically, anything that can store information."

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	700	Seven Pernicious Kingdoms	700	2557
HasMember	B	364	Signal Handler Race Condition	700	899
HasMember	B	367	Time-of-check Time-of-use (TOCTOU) Race Condition	700	906
HasMember	C	377	Insecure Temporary File	700	925
HasMember	V	382	J2EE Bad Practices: Use of System.exit()	700	933
HasMember	V	383	J2EE Bad Practices: Direct Use of Threads	700	935
HasMember		384	Session Fixation	700	936
HasMember	B	412	Unrestricted Externally Accessible Lock	700	1000

References

[REF-6]Katrina Tsipenyuk, Brian Chess and Gary McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". NIST Workshop on Software Security Assurance Tools Techniques and Metrics. 2005 November 7. NIST. < https://samate.nist.gov/SSATTM_Content/papers/Seven%20Pernicious%20Kingdoms%20-%20Taxonomy%20of%20Sw%20Security%20Errors%20-%20Tsipenyuk%20-%20Chess%20-%20McGraw.pdf >.

Category-371: State Issues

Category ID : 371

Summary

Weaknesses in this category are related to improper management of system state.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	15	External Control of System or Configuration Setting	699	17
HasMember	B	372	Incomplete Internal State Distinction	699	919
HasMember	B	374	Passing Mutable Objects to an Untrusted Method	699	920
HasMember	B	375	Returning a Mutable Object to an Untrusted Caller	699	923
HasMember	B	1265	Unintended Reentrant Invocation of Non-reentrant Code Via Nested Calls	699	2088

Category-387: Signal Errors

Category ID : 387

Summary

Weaknesses in this category are related to the improper handling of signals.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	364	Signal Handler Race Condition	699	899

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Signal Errors

Notes

Maintenance

Several weaknesses could exist, but this needs more study. Some weaknesses might be unhandled signals, untrusted signals, and sending the wrong signals.

Category-388: 7PK - Errors

Category ID : 388

Summary

This category represents one of the phyla in the Seven Pernicious Kingdoms vulnerability classification. It includes weaknesses that occur when an application does not properly handle errors that occur during processing. According to the authors of the Seven Pernicious Kingdoms, "Errors and error handling represent a class of API. Errors related to error handling are so common that they deserve a special kingdom of their own. As with 'API Abuse,' there are two ways to introduce an error-related security vulnerability: the most common one is handling errors poorly (or not at all). The second is producing errors that either give out too much information (to possible attackers) or are difficult to handle."

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	700	Seven Pernicious Kingdoms	700	2557
HasMember	B	391	Unchecked Error Condition	700	948
HasMember	B	395	Use of NullPointerException Catch to Detect NULL Pointer Dereference	700	957
HasMember	B	396	Declaration of Catch for Generic Exception	700	959
HasMember	B	397	Declaration of Throws for Generic Exception	700	961

References

[REF-6]Katrina Tsipenyuk, Brian Chess and Gary McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". NIST Workshop on Software Security Assurance Tools Techniques and Metrics. 2005 November 7. NIST. < https://samate.nist.gov/SSATTM_Content/papers/Seven%20Pernicious%20Kingdoms%20-%20Taxonomy%20of%20Sw%20Security%20Errors%20-%20Tsipenyuk%20-%20Chess%20-%20McGraw.pdf >.

Category-389: Error Conditions, Return Values, Status Codes

Category ID : 389

Summary

This category includes weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function. This type of problem is most often found in conditions that are rarely encountered during the normal operation of the product. Presumably, most bugs related to common conditions are found and eliminated during development and testing. In some cases, the attacker can directly control or influence the environment to trigger the rare conditions.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
MemberOf	C	728	OWASP Top Ten 2004 Category A7 - Improper Error Handling	711	2337
HasMember	B	209	Generation of Error Message Containing Sensitive Information	699	533
HasMember	B	248	Uncaught Exception	699	596
HasMember	B	252	Unchecked Return Value	699	606
HasMember	B	253	Incorrect Check of Function Return Value	699	613
HasMember	B	390	Detection of Error Condition Without Action	699	943
HasMember	B	391	Unchecked Error Condition	699	948
HasMember	B	392	Missing Report of Error Condition	699	951
HasMember	B	393	Return of Wrong Status Code	699	953
HasMember	B	394	Unexpected Status Code or Return Value	699	955
HasMember	B	395	Use of NullPointerException Catch to Detect NULL Pointer Dereference	699	957
HasMember	B	396	Declaration of Catch for Generic Exception	699	959
HasMember	B	397	Declaration of Throws for Generic Exception	699	961
HasMember	B	544	Missing Standardized Error Handling Mechanism	699	1256
HasMember	B	584	Return Inside Finally Block	699	1317
HasMember	B	617	Reachable Assertion	699	1378
HasMember	B	756	Missing Custom Error Page	699	1579

Notes

Other

Many researchers focus on the resultant weaknesses and do not necessarily diagnose whether a rare condition is the primary factor. However, since 2005 it seems to be reported more frequently than in the past. This subject needs more study.

References

[REF-44]Michael Howard, David LeBlanc and John Viega. "24 Deadly Sins of Software Security". McGraw-Hill. 2010.

Category-398: 7PK - Code Quality

Category ID : 398

Summary

This category represents one of the phyla in the Seven Pernicious Kingdoms vulnerability classification. It includes weaknesses that do not directly introduce a weakness or vulnerability, but indicate that the product has not been carefully developed or maintained. According to the authors of the Seven Pernicious Kingdoms, "Poor code quality leads to unpredictable behavior. From a user's perspective that often manifests itself as poor usability. For an adversary it provides an opportunity to stress the system in unexpected ways."

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	700	Seven Pernicious Kingdoms	700	2557
MemberOf	C	978	SFP Secondary Cluster: Implementation	888	2408
HasMember	V	401	Missing Release of Memory after Effective Lifetime	700	973
HasMember	C	404	Improper Resource Shutdown or Release	700	980
HasMember	V	415	Double Free	700	1008
HasMember	V	416	Use After Free	700	1012
HasMember	V	457	Use of Uninitialized Variable	700	1094
HasMember	B	474	Use of Function with Inconsistent Implementations	700	1128
HasMember	B	475	Undefined Behavior for Input to API	700	1130
HasMember	B	476	NULL Pointer Dereference	700	1132
HasMember	B	477	Use of Obsolete Function	700	1138

References

[REF-6]Katrina Tsipenyuk, Brian Chess and Gary McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". NIST Workshop on Software Security Assurance Tools Techniques and Metrics. 2005 November 7. NIST. < https://samate.nist.gov/SSATTM_Content/papers/Seven%20Pernicious%20Kingdoms%20-%20Taxonomy%20of%20Sw%20Security%20Errors%20-%20Tsipenyuk%20-%20Chess%20-%20McGraw.pdf >.

Category-399: Resource Management Errors

Category ID : 399

Summary

Weaknesses in this category are related to improper management of system resources.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	635	Weaknesses Originally Used by NVD from 2008 to 2016	635	2552
MemberOf	V	699	Software Development	699	2555
HasMember	B	73	External Control of File Name or Path	699	132
HasMember	B	403	Exposure of File Descriptor to Unintended Control Sphere ('File Descriptor Leak')	699	978
HasMember	B	410	Insufficient Resource Pool	699	998
HasMember	B	470	Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')	699	1118
HasMember	B	502	Deserialization of Untrusted Data	699	1204
HasMember	B	619	Dangling Database Cursor ('Cursor Injection')	699	1382
HasMember	B	641	Improper Restriction of Names for Files and Other Resources	699	1412
HasMember	B	694	Use of Multiple Resources with Duplicate Identifier	699	1523
HasMember	B	763	Release of Invalid Pointer or Reference	699	1599
HasMember	B	770	Allocation of Resources Without Limits or Throttling	699	1613
HasMember	B	771	Missing Reference to Active Allocated Resource	699	1622
HasMember	B	772	Missing Release of Resource after Effective Lifetime	699	1624
HasMember	B	826	Premature Release of Resource During Expected Lifetime	699	1734
HasMember	B	908	Use of Uninitialized Resource	699	1792
HasMember	C	909	Missing Initialization of Resource	699	1797

Nature	Type	ID	Name	V	Page
HasMember	B	910	Use of Expired File Descriptor	699	1800
HasMember	B	911	Improper Update of Reference Count	699	1801
HasMember	B	914	Improper Control of Dynamically-Identified Variables	699	1807
HasMember	B	915	Improperly Controlled Modification of Dynamically-Determined Object Attributes	699	1809
HasMember	B	920	Improper Restriction of Power Consumption	699	1823
HasMember	B	1188	Initialization of a Resource with an Insecure Default	699	1974
HasMember	B	1341	Multiple Releases of Same Resource or Handle	699	2246

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Resource Management Errors

References

[REF-1287]MITRE. "Supplemental Details - 2022 CWE Top 25". 2022 June 8. < https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25_supplemental.html#problematicMappingDetails >.

Category-411: Resource Locking Problems

Category ID : 411

Summary

Weaknesses in this category are related to improper handling of locks that are used to control access to resources.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	412	Unrestricted Externally Accessible Lock	699	1000
HasMember	B	413	Improper Resource Locking	699	1003
HasMember	B	414	Missing Lock Check	699	1007
HasMember	B	609	Double-Checked Locking	699	1362
HasMember	B	764	Multiple Locks of a Critical Resource	699	1604
HasMember	B	765	Multiple Unlocks of a Critical Resource	699	1605
HasMember	B	832	Unlock of a Resource that is not Locked	699	1752
HasMember	B	833	Deadlock	699	1753

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Resource Locking problems

Category-417: Communication Channel Errors

Category ID : 417

Summary

Weaknesses in this category are related to improper handling of communication channels and access paths. These weaknesses include problems in creating, managing, or removing alternate channels and alternate paths. Some of these can overlap virtual file problems and are commonly used in "bypass" attacks, such as those that exploit authentication errors.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	322	Key Exchange without Entity Authentication	699	788
HasMember	C	346	Origin Validation Error	699	853
HasMember	B	385	Covert Timing Channel	699	940
HasMember	B	419	Unprotected Primary Channel	699	1017
HasMember	B	420	Unprotected Alternate Channel	699	1018
HasMember	B	425	Direct Request ('Forced Browsing')	699	1025
HasMember	B	515	Covert Storage Channel	699	1220
HasMember	B	918	Server-Side Request Forgery (SSRF)	699	1820
HasMember	B	924	Improper Enforcement of Message Integrity During Transmission in a Communication Channel	699	1830
HasMember	B	940	Improper Verification of Source of a Communication Channel	699	1842
HasMember	B	941	Incorrectly Specified Destination in a Communication Channel	699	1845
HasMember	B	1327	Binding to an Unrestricted IP Address	699	2215

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER	CHAP.VIRTFILE		Channel and Path Errors

Notes

Research Gap

Most of these issues are probably under-studied. Only a handful of public reports exist.

Category-429: Handler Errors

Category ID : 429

Summary

Weaknesses in this category are related to improper management of handlers.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	430	Deployment of Wrong Handler	699	1042
HasMember	B	431	Missing Handler	699	1043
HasMember	B	434	Unrestricted Upload of File with Dangerous Type	699	1048

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Handler Errors

Category-438: Behavioral Problems

Category ID : 438

Summary

Weaknesses in this category are related to unexpected behaviors from code that an application uses.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	115	Misinterpretation of Input	699	280
HasMember	B	179	Incorrect Behavior Order: Early Validation	699	448
HasMember	B	408	Incorrect Behavior Order: Early Amplification	699	995
HasMember	B	437	Incomplete Model of Endpoint Features	699	1059
HasMember	B	439	Behavioral Change in New Version or Environment	699	1061
HasMember	B	440	Expected Behavior Violation	699	1062
HasMember	B	444	Inconsistent Interpretation of HTTP Requests ('HTTP Request/Response Smuggling')	699	1068
HasMember	B	480	Use of Incorrect Operator	699	1150
HasMember	B	483	Incorrect Block Delimitation	699	1160
HasMember	B	484	Omitted Break Statement in Switch	699	1162
HasMember	B	551	Incorrect Behavior Order: Authorization Before Parsing and Canonicalization	699	1264
HasMember	B	698	Execution After Redirect (EAR)	699	1533
HasMember	B	733	Compiler Optimization Removal or Modification of Security-critical Code	699	1562
HasMember	B	783	Operator Precedence Logic Error	699	1650
HasMember	B	835	Loop with Unreachable Exit Condition ('Infinite Loop')	699	1757
HasMember	B	837	Improper Enforcement of a Single, Unique Action	699	1762
HasMember	B	841	Improper Enforcement of Behavioral Workflow	699	1772
HasMember	B	1025	Comparison Using Wrong Factors	699	1868
HasMember	B	1037	Processor Optimization Removal or Modification of Security-critical Code	699	1870

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Behavioral problems

Category-452: Initialization and Cleanup Errors

Category ID : 452

Summary

Weaknesses in this category occur in behaviors that are used for initialization and breakdown.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	212	Improper Removal of Sensitive Information Before Storage or Transfer	699	544
HasMember	B	454	External Initialization of Trusted Variables or Data Stores	699	1085

Nature	Type	ID	Name	V	Page
HasMember	B	455	Non-exit on Failed Initialization	699	1087
HasMember	B	459	Incomplete Cleanup	699	1099
HasMember	B	1051	Initialization with Hard-Coded Network Resource Configuration Data	699	1886
HasMember	B	1052	Excessive Use of Hard-Coded Literals in Initialization	699	1887
HasMember	B	1188	Initialization of a Resource with an Insecure Default	699	1974

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Initialization and Cleanup Errors

Category-465: Pointer Issues

Category ID : 465

Summary

Weaknesses in this category are related to improper handling of pointers.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	466	Return of Pointer Value Outside of Expected Range	699	1109
HasMember	B	468	Incorrect Pointer Scaling	699	1114
HasMember	B	469	Use of Pointer Subtraction to Determine Size	699	1115
HasMember	B	476	NULL Pointer Dereference	699	1132
HasMember	V	587	Assignment of a Fixed Address to a Pointer	699	1322
HasMember	B	763	Release of Invalid Pointer or Reference	699	1599
HasMember	B	822	Untrusted Pointer Dereference	699	1723
HasMember	B	823	Use of Out-of-range Pointer Offset	699	1726
HasMember	B	824	Access of Uninitialized Pointer	699	1729
HasMember	B	825	Expired Pointer Dereference	699	1732

Category-485: 7PK - Encapsulation

Category ID : 485

Summary

This category represents one of the phyla in the Seven Pernicious Kingdoms vulnerability classification. It includes weaknesses that occur when the product does not sufficiently encapsulate critical data or functionality. According to the authors of the Seven Pernicious Kingdoms, "Encapsulation is about drawing strong boundaries. In a web browser that might mean ensuring that your mobile code cannot be abused by other mobile code. On the server it might mean differentiation between validated data and unvalidated data, between one user's data and another's, or between data users are allowed to see and data that they are not."

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	700	Seven Pernicious Kingdoms	700	2557
HasMember	V	486	Comparison of Classes by Name	700	1164

Nature	Type	ID	Name	V	Page
HasMember	B	488	Exposure of Data Element to Wrong Session	700	1169
HasMember	B	489	Active Debug Code	700	1171
HasMember	V	491	Public cloneable() Method Without Final ('Object Hijack')	700	1174
HasMember	V	492	Use of Inner Class Containing Sensitive Data	700	1175
HasMember	V	493	Critical Public Variable Without Final Modifier	700	1182
HasMember	V	495	Private Data Structure Returned From A Public Method	700	1189
HasMember	V	496	Public Data Assigned to Private Array-Typed Field	700	1192
HasMember	B	497	Exposure of Sensitive System Information to an Unauthorized Control Sphere	700	1193
HasMember	B	501	Trust Boundary Violation	700	1203

Notes

Other

The "encapsulation" term is used in multiple ways. Within some security sources, the term is used to describe the establishment of boundaries between different control spheres. Within general computing circles, it is more about hiding implementation details and maintainability than security. Even within the security usage, there is also a question of whether "encapsulation" encompasses the entire range of security problems.

References

[REF-6]Katrina Tsipenyuk, Brian Chess and Gary McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". NIST Workshop on Software Security Assurance Tools Techniques and Metrics. 2005 November 7. NIST. < https://samate.nist.gov/SSATTM_Content/papers/Seven%20Pernicious%20Kingdoms%20-%20Taxonomy%20of%20Sw%20Security%20Errors%20-%20Tsipenyuk%20-%20Chess%20-%20McGraw.pdf >.

Category-557: Concurrency Issues

Category ID : 557

Summary

Weaknesses in this category are related to concurrent use of shared resources.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	364	Signal Handler Race Condition	699	899
HasMember	B	366	Race Condition within a Thread	699	904
HasMember	B	367	Time-of-check Time-of-use (TOCTOU) Race Condition	699	906
HasMember	B	368	Context Switching Race Condition	699	912
HasMember	B	386	Symbolic Name not Mapping to Correct Object	699	942
HasMember	B	421	Race Condition During Access to Alternate Channel	699	1020
HasMember	B	663	Use of a Non-reentrant Function in a Concurrent Context	699	1452
HasMember	B	820	Missing Synchronization	699	1720
HasMember	B	821	Incorrect Synchronization	699	1722
HasMember	B	1058	Invokable Control Element in Multi-Thread Context with non-Final Static Storable or Member Element	699	1893
HasMember	B	1322	Use of Blocking Code in Single-threaded, Non-blocking Context	699	2207

Category-569: Expression Issues

Category ID : 569

Summary

Weaknesses in this category are related to incorrectly written expressions within code.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	480	Use of Incorrect Operator	699	1150
HasMember	B	570	Expression is Always False	699	1292
HasMember	B	571	Expression is Always True	699	1295
HasMember	B	783	Operator Precedence Logic Error	699	1650

Category-712: OWASP Top Ten 2007 Category A1 - Cross Site Scripting (XSS)

Category ID : 712

Summary

Weaknesses in this category are related to the A1 category in the OWASP Top Ten 2007.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	629	Weaknesses in OWASP Top Ten (2007)	629	2551
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	629	163

References

[REF-572]OWASP. "Top 10 2007-Cross Site Scripting". 2007. < http://www.owasp.org/index.php/Top_10_2007-A1 >.

Category-713: OWASP Top Ten 2007 Category A2 - Injection Flaws

Category ID : 713

Summary

Weaknesses in this category are related to the A2 category in the OWASP Top Ten 2007.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	629	Weaknesses in OWASP Top Ten (2007)	629	2551
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	629	145
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	629	201
HasMember	B	90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	629	212
HasMember	B	91	XML Injection (aka Blind XPath Injection)	629	215

Nature	Type	ID	Name	V	Page
HasMember	B	93	Improper Neutralization of CRLF Sequences ('CRLF Injection')	629	217

Category-714: OWASP Top Ten 2007 Category A3 - Malicious File Execution

Category ID : 714

Summary

Weaknesses in this category are related to the A3 category in the OWASP Top Ten 2007.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	629	Weaknesses in OWASP Top Ten (2007)	629	2551
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	629	151
HasMember	V	95	Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')	629	226
HasMember	V	98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')	629	236
HasMember	B	434	Unrestricted Upload of File with Dangerous Type	629	1048

Category-715: OWASP Top Ten 2007 Category A4 - Insecure Direct Object Reference

Category ID : 715

Summary

Weaknesses in this category are related to the A4 category in the OWASP Top Ten 2007.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	629	Weaknesses in OWASP Top Ten (2007)	629	2551
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	629	33
HasMember	B	472	External Control of Assumed-Immutable Web Parameter	629	1123
HasMember	B	639	Authorization Bypass Through User-Controlled Key	629	1406

References

[REF-528]OWASP. "Top 10 2007-Insecure Direct Object Reference". 2007. < http://www.owasp.org/index.php/Top_10_2007-A4 >.

Category-716: OWASP Top Ten 2007 Category A5 - Cross Site Request Forgery (CSRF)

Category ID : 716

Summary

Weaknesses in this category are related to the A5 category in the OWASP Top Ten 2007.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	629	Weaknesses in OWASP Top Ten (2007)	629	2551
HasMember		352	Cross-Site Request Forgery (CSRF)	629	868

References

[REF-574]OWASP. "Top 10 2007-Cross Site Request Forgery". 2007. < http://www.owasp.org/index.php/Top_10_2007-A5 >.

Category-717: OWASP Top Ten 2007 Category A6 - Information Leakage and Improper Error Handling

Category ID : 717

Summary

Weaknesses in this category are related to the A6 category in the OWASP Top Ten 2007.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	629	Weaknesses in OWASP Top Ten (2007)	629	2551
HasMember	C	200	Exposure of Sensitive Information to an Unauthorized Actor	629	504
HasMember	B	203	Observable Discrepancy	629	518
HasMember	B	209	Generation of Error Message Containing Sensitive Information	629	533
HasMember	B	215	Insertion of Sensitive Information Into Debugging Code	629	551

References

[REF-575]OWASP. "Top 10 2007-Information Leakage and Improper Error Handling". 2007. < http://www.owasp.org/index.php/Top_10_2007-A6 >.

Category-718: OWASP Top Ten 2007 Category A7 - Broken Authentication and Session Management

Category ID : 718

Summary

Weaknesses in this category are related to the A7 category in the OWASP Top Ten 2007.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	629	Weaknesses in OWASP Top Ten (2007)	629	2551
HasMember	C	287	Improper Authentication	629	692
HasMember	B	301	Reflection Attack in an Authentication Protocol	629	733
HasMember	C	522	Insufficiently Protected Credentials	629	1225

References

[REF-237]OWASP. "Top 10 2007-Broken Authentication and Session Management". 2007. < http://www.owasp.org/index.php/Top_10_2007-A7 >.

Category-719: OWASP Top Ten 2007 Category A8 - Insecure Cryptographic Storage

Category ID : 719

Summary

Weaknesses in this category are related to the A8 category in the OWASP Top Ten 2007.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	629	Weaknesses in OWASP Top Ten (2007)	629	2551
HasMember	C	311	Missing Encryption of Sensitive Data	629	757
HasMember	V	321	Use of Hard-coded Cryptographic Key	629	785
HasMember	B	325	Missing Cryptographic Step	629	794
HasMember	C	326	Inadequate Encryption Strength	629	796

References

[REF-577]OWASP. "Top 10 2007-Insecure Cryptographic Storage". 2007. < http://www.owasp.org/index.php/Top_10_2007-A8 >.

Category-720: OWASP Top Ten 2007 Category A9 - Insecure Communications

Category ID : 720

Summary

Weaknesses in this category are related to the A9 category in the OWASP Top Ten 2007.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	629	Weaknesses in OWASP Top Ten (2007)	629	2551
MemberOf	C	1346	OWASP Top Ten 2021 Category A02:2021 - Cryptographic Failures	1344	2488
HasMember	C	311	Missing Encryption of Sensitive Data	629	757
HasMember	V	321	Use of Hard-coded Cryptographic Key	629	785
HasMember	B	325	Missing Cryptographic Step	629	794
HasMember	C	326	Inadequate Encryption Strength	629	796

References

[REF-271]OWASP. "Top 10 2007-Insecure Communications". 2007. < http://www.owasp.org/index.php/Top_10_2007-A9 >.

Category-721: OWASP Top Ten 2007 Category A10 - Failure to Restrict URL Access

Category ID : 721

Summary

Weaknesses in this category are related to the A10 category in the OWASP Top Ten 2007.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	629	Weaknesses in OWASP Top Ten (2007)	629	2551
HasMember	C	285	Improper Authorization	629	684
HasMember	B	288	Authentication Bypass Using an Alternate Path or Channel	629	700
HasMember	B	425	Direct Request ('Forced Browsing')	629	1025

References

[REF-580]OWASP. "Top 10 2007-Failure to Restrict URL Access". 2007. < http://www.owasp.org/index.php/Top_10_2007-A10 >.

Category-722: OWASP Top Ten 2004 Category A1 - Unvalidated Input

Category ID : 722

Summary

Weaknesses in this category are related to the A1 category in the OWASP Top Ten 2004.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	711	Weaknesses in OWASP Top Ten (2004)	711	2559
HasMember	C	20	Improper Input Validation	711	20
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	711	145
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	711	163
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	711	201
HasMember	V	102	Struts: Duplicate Validation Forms	711	246
HasMember	V	103	Struts: Incomplete validate() Method Definition	711	248
HasMember	V	104	Struts: Form Bean Does Not Extend Validation Class	711	251
HasMember	V	106	Struts: Plug-in Framework not in Use	711	256
HasMember	V	109	Struts: Validator Turned Off	711	263
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	711	304
HasMember	B	166	Improper Handling of Missing Special Element	711	423
HasMember	B	167	Improper Handling of Additional Special Element	711	425
HasMember	B	179	Incorrect Behavior Order: Early Validation	711	448
HasMember	V	180	Incorrect Behavior Order: Validate Before Canonicalize	711	451
HasMember	V	181	Incorrect Behavior Order: Validate Before Filter	711	453
HasMember	B	182	Collapse of Data into Unsafe Value	711	455
HasMember	B	183	Permissive List of Allowed Inputs	711	458
HasMember	B	425	Direct Request ('Forced Browsing')	711	1025
HasMember	B	472	External Control of Assumed-Immutable Web Parameter	711	1123
HasMember	B	601	URL Redirection to Untrusted Site ('Open Redirect')	711	1345
HasMember	C	602	Client-Side Enforcement of Server-Side Security	711	1350

References

[REF-581]OWASP. "A1 Unvalidated Input". 2007. < http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=70827 >.

Category-723: OWASP Top Ten 2004 Category A2 - Broken Access Control

Category ID : 723

Summary

Weaknesses in this category are related to the A2 category in the OWASP Top Ten 2004.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	711	Weaknesses in OWASP Top Ten (2004)	711	2559
HasMember	V	9	J2EE Misconfiguration: Weak Access Permissions for EJB Methods	711	8
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	711	33
HasMember	B	41	Improper Resolution of Path Equivalence	711	86
HasMember	B	73	External Control of File Name or Path	711	132
HasMember	B	266	Incorrect Privilege Assignment	711	638
HasMember	B	268	Privilege Chaining	711	644
HasMember	C	275	Permission Issues	711	2317
HasMember	B	283	Unverified Ownership	711	678
HasMember	P	284	Improper Access Control	711	680
HasMember	C	285	Improper Authorization	711	684
HasMember	C	330	Use of Insufficiently Random Values	711	814
HasMember	B	425	Direct Request ('Forced Browsing')	711	1025
HasMember	V	525	Use of Web Browser Cache Containing Sensitive Information	711	1233
HasMember	B	551	Incorrect Behavior Order: Authorization Before Parsing and Canonicalization	711	1264
HasMember	V	556	ASP.NET Misconfiguration: Use of Identity Impersonation	711	1271
HasMember	B	639	Authorization Bypass Through User-Controlled Key	711	1406
HasMember	B	708	Incorrect Ownership Assignment	711	1548

References

[REF-582]OWASP. "A2 Broken Access Control". 2007. < http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=70827 >.

Category-724: OWASP Top Ten 2004 Category A3 - Broken Authentication and Session Management

Category ID : 724

Summary

Weaknesses in this category are related to the A3 category in the OWASP Top Ten 2004.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	711	Weaknesses in OWASP Top Ten (2004)	711	2559
HasMember	C	255	Credentials Management Errors	711	2315
HasMember	V	259	Use of Hard-coded Password	711	623
HasMember	C	287	Improper Authentication	711	692
HasMember	B	296	Improper Following of a Certificate's Chain of Trust	711	719
HasMember	V	298	Improper Validation of Certificate Expiration	711	726
HasMember	B	302	Authentication Bypass by Assumed-Immutable Data	711	735
HasMember	B	304	Missing Critical Step in Authentication	711	738
HasMember	B	307	Improper Restriction of Excessive Authentication Attempts	711	747
HasMember	B	309	Use of Password System for Primary Authentication	711	754
HasMember	C	345	Insufficient Verification of Data Authenticity	711	851
HasMember		384	Session Fixation	711	936
HasMember	B	521	Weak Password Requirements	711	1223
HasMember	C	522	Insufficiently Protected Credentials	711	1225
HasMember	V	525	Use of Web Browser Cache Containing Sensitive Information	711	1233
HasMember	B	613	Insufficient Session Expiration	711	1371
HasMember	B	620	Unverified Password Change	711	1383
HasMember	B	640	Weak Password Recovery Mechanism for Forgotten Password	711	1409
HasMember	B	798	Use of Hard-coded Credentials	711	1690

References

[REF-583]OWASP. "A3 Broken Authentication and Session Management". 2007. < http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=70827 >.

Category-725: OWASP Top Ten 2004 Category A4 - Cross-Site Scripting (XSS) Flaws

Category ID : 725

Summary

Weaknesses in this category are related to the A4 category in the OWASP Top Ten 2004.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	711	Weaknesses in OWASP Top Ten (2004)	711	2559
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	711	163
HasMember	V	644	Improper Neutralization of HTTP Headers for Scripting Syntax	711	1422

References

[REF-584]OWASP. "A4 Cross-Site Scripting (XSS) Flaws". 2007. < http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=70827 >.

Category-726: OWASP Top Ten 2004 Category A5 - Buffer Overflows

Category ID : 726

Summary

Weaknesses in this category are related to the A5 category in the OWASP Top Ten 2004.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	711	Weaknesses in OWASP Top Ten (2004)	711	2559
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	711	293
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	711	304
HasMember	B	134	Use of Externally-Controlled Format String	711	365

References

[REF-585]OWASP. "A5 Buffer Overflows". 2007. < http://sourceforge.net/project/showfiles.php? group_id=64424&package_id=70827 >.

Category-727: OWASP Top Ten 2004 Category A6 - Injection Flaws

Category ID : 727

Summary

Weaknesses in this category are related to the A6 category in the OWASP Top Ten 2004.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	711	Weaknesses in OWASP Top Ten (2004)	711	2559
HasMember	C	74	Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')	711	137
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	711	145
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	711	151
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	711	201
HasMember	B	91	XML Injection (aka Blind XPath Injection)	711	215
HasMember	V	95	Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')	711	226
HasMember	V	98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')	711	236
HasMember	B	117	Improper Output Neutralization for Logs	711	288

References

[REF-586]OWASP. "A6 Injection Flaws". 2007. < http://sourceforge.net/project/showfiles.php? group_id=64424&package_id=70827 >.

Category-728: OWASP Top Ten 2004 Category A7 - Improper Error Handling

Category ID : 728

Summary

Weaknesses in this category are related to the A7 category in the OWASP Top Ten 2004.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	711	Weaknesses in OWASP Top Ten (2004)	711	2559
HasMember	V	7	J2EE Misconfiguration: Missing Custom Error Page	711	4
HasMember	B	203	Observable Discrepancy	711	518
HasMember	B	209	Generation of Error Message Containing Sensitive Information	711	533
HasMember	C	228	Improper Handling of Syntactically Invalid Structure	711	568
HasMember	B	252	Unchecked Return Value	711	606
HasMember	C	389	Error Conditions, Return Values, Status Codes	711	2322
HasMember	B	390	Detection of Error Condition Without Action	711	943
HasMember	B	391	Unchecked Error Condition	711	948
HasMember	B	394	Unexpected Status Code or Return Value	711	955
HasMember	C	636	Not Failing Securely ('Failing Open')	711	1401

References

[REF-587]OWASP. "A7 Improper Error Handling". 2007. < http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=70827 >.

Category-729: OWASP Top Ten 2004 Category A8 - Insecure Storage

Category ID : 729

Summary

Weaknesses in this category are related to the A8 category in the OWASP Top Ten 2004.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	711	Weaknesses in OWASP Top Ten (2004)	711	2559
HasMember	V	14	Compiler Removal of Code to Clear Buffers	711	14
HasMember	B	226	Sensitive Information in Resource Not Removed Before Reuse	711	562
HasMember	B	261	Weak Encoding for Password	711	631
HasMember	C	311	Missing Encryption of Sensitive Data	711	757
HasMember	V	321	Use of Hard-coded Cryptographic Key	711	785
HasMember	C	326	Inadequate Encryption Strength	711	796
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	711	799
HasMember	V	539	Use of Persistent Cookies Containing Sensitive Information	711	1250
HasMember	V	591	Sensitive Data Storage in Improperly Locked Memory	711	1329
HasMember	V	598	Use of GET Request Method With Sensitive Query Strings	711	1340

References

[REF-588]OWASP. "A8 Insecure Storage". 2007. < http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=70827 >.

Category-730: OWASP Top Ten 2004 Category A9 - Denial of Service

Category ID : 730

Summary

Weaknesses in this category are related to the A9 category in the OWASP Top Ten 2004.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	711	Weaknesses in OWASP Top Ten (2004)	711	2559
HasMember	B	170	Improper Null Termination	711	428
HasMember	B	248	Uncaught Exception	711	596
HasMember	B	369	Divide By Zero	711	913
HasMember	V	382	J2EE Bad Practices: Use of System.exit()	711	933
HasMember	C	400	Uncontrolled Resource Consumption	711	964
HasMember	V	401	Missing Release of Memory after Effective Lifetime	711	973
HasMember	C	404	Improper Resource Shutdown or Release	711	980
HasMember	C	405	Asymmetric Resource Consumption (Amplification)	711	986
HasMember	B	410	Insufficient Resource Pool	711	998
HasMember	B	412	Unrestricted Externally Accessible Lock	711	1000
HasMember	B	476	NULL Pointer Dereference	711	1132
HasMember	C	674	Uncontrolled Recursion	711	1484

References

[REF-590]OWASP. "A9 Denial of Service". 2007. < http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=70827 >.

Category-731: OWASP Top Ten 2004 Category A10 - Insecure Configuration Management

Category ID : 731

Summary

Weaknesses in this category are related to the A10 category in the OWASP Top Ten 2004.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	711	Weaknesses in OWASP Top Ten (2004)	711	2559
HasMember	V	5	J2EE Misconfiguration: Data Transmission Without Encryption	711	1
HasMember	V	6	J2EE Misconfiguration: Insufficient Session-ID Length	711	2
HasMember	V	7	J2EE Misconfiguration: Missing Custom Error Page	711	4
HasMember	V	8	J2EE Misconfiguration: Entity Bean Declared Remote	711	6
HasMember	V	9	J2EE Misconfiguration: Weak Access Permissions for EJB Methods	711	8
HasMember	V	11	ASP.NET Misconfiguration: Creating Debug Binary	711	9
HasMember	V	12	ASP.NET Misconfiguration: Missing Custom Error Page	711	11
HasMember	V	13	ASP.NET Misconfiguration: Password in Configuration File	711	13
HasMember	B	209	Generation of Error Message Containing Sensitive Information	711	533

Nature	Type	ID	Name	V	Page
HasMember	B	215	Insertion of Sensitive Information Into Debugging Code	711	551
HasMember	V	219	Storage of File with Sensitive Data Under Web Root	711	553
HasMember	C	275	Permission Issues	711	2317
HasMember	B	295	Improper Certificate Validation	711	714
HasMember	B	459	Incomplete Cleanup	711	1099
HasMember	B	489	Active Debug Code	711	1171
HasMember	V	520	.NET Misconfiguration: Use of Impersonation	711	1222
HasMember	V	526	Cleartext Storage of Sensitive Information in an Environment Variable	711	1234
HasMember	V	527	Exposure of Version-Control Repository to an Unauthorized Control Sphere	711	1236
HasMember	V	528	Exposure of Core Dump File to an Unauthorized Control Sphere	711	1237
HasMember	V	529	Exposure of Access Control List Files to an Unauthorized Control Sphere	711	1238
HasMember	V	530	Exposure of Backup File to an Unauthorized Control Sphere	711	1239
HasMember	V	531	Inclusion of Sensitive Information in Test Code	711	1240
HasMember	B	532	Insertion of Sensitive Information into Log File	711	1241
HasMember	B	540	Inclusion of Sensitive Information in Source Code	711	1251
HasMember	V	541	Inclusion of Sensitive Information in an Include File	711	1253
HasMember	V	548	Exposure of Information Through Directory Listing	711	1261
HasMember	B	552	Files or Directories Accessible to External Parties	711	1265
HasMember	V	554	ASP.NET Misconfiguration: Not Using Input Validation Framework	711	1269
HasMember	V	555	J2EE Misconfiguration: Plaintext Password in Configuration File	711	1270
HasMember	V	556	ASP.NET Misconfiguration: Use of Identity Impersonation	711	1271

References

[REF-591]OWASP. "A10 Insecure Configuration Management". 2007. < http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=70827 >.

Category-735: CERT C Secure Coding Standard (2008) Chapter 2 - Preprocessor (PRE)

Category ID : 735

Summary

Weaknesses in this category are related to the rules and recommendations in the Preprocessor (PRE) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	C	684	Incorrect Provision of Specified Functionality	734	1505

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-684 PRE09-C Do not replace secure functions with less secure functions

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-736: CERT C Secure Coding Standard (2008) Chapter 3 - Declarations and Initialization (DCL)

Category ID : 736

Summary

Weaknesses in this category are related to the rules and recommendations in the Declarations and Initialization (DCL) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	B	547	Use of Hard-coded, Security-relevant Constants	734	1259
HasMember	B	628	Function Call with Incorrectly Specified Arguments	734	1398
HasMember	V	686	Function Call With Incorrect Argument Type	734	1508

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-547 DCL06-C Use meaningful symbolic constants to represent literal values in program logic CWE-628 DCL10-C Maintain the contract between the writer and caller of variadic functions CWE-686 DCL35-C Do not invoke a function using a type that does not match the function definition

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-737: CERT C Secure Coding Standard (2008) Chapter 4 - Expressions (EXP)

Category ID : 737

Summary

Weaknesses in this category are related to the rules and recommendations in the Expressions (EXP) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	V	467	Use of sizeof() on a Pointer Type	734	1110
HasMember	B	468	Incorrect Pointer Scaling	734	1114
HasMember	B	476	NULL Pointer Dereference	734	1132
HasMember	B	628	Function Call with Incorrectly Specified Arguments	734	1398
HasMember	C	704	Incorrect Type Conversion or Cast	734	1538
HasMember	B	783	Operator Precedence Logic Error	734	1650

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-467 EXP01-C Do not take the size of a pointer to determine the size of the pointed-to type CWE-468 EXP08-C Ensure pointer arithmetic is used correctly CWE-476 EXP34-C Ensure a null pointer is not dereferenced CWE-628 EXP37-C Call functions with the arguments intended by the API CWE-704 EXP05-C Do not cast away a const qualification CWE-783 EXP00-C Use parentheses for precedence of operation

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-738: CERT C Secure Coding Standard (2008) Chapter 5 - Integers (INT)

Category ID : 738

Summary

Weaknesses in this category are related to the rules and recommendations in the Integers (INT) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	C	20	Improper Input Validation	734	20
HasMember	V	129	Improper Validation of Array Index	734	341
HasMember	B	190	Integer Overflow or Wraparound	734	472
HasMember	V	192	Integer Coercion Error	734	482
HasMember	B	197	Numeric Truncation Error	734	500
HasMember	B	369	Divide By Zero	734	913
HasMember	B	466	Return of Pointer Value Outside of Expected Range	734	1109
HasMember	V	587	Assignment of a Fixed Address to a Pointer	734	1322
HasMember	B	606	Unchecked Input for Loop Condition	734	1357
HasMember	B	676	Use of Potentially Dangerous Function	734	1489
HasMember	B	681	Incorrect Conversion between Numeric Types	734	1495
HasMember	P	682	Incorrect Calculation	734	1499

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-20 INT06-C Use strtol() or a related function to convert a string token to an integer CWE-129 INT32-C Ensure that operations on signed integers do not result in overflow CWE-190 INT03-C Use a secure integer library CWE-190 INT30-C Ensure that unsigned integer operations do not wrap CWE-190 INT32-C Ensure that operations on signed integers do not result in overflow CWE-190 INT35-C Evaluate integer expressions in a larger size before comparing or assigning to that size CWE-192 INT02-C Understand integer conversion rules CWE-192 INT05-C Do not use input functions to convert character data if they cannot handle all possible inputs CWE-192 INT31-C Ensure that integer conversions do not result in lost or misinterpreted data CWE-197 INT02-C Understand integer conversion rules CWE-197 INT05-C Do not use input functions to convert character data if they cannot handle all possible inputs CWE-197 INT31-C Ensure that integer conversions do not result in lost or misinterpreted data CWE-369 INT33-C Ensure that division and modulo operations do not result in divide-by-zero errors CWE-466 INT11-C Take care when converting from pointer to integer or integer to pointer CWE-587 INT11-C Take care when converting from pointer to integer or integer to pointer CWE-606 INT03-C Use a secure integer library CWE-676 INT06-C Use strtol() or a related function to convert a string token to an integer CWE-681 INT15-C Use intmax_t or uintmax_t for formatted IO on programmer-defined integer types CWE-681 INT31-C Ensure that integer conversions do not result in lost or misinterpreted data CWE-681 INT35-C Evaluate integer expressions in a larger size before comparing or assigning to that size CWE-682 INT07-C Use only explicitly signed or unsigned char type for numeric values CWE-682 INT13-C Use bitwise operators only on unsigned operands

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-739: CERT C Secure Coding Standard (2008) Chapter 6 - Floating Point (FLP)

Category ID : 739

Summary

Weaknesses in this category are related to the rules and recommendations in the Floating Point (FLP) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	B	369	Divide By Zero	734	913
HasMember	B	681	Incorrect Conversion between Numeric Types	734	1495
HasMember	P	682	Incorrect Calculation	734	1499
HasMember	V	686	Function Call With Incorrect Argument Type	734	1508

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-369 FLP03-C Detect and handle floating point errors CWE-681 FLP33-C Convert integers to floating point for floating point operations CWE-681 FLP34-C Ensure that floating point conversions are within range of the new type CWE-682 FLP32-C Prevent or detect domain and range errors in math functions CWE-682 FLP33-C Convert integers to floating point for floating point operations CWE-686 FLP31-C Do not call functions expecting real values with complex values

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-740: CERT C Secure Coding Standard (2008) Chapter 7 - Arrays (ARR)

Category ID : 740

Summary

Weaknesses in this category are related to the rules and recommendations in the Arrays (ARR) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	734	293
HasMember	V	129	Improper Validation of Array Index	734	341
HasMember	V	467	Use of sizeof() on a Pointer Type	734	1110
HasMember	B	469	Use of Pointer Subtraction to Determine Size	734	1115
HasMember	C	665	Improper Initialization	734	1456
HasMember	B	805	Buffer Access with Incorrect Length Value	734	1702

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-119 ARR00-C Understand how arrays work CWE-119 ARR33-C Guarantee that copies are made into storage of sufficient size CWE-119 ARR34-C Ensure that array types in expressions are compatible CWE-119 ARR35-C Do not allow loops to iterate beyond the end of an array CWE-129 ARR00-C Understand how arrays work CWE-129 ARR30-C Guarantee that array indices are within the valid range CWE-129 ARR38-C Do not add or subtract an integer to a pointer if the resulting value does not refer to a valid array element CWE-467 ARR01-C Do not apply the sizeof operator to a pointer when taking the size of an array CWE-469 ARR36-C Do not subtract or compare two pointers that do not refer to the same array CWE-469 ARR37-C Do not add or subtract an integer to a pointer to a non-array object CWE-665 ARR02-C Explicitly specify array bounds, even if implicitly defined by an initializer CWE-805 ARR33-C Guarantee that copies are made into storage of sufficient size

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-741: CERT C Secure Coding Standard (2008) Chapter 8 - Characters and Strings (STR)

Category ID : 741

Summary

Weaknesses in this category are related to the rules and recommendations in the Characters and Strings (STR) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	734	151
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	734	194
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	734	293
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	734	304
HasMember	B	135	Incorrect Calculation of Multi-Byte String Length	734	370
HasMember	B	170	Improper Null Termination	734	428
HasMember	B	193	Off-by-one Error	734	486
HasMember	B	464	Addition of Data Structure Sentinel	734	1107
HasMember	V	686	Function Call With Incorrect Argument Type	734	1508
HasMember	C	704	Incorrect Type Conversion or Cast	734	1538

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-78 STR02-C Sanitize data passed to complex subsystems CWE-88 STR02-C Sanitize data passed to complex subsystems CWE-119 STR31-C Guarantee that storage for strings has sufficient space for character data and the null terminator CWE-119 STR32-C Null-terminate byte strings as required CWE-119 STR33-C Size wide character strings correctly CWE-120 STR35-C Do not copy data from an unbounded source to a fixed-length array CWE-135 STR33-C Size wide character strings correctly CWE-170 STR03-C Do not inadvertently truncate a null-terminated byte string CWE-170 STR32-C Null-terminate byte strings as required CWE-193 STR31-C Guarantee that storage for strings has sufficient space for character data and the null terminator CWE-464 STR03-C Do not inadvertently truncate a null-terminated byte string CWE-464 STR06-C Do not assume that strtok() leaves the parse string unchanged CWE-686 STR37-C Arguments to character handling functions must be representable as an unsigned char CWE-704 STR34-C Cast characters to unsigned types before converting to larger integer sizes CWE-704 STR37-C Arguments to character handling functions must be representable as an unsigned char

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-742: CERT C Secure Coding Standard (2008) Chapter 9 - Memory Management (MEM)

Category ID : 742

Summary

Weaknesses in this category are related to the rules and recommendations in the Memory Management (MEM) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	C	20	Improper Input Validation	734	20
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	734	293
HasMember	B	128	Wrap-around Error	734	339
HasMember	B	131	Incorrect Calculation of Buffer Size	734	355
HasMember	B	190	Integer Overflow or Wraparound	734	472
HasMember	B	226	Sensitive Information in Resource Not Removed Before Reuse	734	562
HasMember	V	244	Improper Clearing of Heap Memory Before Release ('Heap Inspection')	734	591
HasMember	B	252	Unchecked Return Value	734	606
HasMember	V	415	Double Free	734	1008
HasMember	V	416	Use After Free	734	1012
HasMember	B	476	NULL Pointer Dereference	734	1132
HasMember	V	528	Exposure of Core Dump File to an Unauthorized Control Sphere	734	1237
HasMember	V	590	Free of Memory not on the Heap	734	1326
HasMember	V	591	Sensitive Data Storage in Improperly Locked Memory	734	1329
HasMember	B	628	Function Call with Incorrectly Specified Arguments	734	1398
HasMember	C	665	Improper Initialization	734	1456
HasMember	V	687	Function Call With Incorrectly Specified Argument Value	734	1510
HasMember	C	754	Improper Check for Unusual or Exceptional Conditions	734	1568

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-20 MEM10-C Define and use a pointer validation function CWE-119 MEM09-C Do not assume memory allocation routines initialize memory CWE-128 MEM07-C Ensure that the arguments to calloc(), when multiplied, can be represented as a size_t CWE-131 MEM35-C Allocate sufficient memory for an object CWE-190 MEM07-C Ensure that the arguments to calloc(), when multiplied, can be represented as a size_t CWE-190 MEM35-C Allocate sufficient memory for an object CWE-226 MEM03-C Clear sensitive information stored in reusable resources returned for reuse CWE-244 MEM03-C Clear sensitive information stored in reusable resources returned for reuse CWE-252 MEM32-C Detect and handle memory allocation errors CWE-415 MEM00-C Allocate and free memory in the same module, at the same level of abstraction CWE-415 MEM01-C Store a new value in pointers immediately after free() CWE-415 MEM31-C Free dynamically allocated memory exactly once CWE-416 MEM00-C Allocate and free memory in the same module, at the same level of abstraction CWE-416 MEM01-C Store a new value in pointers immediately after free() CWE-416 MEM30-C Do not access freed memory CWE-476 MEM32-C Detect and handle memory allocation errors CWE-528 MEM06-C Ensure that sensitive data is not written out to disk CWE-590 MEM34-C Only free memory allocated dynamically CWE-591 MEM06-C Ensure that sensitive data is not written out to disk CWE-628 MEM08-C Use realloc() only to resize dynamically allocated arrays CWE-665 MEM09-C Do not assume memory allocation routines initialize memory CWE-687 MEM04-C Do not perform zero length allocations CWE-754 MEM32-C Detect and handle memory allocation errors

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-743: CERT C Secure Coding Standard (2008) Chapter 10 - Input Output (FIO)

Category ID : 743

Summary

Weaknesses in this category are related to the rules and recommendations in the Input Output (FIO) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	734	33
HasMember	V	37	Path Traversal: '/absolute/pathname/here'	734	79
HasMember	V	38	Path Traversal: '\absolute\pathname\here'	734	80
HasMember	V	39	Path Traversal: 'C:dirname'	734	82
HasMember	B	41	Improper Resolution of Path Equivalence	734	86
HasMember	B	59	Improper Link Resolution Before File Access ('Link Following')	734	111
HasMember	V	62	UNIX Hard Link	734	119
HasMember	V	64	Windows Shortcut Following (.LNK)	734	121
HasMember	V	65	Windows Hard Link	734	123
HasMember	V	67	Improper Handling of Windows Device Names	734	126
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	734	293
HasMember	B	134	Use of Externally-Controlled Format String	734	365
HasMember	B	241	Improper Handling of Unexpected Data Type	734	584
HasMember	B	276	Incorrect Default Permissions	734	665
HasMember	V	279	Incorrect Execution-Assigned Permissions	734	671
HasMember	C	362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	734	888
HasMember	B	367	Time-of-check Time-of-use (TOCTOU) Race Condition	734	906
HasMember	B	379	Creation of Temporary File in Directory with Insecure Permissions	734	930
HasMember	B	391	Unchecked Error Condition	734	948
HasMember	B	403	Exposure of File Descriptor to Unintended Control Sphere ('File Descriptor Leak')	734	978
HasMember	C	404	Improper Resource Shutdown or Release	734	980
HasMember	B	552	Files or Directories Accessible to External Parties	734	1265
HasMember	C	675	Multiple Operations on Resource in Single-Operation Context	734	1487
HasMember	B	676	Use of Potentially Dangerous Function	734	1489
HasMember	V	686	Function Call With Incorrect Argument Type	734	1508
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	734	1551

CWE-743: CERT C Secure Coding Standard (2008) Chapter 10 - Input Output (FIO)

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-22 FIO02-C Canonicalize path names originating from untrusted sources CWE-37 FIO05-C Identify files using multiple file attributes CWE-38 FIO05-C Identify files using multiple file attributes CWE-39 FIO05-C Identify files using multiple file attributes CWE-41 FIO02-C Canonicalize path names originating from untrusted sources CWE-59 FIO02-C Canonicalize path names originating from untrusted sources CWE-62 FIO05-C Identify files using multiple file attributes CWE-64 FIO05-C Identify files using multiple file attributes CWE-65 FIO05-C Identify files using multiple file attributes CWE-67 FIO32-C Do not perform operations on devices that are only appropriate for files CWE-119 FIO37-C Do not assume character data has been read CWE-134 FIO30-C Exclude user input from format strings CWE-134 FIO30-C Exclude user input from format strings CWE-241 FIO37-C Do not assume character data has been read CWE-276 FIO06-C Create files with appropriate access permissions CWE-279 FIO06-C Create files with appropriate access permissions CWE-362 FIO31-C Do not simultaneously open the same file multiple times CWE-367 FIO01-C Be careful using functions that use file names for identification CWE-379 FIO15-C Ensure that file operations are performed in a secure directory CWE-379 FIO43-C Do not create temporary files in shared directories CWE-391 FIO04-C Detect and handle input and output errors CWE-391 FIO33-C Detect and handle input output errors resulting in undefined behavior CWE-403 FIO42-C Ensure files are properly closed when they are no longer needed CWE-404 FIO42-C Ensure files are properly closed when they are no longer needed CWE-552 FIO15-C Ensure that file operations are performed in a secure directory CWE-675 FIO31-C Do not simultaneously open the same file multiple times CWE-676 FIO01-C Be careful using functions that use file names for identification CWE-686 FIO00-C Take care when creating format strings CWE-732 FIO06-C Create files with appropriate access permissions

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-744: CERT C Secure Coding Standard (2008) Chapter 11 - Environment (ENV)

Category ID : 744

Summary

Weaknesses in this category are related to the rules and recommendations in the Environment (ENV) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	734	151
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	734	194
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	734	293
HasMember	B	426	Untrusted Search Path	734	1028
HasMember	V	462	Duplicate Key in Associative List (Alist)	734	1104

Nature	Type	ID	Name	V	Page
HasMember	C	705	Incorrect Control Flow Scoping	734	1542

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-78 ENV03-C Sanitize the environment when invoking external programs CWE-78 ENV04-C Do not call system() if you do not need a command processor CWE-88 ENV03-C Sanitize the environment when invoking external programs CWE-88 ENV04-C Do not call system() if you do not need a command processor CWE-119 ENV01-C Do not make assumptions about the size of an environment variable CWE-426 ENV03-C Sanitize the environment when invoking external programs CWE-462 ENV02-C Beware of multiple environment variables with the same effective name CWE-705 ENV32-C All atexit handlers must return normally

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-745: CERT C Secure Coding Standard (2008) Chapter 12 - Signals (SIG)

Category ID : 745

Summary

Weaknesses in this category are related to the rules and recommendations in the Signals (SIG) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	V	479	Signal Handler Use of a Non-reentrant Function	734	1147
HasMember	C	662	Improper Synchronization	734	1448

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-432 SIG00-C Mask signals handled by noninterruptible signal handlers CWE-479 SIG30-C Call only asynchronous-safe functions within signal handlers CWE-479 SIG32-C Do not call longjmp() from inside a signal handler CWE-479 SIG33-C Do not recursively invoke the raise() function CWE-479 SIG34-C Do not call signal() from within interruptible signal handlers CWE-662 SIG00-C Mask signals handled by noninterruptible signal handlers CWE-662 SIG31-C Do not access or modify shared objects in signal handlers CWE-828 SIG31-C Do not access or modify shared objects in signal handlers

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-746: CERT C Secure Coding Standard (2008) Chapter 13 - Error Handling (ERR)

Category ID : 746

Summary

Weaknesses in this category are related to the rules and recommendations in the Error Handling (ERR) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	C	20	Improper Input Validation	734	20
HasMember	B	391	Unchecked Error Condition	734	948
HasMember	B	544	Missing Standardized Error Handling Mechanism	734	1256
HasMember	B	676	Use of Potentially Dangerous Function	734	1489
HasMember	C	705	Incorrect Control Flow Scoping	734	1542

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-20 ERR07-C Prefer functions that support error checking over equivalent functions that don't CWE-391 ERR00-C Adopt and implement a consistent and comprehensive error-handling policy CWE-544 ERR00-C Adopt and implement a consistent and comprehensive error-handling policy CWE-676 ERR07-C Prefer functions that support error checking over equivalent functions that don't CWE-705 ERR04-C Choose an appropriate termination strategy

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-747: CERT C Secure Coding Standard (2008) Chapter 14 - Miscellaneous (MSC)

Category ID : 747

Summary

Weaknesses in this category are related to the rules and recommendations in the Miscellaneous (MSC) chapter of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	V	14	Compiler Removal of Code to Clear Buffers	734	14
HasMember	C	20	Improper Input Validation	734	20
HasMember	V	176	Improper Handling of Unicode Encoding	734	440
HasMember	C	330	Use of Insufficiently Random Values	734	814
HasMember	B	480	Use of Incorrect Operator	734	1150

Nature	Type	ID	Name	V	Page
HasMember	V	482	Comparing instead of Assigning	734	1157
HasMember	B	561	Dead Code	734	1275
HasMember	B	563	Assignment to Variable without Use	734	1280
HasMember	B	570	Expression is Always False	734	1292
HasMember	B	571	Expression is Always True	734	1295
HasMember	P	697	Incorrect Comparison	734	1530
HasMember	C	704	Incorrect Type Conversion or Cast	734	1538

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-14 MSC06-C Be aware of compiler optimization when dealing with sensitive data CWE-20 MSC08-C Library functions should validate their parameters CWE-176 MSC10-C Character Encoding - UTF8 Related Issues CWE-330 MSC30-C Do not use the rand() function for generating pseudorandom numbers CWE-480 MSC02-C Avoid errors of omission CWE-480 MSC03-C Avoid errors of addition CWE-482 MSC02-C Avoid errors of omission CWE-561 MSC07-C Detect and remove dead code CWE-563 MSC00-C Compile cleanly at high warning levels CWE-570 MSC00-C Compile cleanly at high warning levels CWE-571 MSC00-C Compile cleanly at high warning levels CWE-697 MSC31-C Ensure that return values are compared against the proper type CWE-704 MSC31-C Ensure that return values are compared against the proper type CWE-758 MSC14-C Do not introduce unnecessary platform dependencies CWE-758 MSC15-C Do not depend on undefined behavior

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-748: CERT C Secure Coding Standard (2008) Appendix - POSIX (POS)

Category ID : 748

Summary

Weaknesses in this category are related to the rules and recommendations in the POSIX (POS) appendix of the CERT C Secure Coding Standard (2008).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	734	Weaknesses Addressed by the CERT C Secure Coding Standard (2008)	734	2560
HasMember	B	59	Improper Link Resolution Before File Access ('Link Following')	734	111
HasMember	B	170	Improper Null Termination	734	428
HasMember	B	242	Use of Inherently Dangerous Function	734	586
HasMember	B	272	Least Privilege Violation	734	656
HasMember	B	273	Improper Check for Dropped Privileges	734	660
HasMember	B	363	Race Condition Enabling Link Following	734	897
HasMember	B	366	Race Condition within a Thread	734	904
HasMember	B	562	Return of Stack Variable Address	734	1278
HasMember	C	667	Improper Locking	734	1464

Nature	Type	ID	Name	V	Page
HasMember	V	686	Function Call With Incorrect Argument Type	734	1508
HasMember	C	696	Incorrect Behavior Order	734	1527

Notes

Relationship

In the 2008 version of the CERT C Secure Coding standard, the following rules were mapped to the following CWE IDs: CWE-59 POS01-C Check for the existence of links when dealing with files CWE-170 POS30-C Use the readlink() function properly CWE-242 POS33-C Do not use vfork() CWE-272 POS02-C Follow the principle of least privilege CWE-273 POS37-C Ensure that privilege relinquishment is successful CWE-363 POS35-C Avoid race conditions while checking for the existence of a symbolic link CWE-366 POS00-C Avoid race conditions with multiple threads CWE-562 POS34-C Do not call putenv() with a pointer to an automatic variable as the argument CWE-667 POS31-C Do not unlock or destroy another thread's mutex CWE-686 POS34-C Do not call putenv() with a pointer to an automatic variable as the argument CWE-696 POS36-C Observe correct revocation order while relinquishing privileges

References

[REF-597]Robert C. Seacord. "The CERT C Secure Coding Standard". 1st Edition. 2008 October 4. Addison-Wesley Professional.

Category-751: 2009 Top 25 - Insecure Interaction Between Components

Category ID : 751

Summary

Weaknesses in this category are listed in the "Insecure Interaction Between Components" section of the 2009 CWE/SANS Top 25 Programming Errors.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	750	Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors	750	2562
HasMember	C	20	Improper Input Validation	750	20
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	750	151
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	750	163
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	750	201
HasMember	C	116	Improper Encoding or Escaping of Output	750	281
HasMember	B	209	Generation of Error Message Containing Sensitive Information	750	533
HasMember	B	319	Cleartext Transmission of Sensitive Information	750	779
HasMember		352	Cross-Site Request Forgery (CSRF)	750	868
HasMember	C	362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	750	888

References

[REF-615]"2009 CWE/SANS Top 25 Most Dangerous Programming Errors". 2009 January 2. < http://cwe.mitre.org/top25/archive/2009/2009_cwe_sans_top25.html >.

Category-752: 2009 Top 25 - Risky Resource Management

Category ID : 752

Summary

Weaknesses in this category are listed in the "Risky Resource Management" section of the 2009 CWE/SANS Top 25 Programming Errors.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	750	Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors	750	2562
HasMember	B	73	External Control of File Name or Path	750	132
HasMember	B	94	Improper Control of Generation of Code ('Code Injection')	750	219
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	750	293
HasMember	C	404	Improper Resource Shutdown or Release	750	980
HasMember	B	426	Untrusted Search Path	750	1028
HasMember	B	494	Download of Code Without Integrity Check	750	1185
HasMember	C	642	External Control of Critical State Data	750	1414
HasMember	C	665	Improper Initialization	750	1456
HasMember	P	682	Incorrect Calculation	750	1499

References

[REF-615]"2009 CWE/SANS Top 25 Most Dangerous Programming Errors". 2009 January 2. < http://cwe.mitre.org/top25/archive/2009/2009_cwe_sans_top25.html >.

Category-753: 2009 Top 25 - Porous Defenses

Category ID : 753

Summary

Weaknesses in this category are listed in the "Porous Defenses" section of the 2009 CWE/SANS Top 25 Programming Errors.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	750	Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors	750	2562
HasMember	B	250	Execution with Unnecessary Privileges	750	599
HasMember	V	259	Use of Hard-coded Password	750	623
HasMember	C	285	Improper Authorization	750	684
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	750	799
HasMember	C	330	Use of Insufficiently Random Values	750	814
HasMember	C	602	Client-Side Enforcement of Server-Side Security	750	1350
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	750	1551
HasMember	B	798	Use of Hard-coded Credentials	750	1690

References

[REF-615]"2009 CWE/SANS Top 25 Most Dangerous Programming Errors". 2009 January 2. < http://cwe.mitre.org/top25/archive/2009/2009_cwe_sans_top25.html >.

Category-801: 2010 Top 25 - Insecure Interaction Between Components

Category ID : 801

Summary

Weaknesses in this category are listed in the "Insecure Interaction Between Components" section of the 2010 CWE/SANS Top 25 Programming Errors.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	800	Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors	800	2563
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	800	151
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	800	163
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	800	201
HasMember	B	209	Generation of Error Message Containing Sensitive Information	800	533
HasMember		352	Cross-Site Request Forgery (CSRF)	800	868
HasMember	C	362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	800	888
HasMember	B	434	Unrestricted Upload of File with Dangerous Type	800	1048
HasMember	B	601	URL Redirection to Untrusted Site ('Open Redirect')	800	1345

References

[REF-732]"2010 CWE/SANS Top 25 Most Dangerous Software Errors". 2010 February 4. < http://cwe.mitre.org/top25/archive/2010/2010_cwe_sans_top25.html >.

Category-802: 2010 Top 25 - Risky Resource Management

Category ID : 802

Summary

Weaknesses in this category are listed in the "Risky Resource Management" section of the 2010 CWE/SANS Top 25 Programming Errors.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	800	Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors	800	2563
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	800	33
HasMember	V	98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')	800	236

Nature	Type	ID	Name	V	Page
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	800	304
HasMember	V	129	Improper Validation of Array Index	800	341
HasMember	B	131	Incorrect Calculation of Buffer Size	800	355
HasMember	B	190	Integer Overflow or Wraparound	800	472
HasMember	B	494	Download of Code Without Integrity Check	800	1185
HasMember	C	754	Improper Check for Unusual or Exceptional Conditions	800	1568
HasMember	B	770	Allocation of Resources Without Limits or Throttling	800	1613
HasMember	B	805	Buffer Access with Incorrect Length Value	800	1702

References

[REF-732]"2010 CWE/SANS Top 25 Most Dangerous Software Errors". 2010 February 4. < http://cwe.mitre.org/top25/archive/2010/2010_cwe_sans_top25.html >.

Category-803: 2010 Top 25 - Porous Defenses

Category ID : 803

Summary

Weaknesses in this category are listed in the "Porous Defenses" section of the 2010 CWE/SANS Top 25 Programming Errors.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	800	Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors	800	2563
HasMember	C	285	Improper Authorization	800	684
HasMember	B	306	Missing Authentication for Critical Function	800	741
HasMember	C	311	Missing Encryption of Sensitive Data	800	757
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	800	799
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	800	1551
HasMember	B	798	Use of Hard-coded Credentials	800	1690
HasMember	B	807	Reliance on Untrusted Inputs in a Security Decision	800	1714

References

[REF-732]"2010 CWE/SANS Top 25 Most Dangerous Software Errors". 2010 February 4. < http://cwe.mitre.org/top25/archive/2010/2010_cwe_sans_top25.html >.

Category-808: 2010 Top 25 - Weaknesses On the Cusp

Category ID : 808

Summary

Weaknesses in this category are not part of the general Top 25, but they were part of the original nominee list from which the Top 25 was drawn.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	800	Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors	800	2563
HasMember	B	59	Improper Link Resolution Before File Access ('Link Following')	800	111
HasMember	B	134	Use of Externally-Controlled Format String	800	365
HasMember	B	212	Improper Removal of Sensitive Information Before Storage or Transfer	800	544
HasMember	B	307	Improper Restriction of Excessive Authentication Attempts	800	747
HasMember	C	330	Use of Insufficiently Random Values	800	814
HasMember	V	416	Use After Free	800	1012
HasMember	B	426	Untrusted Search Path	800	1028
HasMember	B	454	External Initialization of Trusted Variables or Data Stores	800	1085
HasMember	V	456	Missing Initialization of a Variable	800	1089
HasMember	B	476	NULL Pointer Dereference	800	1132
HasMember	C	672	Operation on a Resource after Expiration or Release	800	1479
HasMember	B	681	Incorrect Conversion between Numeric Types	800	1495
HasMember	B	749	Exposed Dangerous Method or Function	800	1564
HasMember	B	772	Missing Release of Resource after Effective Lifetime	800	1624
HasMember	C	799	Improper Control of Interaction Frequency	800	1699
HasMember	B	804	Guessable CAPTCHA	800	1701

References

[REF-732]"2010 CWE/SANS Top 25 Most Dangerous Software Errors". 2010 February 4. < http://cwe.mitre.org/top25/archive/2010/2010_cwe_sans_top25.html >.

Category-810: OWASP Top Ten 2010 Category A1 - Injection

Category ID : 810

Summary

Weaknesses in this category are related to the A1 category in the OWASP Top Ten 2010.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	809	Weaknesses in OWASP Top Ten (2010)	809	2563
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	809	151
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	809	194
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	809	201
HasMember	B	90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	809	212
HasMember	B	91	XML Injection (aka Blind XPath Injection)	809	215

References

[REF-761]OWASP. "Top 10 2010-A1-Injection". < http://www.owasp.org/index.php/Top_10_2010-A1-Injection >.

Category-811: OWASP Top Ten 2010 Category A2 - Cross-Site Scripting (XSS)

Category ID : 811

Summary

Weaknesses in this category are related to the A2 category in the OWASP Top Ten 2010.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	809	Weaknesses in OWASP Top Ten (2010)	809	2563
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	809	163

References

[REF-762]OWASP. "Top 10 2010-A2-Cross-Site Scripting (XSS)". < http://www.owasp.org/index.php/Top_10_2010-A2-Cross-Site_Scripting_%28XSS%29 >.

Category-812: OWASP Top Ten 2010 Category A3 - Broken Authentication and Session Management

Category ID : 812

Summary

Weaknesses in this category are related to the A3 category in the OWASP Top Ten 2010.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	809	Weaknesses in OWASP Top Ten (2010)	809	2563
HasMember	C	287	Improper Authentication	809	692
HasMember	B	306	Missing Authentication for Critical Function	809	741
HasMember	B	307	Improper Restriction of Excessive Authentication Attempts	809	747
HasMember	B	798	Use of Hard-coded Credentials	809	1690

References

[REF-763]OWASP. "Top 10 2010-A3-Broken Authentication and Session Management". < http://www.owasp.org/index.php/Top_10_2010-A3-Broken_Authentication_and_Session_Management >.

Category-813: OWASP Top Ten 2010 Category A4 - Insecure Direct Object References

Category ID : 813

Summary

Weaknesses in this category are related to the A4 category in the OWASP Top Ten 2010.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	809	Weaknesses in OWASP Top Ten (2010)	809	2563
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	809	33

Nature	Type	ID	Name	V	Page
HasMember	C	99	Improper Control of Resource Identifiers ('Resource Injection')	809	243
HasMember	B	434	Unrestricted Upload of File with Dangerous Type	809	1048
HasMember	B	639	Authorization Bypass Through User-Controlled Key	809	1406
HasMember	B	829	Inclusion of Functionality from Untrusted Control Sphere	809	1741
HasMember	C	862	Missing Authorization	809	1780
HasMember	C	863	Incorrect Authorization	809	1787

References

[REF-764]OWASP. "Top 10 2010-A4-Insecure Direct Object References". < http://www.owasp.org/index.php/Top_10_2010-A4-Insecure_Direct_Object_References >.

Category-814: OWASP Top Ten 2010 Category A5 - Cross-Site Request Forgery(CSRF)

Category ID : 814

Summary

Weaknesses in this category are related to the A5 category in the OWASP Top Ten 2010.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	809	Weaknesses in OWASP Top Ten (2010)	809	2563
HasMember		352	Cross-Site Request Forgery (CSRF)	809	868

References

[REF-765]OWASP. "Top 10 2010-A5-Cross-Site Request Forgery (CSRF)". < http://www.owasp.org/index.php/Top_10_2010-A5-Cross-Site_Request_Forgery_%28CSRF%29 >.

Category-815: OWASP Top Ten 2010 Category A6 - Security Misconfiguration

Category ID : 815

Summary

Weaknesses in this category are related to the A6 category in the OWASP Top Ten 2010.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	809	Weaknesses in OWASP Top Ten (2010)	809	2563
HasMember	B	209	Generation of Error Message Containing Sensitive Information	809	533
HasMember	V	219	Storage of File with Sensitive Data Under Web Root	809	553
HasMember	B	250	Execution with Unnecessary Privileges	809	599
HasMember	B	538	Insertion of Sensitive Information into Externally-Accessible File or Directory	809	1248
HasMember	B	552	Files or Directories Accessible to External Parties	809	1265
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	809	1551

References

[REF-766]OWASP. "Top 10 2010-A6-Security Misconfiguration". < http://www.owasp.org/index.php/Top_10_2010-A6-Security_Misconfiguration >.

Category-816: OWASP Top Ten 2010 Category A7 - Insecure Cryptographic Storage

Category ID : 816

Summary

Weaknesses in this category are related to the A7 category in the OWASP Top Ten 2010.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	809	Weaknesses in OWASP Top Ten (2010)	809	2563
HasMember	C	311	Missing Encryption of Sensitive Data	809	757
HasMember	B	312	Cleartext Storage of Sensitive Information	809	764
HasMember	C	326	Inadequate Encryption Strength	809	796
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	809	799
HasMember	V	759	Use of a One-Way Hash without a Salt	809	1585

References

[REF-767]OWASP. "Top 10 2010-A7-Insecure Cryptographic Storage". < http://www.owasp.org/index.php/Top_10_2010-A7-Insecure_Cryptographic_Storage >.

Category-817: OWASP Top Ten 2010 Category A8 - Failure to Restrict URL Access

Category ID : 817

Summary

Weaknesses in this category are related to the A8 category in the OWASP Top Ten 2010.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	809	Weaknesses in OWASP Top Ten (2010)	809	2563
HasMember	C	285	Improper Authorization	809	684
HasMember	C	862	Missing Authorization	809	1780
HasMember	C	863	Incorrect Authorization	809	1787

References

[REF-768]OWASP. "Top 10 2010-A8-Failure to Restrict URL Access". < http://www.owasp.org/index.php/Top_10_2010-A8-Failure_to_Restrict_URL_Access >.

Category-818: OWASP Top Ten 2010 Category A9 - Insufficient Transport Layer Protection

Category ID : 818

Summary

Weaknesses in this category are related to the A9 category in the OWASP Top Ten 2010.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	809	Weaknesses in OWASP Top Ten (2010)	809	2563
MemberOf	C	1346	OWASP Top Ten 2021 Category A02:2021 - Cryptographic Failures	1344	2488
HasMember	C	311	Missing Encryption of Sensitive Data	809	757
HasMember	B	319	Cleartext Transmission of Sensitive Information	809	779

References

[REF-769]OWASP. "Top 10 2010-A9-Insufficient Transport Layer Protection". < http://www.owasp.org/index.php/Top_10_2010-A9-Insufficient_Transport_Layer_Protection >.

Category-819: OWASP Top Ten 2010 Category A10 - Unvalidated Redirects and Forwards

Category ID : 819

Summary

Weaknesses in this category are related to the A10 category in the OWASP Top Ten 2010.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	809	Weaknesses in OWASP Top Ten (2010)	809	2563
HasMember	B	601	URL Redirection to Untrusted Site ('Open Redirect')	809	1345

References

[REF-770]OWASP. "Top 10 2010-A10-Unvalidated Redirects and Forwards". < http://www.owasp.org/index.php/Top_10_2010-A10-Unvalidated_Redirects_and_Forwards >.

Category-840: Business Logic Errors

Category ID : 840

Summary

Weaknesses in this category identify some of the underlying problems that commonly allow attackers to manipulate the business logic of an application. Errors in business logic can be devastating to an entire application. They can be difficult to find automatically, since they typically involve legitimate use of the application's functionality. However, many business logic errors can exhibit patterns that are similar to well-understood implementation and design weaknesses.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
MemberOf	C	1348	OWASP Top Ten 2021 Category A04:2021 - Insecure Design	1344	2491
HasMember	B	283	Unverified Ownership	699	678
HasMember	B	639	Authorization Bypass Through User-Controlled Key	699	1406
HasMember	B	640	Weak Password Recovery Mechanism for Forgotten Password	699	1409

Nature	Type	ID	Name	V	Page
HasMember	B	708	Incorrect Ownership Assignment	699	1548
HasMember	B	770	Allocation of Resources Without Limits or Throttling	699	1613
HasMember	B	826	Premature Release of Resource During Expected Lifetime	699	1734
HasMember	B	837	Improper Enforcement of a Single, Unique Action	699	1762
HasMember	B	841	Improper Enforcement of Behavioral Workflow	699	1772

Notes

Terminology

The "Business Logic" term is generally used to describe issues that require domain-specific knowledge or "business rules" to determine if they are weaknesses or vulnerabilities, instead of legitimate behavior. Such issues might not be easily detectable via automatic code analysis, because the associated operations do not produce clear errors or undefined behavior at the code level. However, many such "business logic" issues can be understood as instances of other weaknesses such as input validation, access control, numeric computation, order of operations, etc.

Research Gap

The classification of business logic flaws has been under-studied, although exploitation of business flaws frequently happens in real-world systems, and many applied vulnerability researchers investigate them. The greatest focus is in web applications. There is debate within the community about whether these problems represent particularly new concepts, or if they are variations of well-known principles. Many business logic flaws appear to be oriented toward business processes, application flows, and sequences of behaviors, which are not as well-represented in CWE as weaknesses related to input validation, memory management, etc.

References

[REF-795]Jeremiah Grossman. "Business Logic Flaws and Yahoo Games". 2006 December 8. < https://blog.jeremiahgrossman.com/2006/12/business-logic-flaws.html >.2023-04-07.

[REF-796]Jeremiah Grossman. "Seven Business Logic Flaws That Put Your Website At Risk". 2007 October. < https://docplayer.net/10021793-Seven-business-logic-flaws-that-put-your-website-at-risk.html >.2023-04-07.

[REF-797]WhiteHat Security. "Business Logic Flaws". < https://web.archive.org/web/20080720171327/http://www.whitehatsec.com/home/solutions/BL_auction.html >.2023-04-07.

[REF-798]WASC. "Abuse of Functionality". < http://projects.webappsec.org/w/page/13246913/Abuse-of-Functionality >.

[REF-799]Rafal Los and Prajakta Jagdale. "Defying Logic: Theory, Design, and Implementation of Complex Systems for Testing Application Logic". 2011. < https://www.slideshare.net/RafalLos/defying-logic-business-logic-testing-with-automation >.2023-04-07.

[REF-667]Rafal Los. "Real-Life Example of a 'Business Logic Defect' (Screen Shots!)". 2011. < http://h30501.www3.hp.com/t5/Following-the-White-Rabbit-A/Real-Life-Example-of-a-Business-Logic-Defect-Screen-Shots/ba-p/22581 >.

[REF-801]Viktoria Felmetsger, Ludovico Cavedon, Christopher Kruegel and Giovanni Vigna. "Toward Automated Detection of Logic Vulnerabilities in Web Applications". USENIX Security Symposium 2010. 2010 August. < https://www.usenix.org/legacy/events/sec10/tech/full_papers/Felmetsger.pdf >.2023-04-07.

[REF-802]Faisal Nabi. "Designing a Framework Method for Secure Business Application Logic Integrity in e-Commerce Systems". International Journal of Network Security, Vol.12, No.1. 2011. < http://ijns.femto.com.tw/contents/ijns-v12-n1/ijns-2011-v12-n1-p29-41.pdf >.

[REF-1102]Chetan Conikee. "Case Files from 20 Years of Business Logic Flaws". 2020 February. < https://published-prd.lanyonevents.com/published/rsaus20/sessionsFiles/18217/2020_USA20_DSO-R02_01_Case%20Files%20from%2020%20Years%20of%20Business%20Logic%20Flaws.pdf >.

Category-845: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 2 - Input Validation and Data Sanitization (IDS)

Category ID : 845

Summary

Weaknesses in this category are related to rules in the Input Validation and Data Sanitization (IDS) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	844	151
HasMember	C	116	Improper Encoding or Escaping of Output	844	281
HasMember	B	134	Use of Externally-Controlled Format String	844	365
HasMember	V	144	Improper Neutralization of Line Delimiters	844	383
HasMember	V	150	Improper Neutralization of Escape, Meta, or Control Sequences	844	394
HasMember	V	180	Incorrect Behavior Order: Validate Before Canonicalize	844	451
HasMember	B	182	Collapse of Data into Unsafe Value	844	455
HasMember	B	289	Authentication Bypass by Alternate Name	844	703
HasMember	B	409	Improper Handling of Highly Compressed Data (Data Amplification)	844	996
HasMember	B	625	Permissive Regular Expression	844	1392
HasMember	V	647	Use of Non-Canonical URL Paths for Authorization Decisions	844	1426
HasMember	B	838	Inappropriate Encoding for Output Context	844	1764

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-846: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 3 - Declarations and Initialization (DCL)

Category ID : 846

Summary

Weaknesses in this category are related to rules in the Declarations and Initialization (DCL) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	C	665	Improper Initialization	844	1456

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-847: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 4 - Expressions (EXP)

Category ID : 847

Summary

Weaknesses in this category are related to rules in the Expressions (EXP) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	B	252	Unchecked Return Value	844	606
HasMember	V	479	Signal Handler Use of a Non-reentrant Function	844	1147
HasMember	V	595	Comparison of Object References Instead of Object Contents	844	1334
HasMember	V	597	Use of Wrong Operator in String Comparison	844	1337

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-848: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 5 - Numeric Types and Operations (NUM)

Category ID : 848

Summary

Weaknesses in this category are related to rules in the Numeric Types and Operations (NUM) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	B	197	Numeric Truncation Error	844	500
HasMember	B	369	Divide By Zero	844	913
HasMember	B	681	Incorrect Conversion between Numeric Types	844	1495

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-849: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 6 - Object Orientation (OBJ)

Category ID : 849

Summary

Weaknesses in this category are related to rules in the Object Orientation (OBJ) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	B	374	Passing Mutable Objects to an Untrusted Method	844	920
HasMember	B	375	Returning a Mutable Object to an Untrusted Caller	844	923
HasMember	V	486	Comparison of Classes by Name	844	1164
HasMember	V	491	Public cloneable() Method Without Final ('Object Hijack')	844	1174
HasMember	V	492	Use of Inner Class Containing Sensitive Data	844	1175
HasMember	V	493	Critical Public Variable Without Final Modifier	844	1182
HasMember	V	498	Cloneable Class Containing Sensitive Information	844	1196
HasMember	V	500	Public Static Field Not Marked Final	844	1200
HasMember	V	582	Array Declared Public, Final, and Static	844	1314
HasMember	B	766	Critical Data Element Declared Public	844	1607

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-850: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 7 - Methods (MET)

Category ID : 850

Summary

Weaknesses in this category are related to rules in the Methods (MET) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	B	487	Reliance on Package-level Scope	844	1167
HasMember	V	568	finalize() Method Without super.finalize()	844	1290

Nature	Type	ID	Name	V	Page
HasMember	C	573	Improper Following of Specification by Caller	844	1298
HasMember	V	581	Object Model Violation: Just One of Equals and Hashcode Defined	844	1312
HasMember	V	583	finalize() Method Declared Public	844	1315
HasMember	B	586	Explicit Call to Finalize()	844	1320
HasMember	V	589	Call to Non-ubiquitous API	844	1325
HasMember	B	617	Reachable Assertion	844	1378

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-851: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 8 - Exceptional Behavior (ERR)

Category ID : 851

Summary

Weaknesses in this category are related to rules in the Exceptional Behavior (ERR) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	B	209	Generation of Error Message Containing Sensitive Information	844	533
HasMember	V	230	Improper Handling of Missing Values	844	570
HasMember	V	232	Improper Handling of Undefined Values	844	573
HasMember	B	248	Uncaught Exception	844	596
HasMember	V	382	J2EE Bad Practices: Use of System.exit()	844	933
HasMember	B	390	Detection of Error Condition Without Action	844	943
HasMember	B	395	Use of NullPointerException Catch to Detect NULL Pointer Dereference	844	957
HasMember	B	397	Declaration of Throws for Generic Exception	844	961
HasMember	B	460	Improper Cleanup on Thrown Exception	844	1102
HasMember	B	497	Exposure of Sensitive System Information to an Unauthorized Control Sphere	844	1193
HasMember	B	584	Return Inside Finally Block	844	1317
HasMember	V	600	Uncaught Exception in Servlet	844	1343
HasMember	∞	690	Unchecked Return Value to NULL Pointer Dereference	844	1514
HasMember	P	703	Improper Check or Handling of Exceptional Conditions	844	1535
HasMember	C	705	Incorrect Control Flow Scoping	844	1542

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-852: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 9 - Visibility and Atomicity (VNA)

Category ID : 852

Summary

Weaknesses in this category are related to rules in the Visibility and Atomicity (VNA) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	C	362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	844	888
HasMember	B	366	Race Condition within a Thread	844	904
HasMember	B	413	Improper Resource Locking	844	1003
HasMember	B	567	Unsynchronized Access to Shared Data in a Multithreaded Context	844	1288
HasMember	C	662	Improper Synchronization	844	1448
HasMember	C	667	Improper Locking	844	1464

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-853: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 10 - Locking (LCK)

Category ID : 853

Summary

Weaknesses in this category are related to rules in the Locking (LCK) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	B	412	Unrestricted Externally Accessible Lock	844	1000
HasMember	B	413	Improper Resource Locking	844	1003
HasMember	B	609	Double-Checked Locking	844	1362
HasMember	C	667	Improper Locking	844	1464
HasMember	B	820	Missing Synchronization	844	1720
HasMember	B	833	Deadlock	844	1753

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-854: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 11 - Thread APIs (THI)

Category ID : 854

Summary

Weaknesses in this category are related to rules in the Thread APIs (THI) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	V	572	Call to Thread run() instead of start()	844	1296
HasMember	C	705	Incorrect Control Flow Scoping	844	1542

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-855: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 12 - Thread Pools (TPS)

Category ID : 855

Summary

Weaknesses in this category are related to rules in the Thread Pools (TPS) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	B	392	Missing Report of Error Condition	844	951
HasMember	C	405	Asymmetric Resource Consumption (Amplification)	844	986
HasMember	B	410	Insufficient Resource Pool	844	998

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-856: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 13 - Thread-Safety Miscellaneous (TSM)

Category ID : 856

Summary

Weaknesses in this category are related to rules in the Thread-Safety Miscellaneous (TSM) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-857: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 14 - Input Output (FIO)

Category ID : 857

Summary

Weaknesses in this category are related to rules in the Input Output (FIO) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	V	67	Improper Handling of Windows Device Names	844	126
HasMember	B	135	Incorrect Calculation of Multi-Byte String Length	844	370
HasMember	V	198	Use of Incorrect Byte Ordering	844	503
HasMember	B	276	Incorrect Default Permissions	844	665
HasMember	V	279	Incorrect Execution-Assigned Permissions	844	671
HasMember	B	359	Exposure of Private Personal Information to an Unauthorized Actor	844	882
HasMember	C	377	Insecure Temporary File	844	925
HasMember	C	404	Improper Resource Shutdown or Release	844	980
HasMember	C	405	Asymmetric Resource Consumption (Amplification)	844	986
HasMember	B	459	Incomplete Cleanup	844	1099
HasMember	B	532	Insertion of Sensitive Information into Log File	844	1241
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	844	1551
HasMember	B	770	Allocation of Resources Without Limits or Throttling	844	1613

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-858: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 15 - Serialization (SER)

Category ID : 858

Summary

Weaknesses in this category are related to rules in the Serialization (SER) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	B	250	Execution with Unnecessary Privileges	844	599
HasMember	B	319	Cleartext Transmission of Sensitive Information	844	779
HasMember	C	400	Uncontrolled Resource Consumption	844	964
HasMember	V	499	Serializable Class Containing Sensitive Data	844	1198
HasMember	B	502	Deserialization of Untrusted Data	844	1204
HasMember	V	589	Call to Non-ubiquitous API	844	1325
HasMember	B	770	Allocation of Resources Without Limits or Throttling	844	1613

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-859: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 16 - Platform Security (SEC)

Category ID : 859

Summary

Weaknesses in this category are related to rules in the Platform Security (SEC) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	V	111	Direct Use of Unsafe JNI	844	266
HasMember	B	266	Incorrect Privilege Assignment	844	638
HasMember	B	272	Least Privilege Violation	844	656
HasMember	C	300	Channel Accessible by Non-Endpoint	844	730
HasMember	B	302	Authentication Bypass by Assumed-Immutable Data	844	735
HasMember	B	319	Cleartext Transmission of Sensitive Information	844	779
HasMember	B	347	Improper Verification of Cryptographic Signature	844	857
HasMember	B	470	Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')	844	1118
HasMember	B	494	Download of Code Without Integrity Check	844	1185
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	844	1551
HasMember	B	807	Reliance on Untrusted Inputs in a Security Decision	844	1714

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-860: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 17 - Runtime Environment (ENV)

Category ID : 860

Summary

Weaknesses in this category are related to rules in the Runtime Environment (ENV) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	B	349	Acceptance of Extraneous Untrusted Data With Trusted Data	844	861
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	844	1551

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-861: The CERT Oracle Secure Coding Standard for Java (2011) Chapter 18 - Miscellaneous (MSC)

Category ID : 861

Summary

Weaknesses in this category are related to rules in the Miscellaneous (MSC) chapter of The CERT Oracle Secure Coding Standard for Java (2011).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	844	Weaknesses Addressed by The CERT Oracle Secure Coding Standard for Java (2011)	844	2564
HasMember	V	259	Use of Hard-coded Password	844	623
HasMember	C	311	Missing Encryption of Sensitive Data	844	757
HasMember	C	330	Use of Insufficiently Random Values	844	814
HasMember	V	332	Insufficient Entropy in PRNG	844	823
HasMember	V	333	Improper Handling of Insufficient Entropy in TRNG	844	825
HasMember	V	336	Same Seed in Pseudo-Random Number Generator (PRNG)	844	832
HasMember	V	337	Predictable Seed in Pseudo-Random Number Generator (PRNG)	844	834
HasMember	C	400	Uncontrolled Resource Consumption	844	964
HasMember	V	401	Missing Release of Memory after Effective Lifetime	844	973
HasMember	V	543	Use of Singleton Pattern Without Synchronization in a Multithreaded Context	844	1255
HasMember	B	770	Allocation of Resources Without Limits or Throttling	844	1613
HasMember	B	798	Use of Hard-coded Credentials	844	1690

References

[REF-813]Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland and David Svoboda. "The CERT Oracle Coding Standard for Java". 1st Edition. 2011 September 8. Addison-Wesley Professional.

Category-864: 2011 Top 25 - Insecure Interaction Between Components

Category ID : 864

Summary

Weaknesses in this category are listed in the "Insecure Interaction Between Components" section of the 2011 CWE/SANS Top 25 Most Dangerous Software Errors.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	900	Weaknesses in the 2011 CWE/SANS Top 25 Most Dangerous Software Errors	900	2572
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	900	151
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	900	163
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	900	201
HasMember	C	352	Cross-Site Request Forgery (CSRF)	900	868
HasMember	B	434	Unrestricted Upload of File with Dangerous Type	900	1048
HasMember	B	601	URL Redirection to Untrusted Site ('Open Redirect')	900	1345
HasMember	B	829	Inclusion of Functionality from Untrusted Control Sphere	900	1741

References

[REF-843]"2011 CWE/SANS Top 25 Most Dangerous Software Errors". 2011 June 7. < http://cwe.mitre.org/top25/archive/2011/2011_cwe_sans_top25.html >.

Category-865: 2011 Top 25 - Risky Resource Management

Category ID : 865

Summary

Weaknesses in this category are listed in the "Risky Resource Management" section of the 2011 CWE/SANS Top 25 Most Dangerous Software Errors.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	900	Weaknesses in the 2011 CWE/SANS Top 25 Most Dangerous Software Errors	900	2572
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	900	33
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	900	304
HasMember	B	131	Incorrect Calculation of Buffer Size	900	355
HasMember	B	134	Use of Externally-Controlled Format String	900	365
HasMember	B	190	Integer Overflow or Wraparound	900	472
HasMember	B	494	Download of Code Without Integrity Check	900	1185

Nature	Type	ID	Name	V	Page
HasMember	B	676	Use of Potentially Dangerous Function	900	1489

References

[REF-843]"2011 CWE/SANS Top 25 Most Dangerous Software Errors". 2011 June 7. < http://cwe.mitre.org/top25/archive/2011/2011_cwe_sans_top25.html >.

Category-866: 2011 Top 25 - Porous Defenses

Category ID : 866

Summary

Weaknesses in this category are listed in the "Porous Defenses" section of the 2011 CWE/SANS Top 25 Most Dangerous Software Errors.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	900	Weaknesses in the 2011 CWE/SANS Top 25 Most Dangerous Software Errors	900	2572
HasMember	B	250	Execution with Unnecessary Privileges	900	599
HasMember	B	306	Missing Authentication for Critical Function	900	741
HasMember	B	307	Improper Restriction of Excessive Authentication Attempts	900	747
HasMember	C	311	Missing Encryption of Sensitive Data	900	757
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	900	799
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	900	1551
HasMember	V	759	Use of a One-Way Hash without a Salt	900	1585
HasMember	B	798	Use of Hard-coded Credentials	900	1690
HasMember	B	807	Reliance on Untrusted Inputs in a Security Decision	900	1714
HasMember	C	862	Missing Authorization	900	1780
HasMember	C	863	Incorrect Authorization	900	1787

References

[REF-843]"2011 CWE/SANS Top 25 Most Dangerous Software Errors". 2011 June 7. < http://cwe.mitre.org/top25/archive/2011/2011_cwe_sans_top25.html >.

Category-867: 2011 Top 25 - Weaknesses On the Cusp

Category ID : 867

Summary

Weaknesses in this category are not part of the general Top 25, but they were part of the original nominee list from which the Top 25 was drawn.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	900	Weaknesses in the 2011 CWE/SANS Top 25 Most Dangerous Software Errors	900	2572
HasMember	V	129	Improper Validation of Array Index	900	341

Nature	Type	ID	Name	V	Page
HasMember	B	209	Generation of Error Message Containing Sensitive Information	900	533
HasMember	B	212	Improper Removal of Sensitive Information Before Storage or Transfer	900	544
HasMember	C	330	Use of Insufficiently Random Values	900	814
HasMember	C	362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	900	888
HasMember	V	456	Missing Initialization of a Variable	900	1089
HasMember	B	476	NULL Pointer Dereference	900	1132
HasMember	B	681	Incorrect Conversion between Numeric Types	900	1495
HasMember	C	754	Improper Check for Unusual or Exceptional Conditions	900	1568
HasMember	B	770	Allocation of Resources Without Limits or Throttling	900	1613
HasMember	B	772	Missing Release of Resource after Effective Lifetime	900	1624
HasMember	B	805	Buffer Access with Incorrect Length Value	900	1702
HasMember	B	822	Untrusted Pointer Dereference	900	1723
HasMember	B	825	Expired Pointer Dereference	900	1732
HasMember	B	838	Inappropriate Encoding for Output Context	900	1764
HasMember	B	841	Improper Enforcement of Behavioral Workflow	900	1772

References

[REF-843]"2011 CWE/SANS Top 25 Most Dangerous Software Errors". 2011 June 7. < http://cwe.mitre.org/top25/archive/2011/2011_cwe_sans_top25.html >.

Category-869: CERT C++ Secure Coding Section 01 - Preprocessor (PRE)

Category ID : 869

Summary

Weaknesses in this category are related to rules in the Preprocessor (PRE) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566

References

[REF-848]The Software Engineering Institute. "01. Preprocessor (PRE)". < https://www.securecoding.cert.org/confluence/display/cplusplus/01.+Preprocessor+%28PRE%29 >.

Category-870: CERT C++ Secure Coding Section 02 - Declarations and Initialization (DCL)

Category ID : 870

Summary

Weaknesses in this category are related to rules in the Declarations and Initialization (DCL) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566

References

[REF-849]CERT. "02. Declarations and Initialization (DCL)". < https://www.securecoding.cert.org/confluence/display/cplusplus/02.+Declarations+and+Initialization+%28DCL%29 >.

Category-871: CERT C++ Secure Coding Section 03 - Expressions (EXP)

Category ID : 871

Summary

Weaknesses in this category are related to rules in the Expressions (EXP) section of the CERT C ++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	B	476	NULL Pointer Dereference	868	1132
HasMember	B	480	Use of Incorrect Operator	868	1150
HasMember	V	768	Incorrect Short Circuit Evaluation	868	1612

References

[REF-850]CERT. "03. Expressions (EXP)". < https://www.securecoding.cert.org/confluence/display/cplusplus/03.+Expressions+%28EXP%29 >.

Category-872: CERT C++ Secure Coding Section 04 - Integers (INT)

Category ID : 872

Summary

Weaknesses in this category are related to rules in the Integers (INT) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	C	20	Improper Input Validation	868	20
HasMember	V	129	Improper Validation of Array Index	868	341
HasMember	B	190	Integer Overflow or Wraparound	868	472
HasMember	V	192	Integer Coercion Error	868	482

Nature	Type	ID	Name	V	Page
HasMember	B	197	Numeric Truncation Error	868	500
HasMember	B	369	Divide By Zero	868	913
HasMember	B	466	Return of Pointer Value Outside of Expected Range	868	1109
HasMember	V	587	Assignment of a Fixed Address to a Pointer	868	1322
HasMember	B	606	Unchecked Input for Loop Condition	868	1357
HasMember	B	676	Use of Potentially Dangerous Function	868	1489
HasMember	B	681	Incorrect Conversion between Numeric Types	868	1495
HasMember	P	682	Incorrect Calculation	868	1499

References

[REF-851]CERT. "04. Integers (INT)". < https://www.securecoding.cert.org/confluence/display/cplusplus/04.+Integers+%28INT%29 >.

Category-873: CERT C++ Secure Coding Section 05 - Floating Point Arithmetic (FLP)

Category ID : 873

Summary

Weaknesses in this category are related to rules in the Floating Point Arithmetic (FLP) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	B	369	Divide By Zero	868	913
HasMember	B	681	Incorrect Conversion between Numeric Types	868	1495
HasMember	P	682	Incorrect Calculation	868	1499
HasMember	V	686	Function Call With Incorrect Argument Type	868	1508

References

[REF-852]CERT. "05. Floating Point Arithmetic (FLP)". < https://www.securecoding.cert.org/confluence/display/cplusplus/05.+Floating+Point+Arithmetic+%28FLP%29 >.

Category-874: CERT C++ Secure Coding Section 06 - Arrays and the STL (ARR)

Category ID : 874

Summary

Weaknesses in this category are related to rules in the Arrays and the STL (ARR) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	868	293
HasMember	V	129	Improper Validation of Array Index	868	341
HasMember	V	467	Use of sizeof() on a Pointer Type	868	1110
HasMember	B	469	Use of Pointer Subtraction to Determine Size	868	1115
HasMember	C	665	Improper Initialization	868	1456
HasMember	B	805	Buffer Access with Incorrect Length Value	868	1702

References

[REF-853]CERT. "06. Arrays and the STL (ARR)". < https://www.securecoding.cert.org/confluence/display/cplusplus/06.+Arrays+and+the+STL+%28ARR%29 >.

Category-875: CERT C++ Secure Coding Section 07 - Characters and Strings (STR)

Category ID : 875

Summary

Weaknesses in this category are related to rules in the Characters and Strings (STR) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	868	151
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	868	194
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	868	293
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	868	304
HasMember	B	170	Improper Null Termination	868	428
HasMember	B	193	Off-by-one Error	868	486
HasMember	B	464	Addition of Data Structure Sentinel	868	1107
HasMember	V	686	Function Call With Incorrect Argument Type	868	1508
HasMember	C	704	Incorrect Type Conversion or Cast	868	1538

References

[REF-854]CERT. "07. Characters and Strings (STR)". < https://www.securecoding.cert.org/confluence/display/cplusplus/07.+Characters+and+Strings+%28STR%29 >.

Category-876: CERT C++ Secure Coding Section 08 - Memory Management (MEM)

Category ID : 876

Summary

Weaknesses in this category are related to rules in the Memory Management (MEM) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	C	20	Improper Input Validation	868	20
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	868	293
HasMember	B	128	Wrap-around Error	868	339
HasMember	B	131	Incorrect Calculation of Buffer Size	868	355
HasMember	B	190	Integer Overflow or Wraparound	868	472
HasMember	B	226	Sensitive Information in Resource Not Removed Before Reuse	868	562
HasMember	V	244	Improper Clearing of Heap Memory Before Release ('Heap Inspection')	868	591
HasMember	B	252	Unchecked Return Value	868	606
HasMember	B	391	Unchecked Error Condition	868	948
HasMember	C	404	Improper Resource Shutdown or Release	868	980
HasMember	V	415	Double Free	868	1008
HasMember	V	416	Use After Free	868	1012
HasMember	B	476	NULL Pointer Dereference	868	1132
HasMember	V	528	Exposure of Core Dump File to an Unauthorized Control Sphere	868	1237
HasMember	V	590	Free of Memory not on the Heap	868	1326
HasMember	V	591	Sensitive Data Storage in Improperly Locked Memory	868	1329
HasMember	C	665	Improper Initialization	868	1456
HasMember	V	687	Function Call With Incorrectly Specified Argument Value	868	1510
HasMember		690	Unchecked Return Value to NULL Pointer Dereference	868	1514
HasMember	P	703	Improper Check or Handling of Exceptional Conditions	868	1535
HasMember	C	754	Improper Check for Unusual or Exceptional Conditions	868	1568
HasMember	V	762	Mismatched Memory Management Routines	868	1596
HasMember	B	770	Allocation of Resources Without Limits or Throttling	868	1613
HasMember	B	822	Untrusted Pointer Dereference	868	1723

References

[REF-855]CERT. "08. Memory Management (MEM)". < https://www.securecoding.cert.org/confluence/display/cplusplus/08.+Memory+Management+%28MEM%29 >.

Category-877: CERT C++ Secure Coding Section 09 - Input Output (FIO)

Category ID : 877

Summary

Weaknesses in this category are related to rules in the Input Output (FIO) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	868	33
HasMember	V	37	Path Traversal: '/absolute/pathname/here'	868	79
HasMember	V	38	Path Traversal: '\absolute\pathname\here'	868	80
HasMember	V	39	Path Traversal: 'C:dirname'	868	82
HasMember	B	41	Improper Resolution of Path Equivalence	868	86
HasMember	B	59	Improper Link Resolution Before File Access ('Link Following')	868	111
HasMember	V	62	UNIX Hard Link	868	119
HasMember	V	64	Windows Shortcut Following (.LNK)	868	121
HasMember	V	65	Windows Hard Link	868	123
HasMember	V	67	Improper Handling of Windows Device Names	868	126
HasMember	B	73	External Control of File Name or Path	868	132
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	868	293
HasMember	B	134	Use of Externally-Controlled Format String	868	365
HasMember	B	241	Improper Handling of Unexpected Data Type	868	584
HasMember	B	276	Incorrect Default Permissions	868	665
HasMember	V	279	Incorrect Execution-Assigned Permissions	868	671
HasMember	C	362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	868	888
HasMember	B	367	Time-of-check Time-of-use (TOCTOU) Race Condition	868	906
HasMember	B	379	Creation of Temporary File in Directory with Insecure Permissions	868	930
HasMember	B	391	Unchecked Error Condition	868	948
HasMember	B	403	Exposure of File Descriptor to Unintended Control Sphere ('File Descriptor Leak')	868	978
HasMember	C	404	Improper Resource Shutdown or Release	868	980
HasMember	B	552	Files or Directories Accessible to External Parties	868	1265
HasMember	C	675	Multiple Operations on Resource in Single-Operation Context	868	1487
HasMember	B	676	Use of Potentially Dangerous Function	868	1489
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	868	1551
HasMember	B	770	Allocation of Resources Without Limits or Throttling	868	1613

References

[REF-856]CERT. "09. Input Output (FIO)". < https://www.securecoding.cert.org/confluence/display/cplusplus/09.+Input+Output+%28FIO%29 >.

Category-878: CERT C++ Secure Coding Section 10 - Environment (ENV)

Category ID : 878

Summary

Weaknesses in this category are related to rules in the Environment (ENV) section of the CERT C ++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	868	151
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	868	194
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	868	293
HasMember	B	426	Untrusted Search Path	868	1028
HasMember	V	462	Duplicate Key in Associative List (Alist)	868	1104
HasMember	C	705	Incorrect Control Flow Scoping	868	1542
HasMember	B	807	Reliance on Untrusted Inputs in a Security Decision	868	1714

References

[REF-857]CERT. "10. Environment (ENV)". < https://www.securecoding.cert.org/confluence/display/cplusplus/10.+Environment+%28ENV%29 >.

Category-879: CERT C++ Secure Coding Section 11 - Signals (SIG)

Category ID : 879

Summary

Weaknesses in this category are related to rules in the Signals (SIG) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	V	479	Signal Handler Use of a Non-reentrant Function	868	1147
HasMember	C	662	Improper Synchronization	868	1448

References

[REF-858]CERT. "11. Signals (SIG)". < https://www.securecoding.cert.org/confluence/display/cplusplus/11.+Signals+%28SIG%29 >.

Category-880: CERT C++ Secure Coding Section 12 - Exceptions and Error Handling (ERR)

Category ID : 880

Summary

Weaknesses in this category are related to rules in the Exceptions and Error Handling (ERR) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	B	209	Generation of Error Message Containing Sensitive Information	868	533
HasMember	B	390	Detection of Error Condition Without Action	868	943
HasMember	B	391	Unchecked Error Condition	868	948
HasMember	B	460	Improper Cleanup on Thrown Exception	868	1102
HasMember	B	497	Exposure of Sensitive System Information to an Unauthorized Control Sphere	868	1193
HasMember	B	544	Missing Standardized Error Handling Mechanism	868	1256
HasMember	P	703	Improper Check or Handling of Exceptional Conditions	868	1535
HasMember	C	705	Incorrect Control Flow Scoping	868	1542
HasMember	C	754	Improper Check for Unusual or Exceptional Conditions	868	1568
HasMember	C	755	Improper Handling of Exceptional Conditions	868	1576

References

[REF-861]CERT. "12. Exceptions and Error Handling (ERR)". < https://www.securecoding.cert.org/confluence/display/cplusplus/12.+Exceptions+and+Error+Handling+%28ERR%29 >.

Category-881: CERT C++ Secure Coding Section 13 - Object Oriented Programming (OOP)

Category ID : 881

Summary

Weaknesses in this category are related to rules in the Object Oriented Programming (OOP) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566

References

[REF-862]CERT. "13. Object Oriented Programming (OOP)". < https://www.securecoding.cert.org/confluence/display/cplusplus/13.+Object+Oriented+Programming+%28OOP%29 >.

Category-882: CERT C++ Secure Coding Section 14 - Concurrency (CON)

Category ID : 882

Summary

Weaknesses in this category are related to rules in the Concurrency (CON) section of the CERT C ++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	C	362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	868	888
HasMember	B	366	Race Condition within a Thread	868	904
HasMember	C	404	Improper Resource Shutdown or Release	868	980
HasMember	B	488	Exposure of Data Element to Wrong Session	868	1169
HasMember	B	772	Missing Release of Resource after Effective Lifetime	868	1624

References

[REF-863]CERT. "14. Concurrency (CON)". < https://www.securecoding.cert.org/confluence/display/cplusplus/14.+Concurrency+%28CON%29 >.

Category-883: CERT C++ Secure Coding Section 49 - Miscellaneous (MSC)

Category ID : 883

Summary

Weaknesses in this category are related to rules in the Miscellaneous (MSC) section of the CERT C++ Secure Coding Standard. Since not all rules map to specific weaknesses, this category may be incomplete.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	868	Weaknesses Addressed by the SEI CERT C++ Coding Standard (2016 Version)	868	2566
HasMember	V	14	Compiler Removal of Code to Clear Buffers	868	14
HasMember	C	20	Improper Input Validation	868	20
HasMember	C	116	Improper Encoding or Escaping of Output	868	281
HasMember	V	176	Improper Handling of Unicode Encoding	868	440
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	868	799
HasMember	C	330	Use of Insufficiently Random Values	868	814
HasMember	B	480	Use of Incorrect Operator	868	1150
HasMember	V	482	Comparing instead of Assigning	868	1157
HasMember	B	561	Dead Code	868	1275
HasMember	B	563	Assignment to Variable without Use	868	1280
HasMember	B	570	Expression is Always False	868	1292
HasMember	B	571	Expression is Always True	868	1295
HasMember	P	697	Incorrect Comparison	868	1530
HasMember	C	704	Incorrect Type Conversion or Cast	868	1538

References

[REF-864]CERT. "49. Miscellaneous (MSC)". < https://www.securecoding.cert.org/confluence/display/cplusplus/49.+Miscellaneous+%28MSC%29 >.

Category-885: SFP Primary Cluster: Risky Values

Category ID : 885

Summary

This category identifies Software Fault Patterns (SFPs) within the Risky Values cluster (SFP1).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	998	SFP Secondary Cluster: Glitch in Computation	888	2419

Category-886: SFP Primary Cluster: Unused entities

Category ID : 886

Summary

This category identifies Software Fault Patterns (SFPs) within the Unused entities cluster (SFP2).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	V	482	Comparing instead of Assigning	888	1157
HasMember	B	561	Dead Code	888	1275
HasMember	B	563	Assignment to Variable without Use	888	1280

Category-887: SFP Primary Cluster: API

Category ID : 887

Summary

This category identifies Software Fault Patterns (SFPs) within the API cluster (SFP3).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	1001	SFP Secondary Cluster: Use of an Improper API	888	2420

Category-889: SFP Primary Cluster: Exception Management

Category ID : 889

Summary

This category identifies Software Fault Patterns (SFPs) within the Exception Management cluster (SFP4, SFP5, SFP6).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571

Nature	Type	ID	Name	V	Page
HasMember	C	960	SFP Secondary Cluster: Ambiguous Exception Type	888	2399
HasMember	C	961	SFP Secondary Cluster: Incorrect Exception Behavior	888	2399
HasMember	C	962	SFP Secondary Cluster: Unchecked Status Condition	888	2400

Category-890: SFP Primary Cluster: Memory Access

Category ID : 890

Summary

This category identifies Software Fault Patterns (SFPs) within the Memory Access cluster (SFP7, SFP8).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	970	SFP Secondary Cluster: Faulty Buffer Access	888	2405
HasMember	C	971	SFP Secondary Cluster: Faulty Pointer Use	888	2405
HasMember	C	972	SFP Secondary Cluster: Faulty String Expansion	888	2405
HasMember	C	973	SFP Secondary Cluster: Improper NULL Termination	888	2406
HasMember	C	974	SFP Secondary Cluster: Incorrect Buffer Length Computation	888	2406

Category-891: SFP Primary Cluster: Memory Management

Category ID : 891

Summary

This category identifies Software Fault Patterns (SFPs) within the Memory Management cluster (SFP38).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	969	SFP Secondary Cluster: Faulty Memory Release	888	2404

Category-892: SFP Primary Cluster: Resource Management

Category ID : 892

Summary

This category identifies Software Fault Patterns (SFPs) within the Resource Management cluster (SFP37).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	982	SFP Secondary Cluster: Failure to Release Resource	888	2410

Nature	Type	ID	Name	V	Page
HasMember	C	983	SFP Secondary Cluster: Faulty Resource Use	888	2410
HasMember	C	984	SFP Secondary Cluster: Life Cycle	888	2411
HasMember	C	985	SFP Secondary Cluster: Unrestricted Consumption	888	2411

Category-893: SFP Primary Cluster: Path Resolution

Category ID : 893

Summary

This category identifies Software Fault Patterns (SFPs) within the Path Resolution cluster (SFP16, SFP17, SFP18).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	979	SFP Secondary Cluster: Failed Chroot Jail	888	2408
HasMember	C	980	SFP Secondary Cluster: Link in Resource Name Resolution	888	2409
HasMember	C	981	SFP Secondary Cluster: Path Traversal	888	2409

Category-894: SFP Primary Cluster: Synchronization

Category ID : 894

Summary

This category identifies Software Fault Patterns (SFPs) within the Synchronization cluster (SFP19, SFP20, SFP21, SFP22).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	986	SFP Secondary Cluster: Missing Lock	888	2411
HasMember	C	987	SFP Secondary Cluster: Multiple Locks/Unlocks	888	2412
HasMember	C	988	SFP Secondary Cluster: Race Condition Window	888	2412
HasMember	C	989	SFP Secondary Cluster: Unrestricted Lock	888	2413

Category-895: SFP Primary Cluster: Information Leak

Category ID : 895

Summary

This category identifies Software Fault Patterns (SFPs) within the Information Leak cluster (SFP23).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571

Nature	Type	ID	Name	V	Page
HasMember	C	963	SFP Secondary Cluster: Exposed Data	888	2400
HasMember	C	964	SFP Secondary Cluster: Exposure Temporary File	888	2402
HasMember	C	965	SFP Secondary Cluster: Insecure Session Management	888	2403
HasMember	C	966	SFP Secondary Cluster: Other Exposures	888	2403
HasMember	C	967	SFP Secondary Cluster: State Disclosure	888	2403

Category-896: SFP Primary Cluster: Tainted Input

Category ID : 896

Summary

This category identifies Software Fault Patterns (SFPs) within the Tainted Input cluster (SFP24, SFP25, SFP26, SFP27).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	990	SFP Secondary Cluster: Tainted Input to Command	888	2413
HasMember	C	991	SFP Secondary Cluster: Tainted Input to Environment	888	2416
HasMember	C	992	SFP Secondary Cluster: Faulty Input Transformation	888	2416
HasMember	C	993	SFP Secondary Cluster: Incorrect Input Handling	888	2417
HasMember	C	994	SFP Secondary Cluster: Tainted Input to Variable	888	2417

Category-897: SFP Primary Cluster: Entry Points

Category ID : 897

Summary

This category identifies Software Fault Patterns (SFPs) within the Entry Points cluster (SFP28).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	1002	SFP Secondary Cluster: Unexpected Entry Points	888	2421

Category-898: SFP Primary Cluster: Authentication

Category ID : 898

Summary

This category identifies Software Fault Patterns (SFPs) within the Authentication cluster (SFP29, SFP30, SFP31, SFP32, SFP33, SFP34).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	947	SFP Secondary Cluster: Authentication Bypass	888	2394

Nature	Type	ID	Name	V	Page
HasMember	C	948	SFP Secondary Cluster: Digital Certificate	888	2395
HasMember	C	949	SFP Secondary Cluster: Faulty Endpoint Authentication	888	2395
HasMember	C	950	SFP Secondary Cluster: Hardcoded Sensitive Data	888	2396
HasMember	C	951	SFP Secondary Cluster: Insecure Authentication Policy	888	2396
HasMember	C	952	SFP Secondary Cluster: Missing Authentication	888	2396
HasMember	C	953	SFP Secondary Cluster: Missing Endpoint Authentication	888	2397
HasMember	C	954	SFP Secondary Cluster: Multiple Binds to the Same Port	888	2397
HasMember	C	955	SFP Secondary Cluster: Unrestricted Authentication	888	2397

Category-899: SFP Primary Cluster: Access Control

Category ID : 899

Summary

This category identifies Software Fault Patterns (SFPs) within the Access Control cluster (SFP35).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	944	SFP Secondary Cluster: Access Management	888	2393
HasMember	C	945	SFP Secondary Cluster: Insecure Resource Access	888	2394
HasMember	C	946	SFP Secondary Cluster: Insecure Resource Permissions	888	2394

Category-901: SFP Primary Cluster: Privilege

Category ID : 901

Summary

This category identifies Software Fault Patterns (SFPs) within the Privilege cluster (SFP36).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	V	9	J2EE Misconfiguration: Weak Access Permissions for EJB Methods	888	8
HasMember	B	250	Execution with Unnecessary Privileges	888	599
HasMember	B	266	Incorrect Privilege Assignment	888	638
HasMember	B	267	Privilege Defined With Unsafe Actions	888	641
HasMember	B	268	Privilege Chaining	888	644
HasMember	C	269	Improper Privilege Management	888	646
HasMember	B	270	Privilege Context Switching Error	888	651
HasMember	C	271	Privilege Dropping / Lowering Errors	888	653
HasMember	B	272	Least Privilege Violation	888	656
HasMember	B	274	Improper Handling of Insufficient Privileges	888	663
HasMember	V	520	.NET Misconfiguration: Use of Impersonation	888	1222

Nature	Type	ID	Name	V	Page
HasMember	C	653	Improper Isolation or Compartmentalization	888	1437

Category-902: SFP Primary Cluster: Channel

Category ID : 902

Summary

This category identifies Software Fault Patterns (SFPs) within the Channel cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	956	SFP Secondary Cluster: Channel Attack	888	2397
HasMember	C	957	SFP Secondary Cluster: Protocol Error	888	2398

Category-903: SFP Primary Cluster: Cryptography

Category ID : 903

Summary

This category identifies Software Fault Patterns (SFPs) within the Cryptography cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	958	SFP Secondary Cluster: Broken Cryptography	888	2398
HasMember	C	959	SFP Secondary Cluster: Weak Cryptography	888	2398

Category-904: SFP Primary Cluster: Malware

Category ID : 904

Summary

This category identifies Software Fault Patterns (SFPs) within the Malware cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	V	69	Improper Handling of Windows ::DATA Alternate Data Stream	888	129
HasMember	C	506	Embedded Malicious Code	888	1210
HasMember	B	507	Trojan Horse	888	1212
HasMember	B	508	Non-Replicating Malicious Code	888	1213
HasMember	B	509	Replicating Malicious Code (Virus or Worm)	888	1214
HasMember	B	510	Trapdoor	888	1215
HasMember	B	511	Logic/Time Bomb	888	1216
HasMember	B	512	Spyware	888	1218

Nature	Type	ID	Name	V	Page
HasMember	C	968	SFP Secondary Cluster: Covert Channel	888	2404

Category-905: SFP Primary Cluster: Predictability

Category ID : 905

Summary

This category identifies Software Fault Patterns (SFPs) within the Predictability cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	330	Use of Insufficiently Random Values	888	814
HasMember	B	331	Insufficient Entropy	888	821
HasMember	V	332	Insufficient Entropy in PRNG	888	823
HasMember	V	333	Improper Handling of Insufficient Entropy in TRNG	888	825
HasMember	B	334	Small Space of Random Values	888	827
HasMember	B	335	Incorrect Usage of Seeds in Pseudo-Random Number Generator (PRNG)	888	829
HasMember	V	336	Same Seed in Pseudo-Random Number Generator (PRNG)	888	832
HasMember	V	337	Predictable Seed in Pseudo-Random Number Generator (PRNG)	888	834
HasMember	B	338	Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)	888	837
HasMember	V	339	Small Seed Space in PRNG	888	840
HasMember	C	340	Generation of Predictable Numbers or Identifiers	888	842
HasMember	B	341	Predictable from Observable State	888	843
HasMember	B	342	Predictable Exact Value from Previous Values	888	845
HasMember	B	343	Predictable Value Range from Previous Values	888	847
HasMember	B	344	Use of Invariant Value in Dynamically Changing Context	888	849

Category-906: SFP Primary Cluster: UI

Category ID : 906

Summary

This category identifies Software Fault Patterns (SFPs) within the UI cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	995	SFP Secondary Cluster: Feature	888	2418
HasMember	C	996	SFP Secondary Cluster: Security	888	2418
HasMember	C	997	SFP Secondary Cluster: Information Loss	888	2418

Category-907: SFP Primary Cluster: Other

Category ID : 907

Summary

This category identifies Software Fault Patterns (SFPs) within the Other cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	C	975	SFP Secondary Cluster: Architecture	888	2406
HasMember	C	976	SFP Secondary Cluster: Compiler	888	2407
HasMember	C	977	SFP Secondary Cluster: Design	888	2407
HasMember	C	978	SFP Secondary Cluster: Implementation	888	2408

Category-929: OWASP Top Ten 2013 Category A1 - Injection

Category ID : 929

Summary

Weaknesses in this category are related to the A1 category in the OWASP Top Ten 2013.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	928	Weaknesses in OWASP Top Ten (2013)	928	2574
HasMember	C	74	Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')	928	137
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	928	145
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	928	151
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	928	194
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	928	201
HasMember	B	90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	928	212
HasMember	B	91	XML Injection (aka Blind XPath Injection)	928	215
HasMember	B	643	Improper Neutralization of Data within XPath Expressions ('XPath Injection')	928	1419
HasMember	B	652	Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')	928	1435

References

[REF-927]OWASP. "Top 10 2013-A1-Injection". < https://www.owasp.org/index.php/Top_10_2013-A1-Injection >.

Category-930: OWASP Top Ten 2013 Category A2 - Broken Authentication and Session Management

Category ID : 930

Summary

Weaknesses in this category are related to the A2 category in the OWASP Top Ten 2013.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	928	Weaknesses in OWASP Top Ten (2013)	928	2574
HasMember	B	256	Plaintext Storage of a Password	928	615
HasMember	C	287	Improper Authentication	928	692
HasMember	C	311	Missing Encryption of Sensitive Data	928	757
HasMember		384	Session Fixation	928	936
HasMember	C	522	Insufficiently Protected Credentials	928	1225
HasMember	B	523	Unprotected Transport of Credentials	928	1230
HasMember	B	613	Insufficient Session Expiration	928	1371
HasMember	B	620	Unverified Password Change	928	1383
HasMember	B	640	Weak Password Recovery Mechanism for Forgotten Password	928	1409

References

[REF-929]OWASP. "Top 10 2013-A2-Broken Authentication and Session Management". < https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management >.

Category-931: OWASP Top Ten 2013 Category A3 - Cross-Site Scripting (XSS)

Category ID : 931

Summary

Weaknesses in this category are related to the A3 category in the OWASP Top Ten 2013.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	928	Weaknesses in OWASP Top Ten (2013)	928	2574
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	928	163

References

[REF-930]OWASP. "Top 10 2013-A3-Cross-Site Scripting (XSS)". < https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_%28XSS%29 >.

Category-932: OWASP Top Ten 2013 Category A4 - Insecure Direct Object References

Category ID : 932

Summary

Weaknesses in this category are related to the A4 category in the OWASP Top Ten 2013.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	928	Weaknesses in OWASP Top Ten (2013)	928	2574

Nature	Type	ID	Name	V	Page
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	928	33
HasMember	C	99	Improper Control of Resource Identifiers ('Resource Injection')	928	243
HasMember	B	639	Authorization Bypass Through User-Controlled Key	928	1406
HasMember	C	706	Use of Incorrectly-Resolved Name or Reference	928	1544

References

[REF-931]OWASP. "Top 10 2013-A4-Insecure Direct Object References". < https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References >.

Category-933: OWASP Top Ten 2013 Category A5 - Security Misconfiguration

Category ID : 933

Summary

Weaknesses in this category are related to the A5 category in the OWASP Top Ten 2013.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	928	Weaknesses in OWASP Top Ten (2013)	928	2574
HasMember	C	2	7PK - Environment	928	2308
HasMember	C	16	Configuration	928	2309
HasMember	B	209	Generation of Error Message Containing Sensitive Information	928	533
HasMember	B	215	Insertion of Sensitive Information Into Debugging Code	928	551
HasMember	V	548	Exposure of Information Through Directory Listing	928	1261

References

[REF-932]OWASP. "Top 10 2013-A5-Security Misconfiguration". < https://www.owasp.org/index.php/Top_10_2013-A5-Security_Misconfiguration >.

Category-934: OWASP Top Ten 2013 Category A6 - Sensitive Data Exposure

Category ID : 934

Summary

Weaknesses in this category are related to the A6 category in the OWASP Top Ten 2013.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	928	Weaknesses in OWASP Top Ten (2013)	928	2574
HasMember	C	311	Missing Encryption of Sensitive Data	928	757
HasMember	B	312	Cleartext Storage of Sensitive Information	928	764
HasMember	B	319	Cleartext Transmission of Sensitive Information	928	779
HasMember	C	320	Key Management Errors	928	2319
HasMember	B	325	Missing Cryptographic Step	928	794
HasMember	C	326	Inadequate Encryption Strength	928	796
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	928	799

Nature	Type	ID	Name	V	Page
HasMember	B	328	Use of Weak Hash	928	806

References

[REF-933]OWASP. "Top 10 2013-A6-Sensitive Data Exposure". < https://www.owasp.org/index.php/Top_10_2013-A6-Sensitive_Data_Exposure >.

Category-935: OWASP Top Ten 2013 Category A7 - Missing Function Level Access Control

Category ID : 935

Summary

Weaknesses in this category are related to the A7 category in the OWASP Top Ten 2013.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	928	Weaknesses in OWASP Top Ten (2013)	928	2574
HasMember	C	285	Improper Authorization	928	684

References

[REF-934]OWASP. "Top 10 2013-A7-Missing Function Level Access Control". < https://www.owasp.org/index.php/Top_10_2013-A7-Missing_Function_Level_Access_Control >.

Category-936: OWASP Top Ten 2013 Category A8 - Cross-Site Request Forgery (CSRF)

Category ID : 936

Summary

Weaknesses in this category are related to the A8 category in the OWASP Top Ten 2013.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	928	Weaknesses in OWASP Top Ten (2013)	928	2574
HasMember	Composite	352	Cross-Site Request Forgery (CSRF)	928	868

References

[REF-935]OWASP. "Top 10 2013-A8-Cross-Site Request Forgery (CSRF)". < https://www.owasp.org/index.php/Top_10_2013-A8-Cross-Site_Request_Forgery_%28CSRF%29 >.

Category-937: OWASP Top Ten 2013 Category A9 - Using Components with Known Vulnerabilities

Category ID : 937

Summary

Weaknesses in this category are related to the A9 category in the OWASP Top Ten 2013.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	928	Weaknesses in OWASP Top Ten (2013)	928	2574
MemberOf	C	1352	OWASP Top Ten 2021 Category A06:2021 - Vulnerable and Outdated Components	1344	2494

Notes

Relationship

This is an unusual category. CWE does not cover the limitations of human processes and procedures that cannot be described in terms of a specific technical weakness as resident in the code, architecture, or configuration of the software. Since "known vulnerabilities" can arise from any kind of weakness, it is not possible to map this OWASP category to other CWE entries, since it would effectively require mapping this category to ALL weaknesses.

References

[REF-936]OWASP. "Top 10 2013-A9-Using Components with Known Vulnerabilities". < https://www.owasp.org/index.php/Top_10_2013-A9-Using_Components_with_Known_Vulnerabilities >.

Category-938: OWASP Top Ten 2013 Category A10 - Unvalidated Redirects and Forwards

Category ID : 938

Summary

Weaknesses in this category are related to the A10 category in the OWASP Top Ten 2013.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	928	Weaknesses in OWASP Top Ten (2013)	928	2574
HasMember	B	601	URL Redirection to Untrusted Site ('Open Redirect')	928	1345

References

[REF-937]OWASP. "Top 10 2013-A10-Unvalidated Redirects and Forwards". < https://www.owasp.org/index.php/Top_10_2013-A10-Unvalidated_Redirects_and_Forwards >.

Category-944: SFP Secondary Cluster: Access Management

Category ID : 944

Summary

This category identifies Software Fault Patterns (SFPs) within the Access Management cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	899	SFP Primary Cluster: Access Control	888	2386
HasMember	C	282	Improper Ownership Management	888	676
HasMember	B	283	Unverified Ownership	888	678
HasMember	P	284	Improper Access Control	888	680
HasMember	C	286	Incorrect User Management	888	691
HasMember	B	708	Incorrect Ownership Assignment	888	1548

Category-945: SFP Secondary Cluster: Insecure Resource Access

Category ID : 945

Summary

This category identifies Software Fault Patterns (SFPs) within the Insecure Resource Access cluster (SFP35).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	899	SFP Primary Cluster: Access Control	888	2386
HasMember	C	285	Improper Authorization	888	684
HasMember	C	424	Improper Protection of Alternate Path	888	1023
HasMember	B	639	Authorization Bypass Through User-Controlled Key	888	1406
HasMember	V	650	Trusting HTTP Permission Methods on the Server Side	888	1432

Category-946: SFP Secondary Cluster: Insecure Resource Permissions

Category ID : 946

Summary

This category identifies Software Fault Patterns (SFPs) within the Insecure Resource Permissions cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	899	SFP Primary Cluster: Access Control	888	2386
HasMember	B	276	Incorrect Default Permissions	888	665
HasMember	V	277	Insecure Inherited Permissions	888	668
HasMember	V	278	Insecure Preserved Inherited Permissions	888	669
HasMember	V	279	Incorrect Execution-Assigned Permissions	888	671
HasMember	B	281	Improper Preservation of Permissions	888	674
HasMember	V	560	Use of umask() with chmod-style Argument	888	1274
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	888	1551

Category-947: SFP Secondary Cluster: Authentication Bypass

Category ID : 947

Summary

This category identifies Software Fault Patterns (SFPs) within the Authentication Bypass cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	898	SFP Primary Cluster: Authentication	888	2385
HasMember	C	287	Improper Authentication	888	692
HasMember	B	288	Authentication Bypass Using an Alternate Path or Channel	888	700
HasMember	B	289	Authentication Bypass by Alternate Name	888	703
HasMember	B	303	Incorrect Implementation of Authentication Algorithm	888	737

Nature	Type	ID	Name	V	Page
HasMember	B	304	Missing Critical Step in Authentication	888	738
HasMember	B	305	Authentication Bypass by Primary Weakness	888	740
HasMember	B	308	Use of Single-factor Authentication	888	752
HasMember	B	309	Use of Password System for Primary Authentication	888	754
HasMember	B	603	Use of Client-Side Authentication	888	1354

Category-948: SFP Secondary Cluster: Digital Certificate

Category ID : 948

Summary

This category identifies Software Fault Patterns (SFPs) within the Digital Certificate cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	898	SFP Primary Cluster: Authentication	888	2385
HasMember	B	296	Improper Following of a Certificate's Chain of Trust	888	719
HasMember	V	297	Improper Validation of Certificate with Host Mismatch	888	722
HasMember	V	298	Improper Validation of Certificate Expiration	888	726
HasMember	B	299	Improper Check for Certificate Revocation	888	727
HasMember	V	593	Authentication Bypass: OpenSSL CTX Object Modified after SSL Objects are Created	888	1331
HasMember	V	599	Missing Validation of OpenSSL Certificate	888	1341

Category-949: SFP Secondary Cluster: Faulty Endpoint Authentication

Category ID : 949

Summary

This category identifies Software Fault Patterns (SFPs) within the Faulty Endpoint Authentication cluster (SFP29).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	898	SFP Primary Cluster: Authentication	888	2385
HasMember	V	293	Using Referer Field for Authentication	888	710
HasMember	B	302	Authentication Bypass by Assumed-Immutable Data	888	735
HasMember	C	345	Insufficient Verification of Data Authenticity	888	851
HasMember	C	346	Origin Validation Error	888	853
HasMember	V	350	Reliance on Reverse DNS Resolution for a Security-Critical Action	888	863
HasMember	B	360	Trust of System Event Data	888	887
HasMember	B	551	Incorrect Behavior Order: Authorization Before Parsing and Canonicalization	888	1264
HasMember	B	565	Reliance on Cookies without Validation and Integrity Checking	888	1283
HasMember	V	647	Use of Non-Canonical URL Paths for Authorization Decisions	888	1426

Category-950: SFP Secondary Cluster: Hardcoded Sensitive Data

Category ID : 950

Summary

This category identifies Software Fault Patterns (SFPs) within the Hardcoded Sensitive Data cluster (SFP33).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	898	SFP Primary Cluster: Authentication	888	2385
HasMember	V	258	Empty Password in Configuration File	888	621
HasMember	V	259	Use of Hard-coded Password	888	623
HasMember	V	321	Use of Hard-coded Cryptographic Key	888	785
HasMember	B	547	Use of Hard-coded, Security-relevant Constants	888	1259

Category-951: SFP Secondary Cluster: Insecure Authentication Policy

Category ID : 951

Summary

This category identifies Software Fault Patterns (SFPs) within the Insecure Authentication Policy cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	898	SFP Primary Cluster: Authentication	888	2385
HasMember	B	262	Not Using Password Aging	888	633
HasMember	B	263	Password Aging with Long Expiration	888	636
HasMember	B	521	Weak Password Requirements	888	1223
HasMember	V	556	ASP.NET Misconfiguration: Use of Identity Impersonation	888	1271
HasMember	B	613	Insufficient Session Expiration	888	1371
HasMember	B	645	Overly Restrictive Account Lockout Mechanism	888	1423

Category-952: SFP Secondary Cluster: Missing Authentication

Category ID : 952

Summary

This category identifies Software Fault Patterns (SFPs) within the Missing Authentication cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	898	SFP Primary Cluster: Authentication	888	2385
HasMember	B	306	Missing Authentication for Critical Function	888	741
HasMember	B	620	Unverified Password Change	888	1383

Category-953: SFP Secondary Cluster: Missing Endpoint Authentication

Category ID : 953

Summary

This category identifies Software Fault Patterns (SFPs) within the Missing Endpoint Authentication cluster (SFP30).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	898	SFP Primary Cluster: Authentication	888	2385
HasMember	V	422	Unprotected Windows Messaging Channel ('Shatter')	888	1022
HasMember	B	425	Direct Request ('Forced Browsing')	888	1025

Category-954: SFP Secondary Cluster: Multiple Binds to the Same Port

Category ID : 954

Summary

This category identifies Software Fault Patterns (SFPs) within the Multiple Binds to the Same Port cluster (SFP32).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	898	SFP Primary Cluster: Authentication	888	2385
HasMember	V	605	Multiple Binds to the Same Port	888	1356

Category-955: SFP Secondary Cluster: Unrestricted Authentication

Category ID : 955

Summary

This category identifies Software Fault Patterns (SFPs) within the Unrestricted Authentication cluster (SFP34).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	898	SFP Primary Cluster: Authentication	888	2385
HasMember	B	307	Improper Restriction of Excessive Authentication Attempts	888	747

Category-956: SFP Secondary Cluster: Channel Attack

Category ID : 956

Summary

This category identifies Software Fault Patterns (SFPs) within the Channel Attack cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	902	SFP Primary Cluster: Channel	888	2387
HasMember	B	290	Authentication Bypass by Spoofing	888	705
HasMember	B	294	Authentication Bypass by Capture-replay	888	712
HasMember	C	300	Channel Accessible by Non-Endpoint	888	730
HasMember	B	301	Reflection Attack in an Authentication Protocol	888	733
HasMember	B	419	Unprotected Primary Channel	888	1017
HasMember	B	420	Unprotected Alternate Channel	888	1018
HasMember	B	421	Race Condition During Access to Alternate Channel	888	1020
HasMember	C	441	Unintended Proxy or Intermediary ('Confused Deputy')	888	1064

Category-957: SFP Secondary Cluster: Protocol Error

Category ID : 957

Summary

This category identifies Software Fault Patterns (SFPs) within the Protocol Error cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	902	SFP Primary Cluster: Channel	888	2387
HasMember	B	353	Missing Support for Integrity Check	888	874
HasMember	P	435	Improper Interaction Between Multiple Correctly-Behaving Entities	888	1055
HasMember	C	436	Interpretation Conflict	888	1057
HasMember	B	437	Incomplete Model of Endpoint Features	888	1059
HasMember	B	757	Selection of Less-Secure Algorithm During Negotiation ('Algorithm Downgrade')	888	1581

Category-958: SFP Secondary Cluster: Broken Cryptography

Category ID : 958

Summary

This category identifies Software Fault Patterns (SFPs) within the Broken Cryptography cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	903	SFP Primary Cluster: Cryptography	888	2387
HasMember	B	325	Missing Cryptographic Step	888	794
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	888	799
HasMember	B	328	Use of Weak Hash	888	806
HasMember	V	759	Use of a One-Way Hash without a Salt	888	1585
HasMember	V	760	Use of a One-Way Hash with a Predictable Salt	888	1589

Category-959: SFP Secondary Cluster: Weak Cryptography

Category ID : 959

Summary

This category identifies Software Fault Patterns (SFPs) within the Weak Cryptography cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	903	SFP Primary Cluster: Cryptography	888	2387
HasMember	B	261	Weak Encoding for Password	888	631
HasMember	B	322	Key Exchange without Entity Authentication	888	788
HasMember	B	323	Reusing a Nonce, Key Pair in Encryption	888	790
HasMember	B	324	Use of a Key Past its Expiration Date	888	792
HasMember	C	326	Inadequate Encryption Strength	888	796
HasMember	V	329	Generation of Predictable IV with CBC Mode	888	811
HasMember	B	347	Improper Verification of Cryptographic Signature	888	857
HasMember	B	640	Weak Password Recovery Mechanism for Forgotten Password	888	1409

Category-960: SFP Secondary Cluster: Ambiguous Exception Type

Category ID : 960

Summary

This category identifies Software Fault Patterns (SFPs) within the Ambiguous Exception Type cluster (SFP5).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	889	SFP Primary Cluster: Exception Management	888	2382
HasMember	B	396	Declaration of Catch for Generic Exception	888	959
HasMember	B	397	Declaration of Throws for Generic Exception	888	961

Category-961: SFP Secondary Cluster: Incorrect Exception Behavior

Category ID : 961

Summary

This category identifies Software Fault Patterns (SFPs) within the Incorrect Exception Behavior cluster (SFP6).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	889	SFP Primary Cluster: Exception Management	888	2382
HasMember	B	392	Missing Report of Error Condition	888	951
HasMember	B	393	Return of Wrong Status Code	888	953
HasMember	B	455	Non-exit on Failed Initialization	888	1087
HasMember	B	460	Improper Cleanup on Thrown Exception	888	1102
HasMember	B	544	Missing Standardized Error Handling Mechanism	888	1256
HasMember	B	584	Return Inside Finally Block	888	1317
HasMember	C	636	Not Failing Securely ('Failing Open')	888	1401

Nature	Type	ID	Name	V	Page
HasMember	P	703	Improper Check or Handling of Exceptional Conditions	888	1535

Category-962: SFP Secondary Cluster: Unchecked Status Condition

Category ID : 962

Summary

This category identifies Software Fault Patterns (SFPs) within the Unchecked Status Condition cluster (SFP4).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	889	SFP Primary Cluster: Exception Management	888	2382
HasMember	B	248	Uncaught Exception	888	596
HasMember	B	252	Unchecked Return Value	888	606
HasMember	B	253	Incorrect Check of Function Return Value	888	613
HasMember	B	273	Improper Check for Dropped Privileges	888	660
HasMember	B	280	Improper Handling of Insufficient Permissions or Privileges	888	672
HasMember	B	372	Incomplete Internal State Distinction	888	919
HasMember	B	390	Detection of Error Condition Without Action	888	943
HasMember	B	391	Unchecked Error Condition	888	948
HasMember	B	394	Unexpected Status Code or Return Value	888	955
HasMember	B	395	Use of NullPointerException Catch to Detect NULL Pointer Dereference	888	957
HasMember	B	431	Missing Handler	888	1043
HasMember	B	478	Missing Default Case in Multiple Condition Expression	888	1142
HasMember	B	484	Omitted Break Statement in Switch	888	1162
HasMember	V	600	Uncaught Exception in Servlet	888	1343
HasMember	C	665	Improper Initialization	888	1456
HasMember	C	754	Improper Check for Unusual or Exceptional Conditions	888	1568
HasMember	C	755	Improper Handling of Exceptional Conditions	888	1576

Category-963: SFP Secondary Cluster: Exposed Data

Category ID : 963

Summary

This category identifies Software Fault Patterns (SFPs) within the Exposed Data cluster (SFP23).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	895	SFP Primary Cluster: Information Leak	888	2384
HasMember	V	5	J2EE Misconfiguration: Data Transmission Without Encryption	888	1
HasMember	V	7	J2EE Misconfiguration: Missing Custom Error Page	888	4
HasMember	V	8	J2EE Misconfiguration: Entity Bean Declared Remote	888	6
HasMember	V	11	ASP.NET Misconfiguration: Creating Debug Binary	888	9

Nature	Type	ID	Name	V	Page
HasMember	V	12	ASP.NET Misconfiguration: Missing Custom Error Page	888	11
HasMember	V	13	ASP.NET Misconfiguration: Password in Configuration File	888	13
HasMember	V	14	Compiler Removal of Code to Clear Buffers	888	14
HasMember	B	117	Improper Output Neutralization for Logs	888	288
HasMember	C	200	Exposure of Sensitive Information to an Unauthorized Actor	888	504
HasMember	B	201	Insertion of Sensitive Information Into Sent Data	888	514
HasMember	B	209	Generation of Error Message Containing Sensitive Information	888	533
HasMember	B	210	Self-generated Error Message Containing Sensitive Information	888	539
HasMember	B	211	Externally-Generated Error Message Containing Sensitive Information	888	541
HasMember	B	212	Improper Removal of Sensitive Information Before Storage or Transfer	888	544
HasMember	B	213	Exposure of Sensitive Information Due to Incompatible Policies	888	547
HasMember	B	214	Invocation of Process Using Visible Sensitive Information	888	549
HasMember	B	215	Insertion of Sensitive Information Into Debugging Code	888	551
HasMember	V	219	Storage of File with Sensitive Data Under Web Root	888	553
HasMember	V	220	Storage of File With Sensitive Data Under FTP Root	888	555
HasMember	B	226	Sensitive Information in Resource Not Removed Before Reuse	888	562
HasMember	V	244	Improper Clearing of Heap Memory Before Release ('Heap Inspection')	888	591
HasMember	B	256	Plaintext Storage of a Password	888	615
HasMember	B	257	Storing Passwords in a Recoverable Format	888	618
HasMember	B	260	Password in Configuration File	888	629
HasMember	C	311	Missing Encryption of Sensitive Data	888	757
HasMember	B	312	Cleartext Storage of Sensitive Information	888	764
HasMember	V	313	Cleartext Storage in a File or on Disk	888	770
HasMember	V	314	Cleartext Storage in the Registry	888	772
HasMember	V	315	Cleartext Storage of Sensitive Information in a Cookie	888	774
HasMember	V	316	Cleartext Storage of Sensitive Information in Memory	888	775
HasMember	V	317	Cleartext Storage of Sensitive Information in GUI	888	777
HasMember	V	318	Cleartext Storage of Sensitive Information in Executable	888	778
HasMember	B	319	Cleartext Transmission of Sensitive Information	888	779
HasMember	B	374	Passing Mutable Objects to an Untrusted Method	888	920
HasMember	B	375	Returning a Mutable Object to an Untrusted Caller	888	923
HasMember	C	402	Transmission of Private Resources into a New Sphere ('Resource Leak')	888	976
HasMember	B	403	Exposure of File Descriptor to Unintended Control Sphere ('File Descriptor Leak')	888	978
HasMember	V	433	Unparsed Raw Web Content Delivery	888	1046
HasMember	V	495	Private Data Structure Returned From A Public Method	888	1189
HasMember	B	497	Exposure of Sensitive System Information to an Unauthorized Control Sphere	888	1193
HasMember	V	498	Cloneable Class Containing Sensitive Information	888	1196
HasMember	V	499	Serializable Class Containing Sensitive Data	888	1198

Nature	Type	ID	Name	V	Page
HasMember	B	501	Trust Boundary Violation	888	1203
HasMember	C	522	Insufficiently Protected Credentials	888	1225
HasMember	B	523	Unprotected Transport of Credentials	888	1230
HasMember	V	526	Cleartext Storage of Sensitive Information in an Environment Variable	888	1234
HasMember	V	527	Exposure of Version-Control Repository to an Unauthorized Control Sphere	888	1236
HasMember	V	528	Exposure of Core Dump File to an Unauthorized Control Sphere	888	1237
HasMember	V	529	Exposure of Access Control List Files to an Unauthorized Control Sphere	888	1238
HasMember	V	530	Exposure of Backup File to an Unauthorized Control Sphere	888	1239
HasMember	B	532	Insertion of Sensitive Information into Log File	888	1241
HasMember	V	535	Exposure of Information Through Shell Error Message	888	1244
HasMember	V	536	Servlet Runtime Error Message Containing Sensitive Information	888	1245
HasMember	V	537	Java Runtime Error Message Containing Sensitive Information	888	1246
HasMember	B	538	Insertion of Sensitive Information into Externally-Accessible File or Directory	888	1248
HasMember	V	539	Use of Persistent Cookies Containing Sensitive Information	888	1250
HasMember	B	540	Inclusion of Sensitive Information in Source Code	888	1251
HasMember	V	541	Inclusion of Sensitive Information in an Include File	888	1253
HasMember	V	546	Suspicious Comment	888	1258
HasMember	V	548	Exposure of Information Through Directory Listing	888	1261
HasMember	V	550	Server-generated Error Message Containing Sensitive Information	888	1263
HasMember	B	552	Files or Directories Accessible to External Parties	888	1265
HasMember	V	555	J2EE Misconfiguration: Plaintext Password in Configuration File	888	1270
HasMember	V	591	Sensitive Data Storage in Improperly Locked Memory	888	1329
HasMember	V	598	Use of GET Request Method With Sensitive Query Strings	888	1340
HasMember	V	607	Public Static Final Field References Mutable Object	888	1360
HasMember	B	612	Improper Authorization of Index Containing Sensitive Information	888	1370
HasMember	V	615	Inclusion of Sensitive Information in Source Code Comments	888	1375
HasMember	C	642	External Control of Critical State Data	888	1414
HasMember	C	668	Exposure of Resource to Wrong Sphere	888	1469
HasMember	C	669	Incorrect Resource Transfer Between Spheres	888	1471
HasMember	B	756	Missing Custom Error Page	888	1579
HasMember	B	767	Access to Critical Private Variable via Public Method	888	1610

Category-964: SFP Secondary Cluster: Exposure Temporary File

Category ID : 964

Summary

This category identifies Software Fault Patterns (SFPs) within the Exposure Temporary File cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	895	SFP Primary Cluster: Information Leak	888	2384
HasMember	C	377	Insecure Temporary File	888	925
HasMember	B	378	Creation of Temporary File With Insecure Permissions	888	928
HasMember	B	379	Creation of Temporary File in Directory with Insecure Permissions	888	930

Category-965: SFP Secondary Cluster: Insecure Session Management

Category ID : 965

Summary

This category identifies Software Fault Patterns (SFPs) within the Insecure Session Management cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	895	SFP Primary Cluster: Information Leak	888	2384
HasMember	V	6	J2EE Misconfiguration: Insufficient Session-ID Length	888	2
HasMember	B	488	Exposure of Data Element to Wrong Session	888	1169
HasMember	B	524	Use of Cache Containing Sensitive Information	888	1232

Category-966: SFP Secondary Cluster: Other Exposures

Category ID : 966

Summary

This category identifies Software Fault Patterns (SFPs) within the Other Exposures cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	895	SFP Primary Cluster: Information Leak	888	2384
HasMember	V	453	Insecure Default Variable Initialization	888	1083
HasMember	B	487	Reliance on Package-level Scope	888	1167
HasMember	V	492	Use of Inner Class Containing Sensitive Data	888	1175
HasMember	V	525	Use of Web Browser Cache Containing Sensitive Information	888	1233
HasMember	V	614	Sensitive Cookie in HTTPS Session Without 'Secure' Attribute	888	1373
HasMember	V	651	Exposure of WSDL File Containing Sensitive Information	888	1433

Category-967: SFP Secondary Cluster: State Disclosure

Category ID : 967

Summary

This category identifies Software Fault Patterns (SFPs) within the State Disclosure cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	895	SFP Primary Cluster: Information Leak	888	2384
HasMember	B	202	Exposure of Sensitive Information Through Data Queries	888	516
HasMember	B	203	Observable Discrepancy	888	518
HasMember	B	204	Observable Response Discrepancy	888	523
HasMember	B	205	Observable Behavioral Discrepancy	888	526
HasMember	V	206	Observable Internal Behavioral Discrepancy	888	527
HasMember	V	207	Observable Behavioral Discrepancy With Equivalent Products	888	528
HasMember	B	208	Observable Timing Discrepancy	888	529

Category-968: SFP Secondary Cluster: Covert Channel

Category ID : 968

Summary

This category identifies Software Fault Patterns (SFPs) within the Covert Channel cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	904	SFP Primary Cluster: Malware	888	2387
HasMember	B	385	Covert Timing Channel	888	940
HasMember	C	514	Covert Channel	888	1218
HasMember	B	515	Covert Storage Channel	888	1220

Category-969: SFP Secondary Cluster: Faulty Memory Release

Category ID : 969

Summary

This category identifies Software Fault Patterns (SFPs) within the Faulty Memory Release cluster (SFP12).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	891	SFP Primary Cluster: Memory Management	888	2383
HasMember	V	415	Double Free	888	1008
HasMember	V	590	Free of Memory not on the Heap	888	1326
HasMember	V	761	Free of Pointer not at Start of Buffer	888	1592
HasMember	B	763	Release of Invalid Pointer or Reference	888	1599

Category-970: SFP Secondary Cluster: Faulty Buffer Access

Category ID : 970

Summary

This category identifies Software Fault Patterns (SFPs) within the Faulty Buffer Access cluster (SFP8).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	890	SFP Primary Cluster: Memory Access	888	2383
HasMember	C	118	Incorrect Access of Indexable Resource ('Range Error')	888	292
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	888	293
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	888	304
HasMember	V	121	Stack-based Buffer Overflow	888	314
HasMember	V	122	Heap-based Buffer Overflow	888	318
HasMember	B	123	Write-what-where Condition	888	323
HasMember	B	124	Buffer Underwrite ('Buffer Underflow')	888	326
HasMember	B	125	Out-of-bounds Read	888	330
HasMember	V	126	Buffer Over-read	888	334
HasMember	V	127	Buffer Under-read	888	337
HasMember	V	129	Improper Validation of Array Index	888	341

Category-971: SFP Secondary Cluster: Faulty Pointer Use

Category ID : 971

Summary

This category identifies Software Fault Patterns (SFPs) within the Faulty Pointer Use cluster (SFP7).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	890	SFP Primary Cluster: Memory Access	888	2383
HasMember	B	469	Use of Pointer Subtraction to Determine Size	888	1115
HasMember	B	476	NULL Pointer Dereference	888	1132
HasMember	V	588	Attempt to Access Child of a Non-structure Pointer	888	1323

Category-972: SFP Secondary Cluster: Faulty String Expansion

Category ID : 972

Summary

This category identifies Software Fault Patterns (SFPs) within the Faulty String Expansion cluster (SFP9).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	890	SFP Primary Cluster: Memory Access	888	2383
HasMember	V	785	Use of Path Manipulation Function without Maximum-sized Buffer	888	1656

Category-973: SFP Secondary Cluster: Improper NULL Termination

Category ID : 973

Summary

This category identifies Software Fault Patterns (SFPs) within the Improper NULL Termination cluster (SFP11).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	890	SFP Primary Cluster: Memory Access	888	2383
HasMember	B	170	Improper Null Termination	888	428

Category-974: SFP Secondary Cluster: Incorrect Buffer Length Computation

Category ID : 974

Summary

This category identifies Software Fault Patterns (SFPs) within the Incorrect Buffer Length Computation cluster (SFP10).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	890	SFP Primary Cluster: Memory Access	888	2383
HasMember	B	131	Incorrect Calculation of Buffer Size	888	355
HasMember	B	135	Incorrect Calculation of Multi-Byte String Length	888	370
HasMember	C	251	Often Misused: String Management	888	2314
HasMember	V	467	Use of sizeof() on a Pointer Type	888	1110

Category-975: SFP Secondary Cluster: Architecture

Category ID : 975

Summary

This category identifies Software Fault Patterns (SFPs) within the Architecture cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	907	SFP Primary Cluster: Other	888	2388
HasMember	B	348	Use of Less Trusted Source	888	859
HasMember	B	359	Exposure of Private Personal Information to an Unauthorized Actor	888	882
HasMember	C	602	Client-Side Enforcement of Server-Side Security	888	1350

Nature	Type	ID	Name	V	Page
HasMember	C	637	Unnecessary Complexity in Protection Mechanism (Not Using 'Economy of Mechanism')	888	1403
HasMember	B	649	Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking	888	1430
HasMember	B	654	Reliance on a Single Factor in a Security Decision	888	1439
HasMember	C	656	Reliance on Security Through Obscurity	888	1444
HasMember	C	657	Violation of Secure Design Principles	888	1446
HasMember	C	671	Lack of Administrator Control over Security	888	1478
HasMember	P	693	Protection Mechanism Failure	888	1520
HasMember	B	749	Exposed Dangerous Method or Function	888	1564

Category-976: SFP Secondary Cluster: Compiler

Category ID : 976

Summary

This category identifies Software Fault Patterns (SFPs) within the Compiler cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	907	SFP Primary Cluster: Other	888	2388
HasMember	B	733	Compiler Optimization Removal or Modification of Security-critical Code	888	1562

Category-977: SFP Secondary Cluster: Design

Category ID : 977

Summary

This category identifies Software Fault Patterns (SFPs) within the Design cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	907	SFP Primary Cluster: Other	888	2388
HasMember	B	115	Misinterpretation of Input	888	280
HasMember	V	187	Partial String Comparison	888	467
HasMember	B	188	Reliance on Data/Memory Layout	888	470
HasMember	B	193	Off-by-one Error	888	486
HasMember	B	349	Acceptance of Extraneous Untrusted Data With Trusted Data	888	861
HasMember	C	405	Asymmetric Resource Consumption (Amplification)	888	986
HasMember	C	406	Insufficient Control of Network Message Volume (Network Amplification)	888	990
HasMember	C	407	Inefficient Algorithmic Complexity	888	992
HasMember	B	408	Incorrect Behavior Order: Early Amplification	888	995
HasMember	B	409	Improper Handling of Highly Compressed Data (Data Amplification)	888	996
HasMember	B	410	Insufficient Resource Pool	888	998

Nature	Type	ID	Name	V	Page
HasMember	B	430	Deployment of Wrong Handler	888	1042
HasMember	V	462	Duplicate Key in Associative List (Alist)	888	1104
HasMember	B	463	Deletion of Data Structure Sentinel	888	1105
HasMember	B	464	Addition of Data Structure Sentinel	888	1107
HasMember	B	483	Incorrect Block Delimitation	888	1160
HasMember	V	581	Object Model Violation: Just One of Equals and Hashcode Defined	888	1312
HasMember	V	595	Comparison of Object References Instead of Object Contents	888	1334
HasMember	V	618	Exposed Unsafe ActiveX Method	888	1380
HasMember	B	648	Incorrect Use of Privileged APIs	888	1428
HasMember	C	670	Always-Incorrect Control Flow Implementation	888	1475
HasMember	P	682	Incorrect Calculation	888	1499
HasMember	P	691	Insufficient Control Flow Management	888	1517
HasMember	C	696	Incorrect Behavior Order	888	1527
HasMember	P	697	Incorrect Comparison	888	1530
HasMember	B	698	Execution After Redirect (EAR)	888	1533
HasMember	C	705	Incorrect Control Flow Scoping	888	1542

Category-978: SFP Secondary Cluster: Implementation

Category ID : 978

Summary

This category identifies Software Fault Patterns (SFPs) within the Implementation cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	907	SFP Primary Cluster: Other	888	2388
HasMember	B	358	Improperly Implemented Security Check for Standard	888	881
HasMember	C	398	7PK - Code Quality	888	2323
HasMember	V	623	Unsafe ActiveX Control Marked Safe For Scripting	888	1389
HasMember	P	710	Improper Adherence to Coding Standards	888	1549

Category-979: SFP Secondary Cluster: Failed Chroot Jail

Category ID : 979

Summary

This category identifies Software Fault Patterns (SFPs) within the Failed Chroot Jail cluster (SFP17).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	893	SFP Primary Cluster: Path Resolution	888	2384
HasMember	V	243	Creation of chroot Jail Without Changing Working Directory	888	589

Category-980: SFP Secondary Cluster: Link in Resource Name Resolution

Category ID : 980

Summary

This category identifies Software Fault Patterns (SFPs) within the Link in Resource Name Resolution cluster (SFP18).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	893	SFP Primary Cluster: Path Resolution	888	2384
HasMember	B	59	Improper Link Resolution Before File Access ('Link Following')	888	111
HasMember	V	62	UNIX Hard Link	888	119
HasMember	V	64	Windows Shortcut Following (.LNK)	888	121
HasMember	V	65	Windows Hard Link	888	123
HasMember	B	386	Symbolic Name not Mapping to Correct Object	888	942
HasMember	C	610	Externally Controlled Reference to a Resource in Another Sphere	888	1364

Category-981: SFP Secondary Cluster: Path Traversal

Category ID : 981

Summary

This category identifies Software Fault Patterns (SFPs) within the Path Traversal cluster (SFP16).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	893	SFP Primary Cluster: Path Resolution	888	2384
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	888	33
HasMember	B	23	Relative Path Traversal	888	46
HasMember	V	24	Path Traversal: '../filedir'	888	53
HasMember	V	25	Path Traversal: '/../filedir'	888	54
HasMember	V	26	Path Traversal: '/dir/../filename'	888	56
HasMember	V	27	Path Traversal: 'dir/../../filename'	888	58
HasMember	V	28	Path Traversal: '..\filedir'	888	59
HasMember	V	29	Path Traversal: '\..\filename'	888	61
HasMember	V	30	Path Traversal: '\dir\..\filename'	888	63
HasMember	V	31	Path Traversal: 'dir\..\..\filename'	888	65
HasMember	V	32	Path Traversal: '...' (Triple Dot)	888	67
HasMember	V	33	Path Traversal: '....' (Multiple Dot)	888	69
HasMember	V	34	Path Traversal: '....//'	888	71
HasMember	V	35	Path Traversal: '.../...//'	888	73
HasMember	B	36	Absolute Path Traversal	888	75
HasMember	V	37	Path Traversal: '/absolute/pathname/here'	888	79
HasMember	V	38	Path Traversal: '\absolute\pathname\here'	888	80
HasMember	V	39	Path Traversal: 'C:dirname'	888	82
HasMember	V	40	Path Traversal: '\\UNC\share\name\' (Windows UNC Share)	888	85

Nature	Type	ID	Name	V	Page
HasMember	B	41	Improper Resolution of Path Equivalence	888	86
HasMember	V	42	Path Equivalence: 'filename.' (Trailing Dot)	888	92
HasMember	V	43	Path Equivalence: 'filename....' (Multiple Trailing Dot)	888	93
HasMember	V	44	Path Equivalence: 'file.name' (Internal Dot)	888	94
HasMember	V	45	Path Equivalence: 'file...name' (Multiple Internal Dot)	888	95
HasMember	V	46	Path Equivalence: 'filename ' (Trailing Space)	888	96
HasMember	V	47	Path Equivalence: ' filename' (Leading Space)	888	97
HasMember	V	48	Path Equivalence: 'file name' (Internal Whitespace)	888	98
HasMember	V	49	Path Equivalence: 'filename/' (Trailing Slash)	888	99
HasMember	V	50	Path Equivalence: '//multiple/leading/slash'	888	100
HasMember	V	51	Path Equivalence: '/multiple//internal/slash'	888	102
HasMember	V	52	Path Equivalence: '/multiple/trailing/slash//'	888	103
HasMember	V	53	Path Equivalence: '\multiple\\internal\backslash'	888	104
HasMember	V	54	Path Equivalence: 'filedir\' (Trailing Backslash)	888	105
HasMember	V	55	Path Equivalence: '/./' (Single Dot Directory)	888	106
HasMember	V	56	Path Equivalence: 'filedir*' (Wildcard)	888	107
HasMember	V	57	Path Equivalence: 'fakedir/../realdir/filename'	888	108
HasMember	V	58	Path Equivalence: Windows 8.3 Filename	888	110
HasMember	B	66	Improper Handling of File Names that Identify Virtual Resources	888	124
HasMember	V	67	Improper Handling of Windows Device Names	888	126
HasMember	V	72	Improper Handling of Apple HFS+ Alternate Data Stream Path	888	130
HasMember	B	73	External Control of File Name or Path	888	132
HasMember	B	428	Unquoted Search Path or Element	888	1039
HasMember	C	706	Use of Incorrectly-Resolved Name or Reference	888	1544

Category-982: SFP Secondary Cluster: Failure to Release Resource

Category ID : 982

Summary

This category identifies Software Fault Patterns (SFPs) within the Failure to Release Resource cluster (SFP14).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	892	SFP Primary Cluster: Resource Management	888	2383
HasMember	C	404	Improper Resource Shutdown or Release	888	980
HasMember	B	459	Incomplete Cleanup	888	1099
HasMember	B	771	Missing Reference to Active Allocated Resource	888	1622
HasMember	B	772	Missing Release of Resource after Effective Lifetime	888	1624
HasMember	V	773	Missing Reference to Active File Descriptor or Handle	888	1629
HasMember	V	775	Missing Release of File Descriptor or Handle after Effective Lifetime	888	1631

Category-983: SFP Secondary Cluster: Faulty Resource Use

Category ID : 983

Summary

This category identifies Software Fault Patterns (SFPs) within the Faulty Resource Use cluster (SFP15).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	892	SFP Primary Cluster: Resource Management	888	2383
HasMember	V	416	Use After Free	888	1012
HasMember	C	672	Operation on a Resource after Expiration or Release	888	1479

Category-984: SFP Secondary Cluster: Life Cycle

Category ID : 984

Summary

This category identifies Software Fault Patterns (SFPs) within the Life Cycle cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	892	SFP Primary Cluster: Resource Management	888	2383
HasMember	P	664	Improper Control of a Resource Through its Lifetime	888	1454
HasMember	C	666	Operation on Resource in Wrong Phase of Lifetime	888	1462
HasMember	C	675	Multiple Operations on Resource in Single-Operation Context	888	1487
HasMember	B	694	Use of Multiple Resources with Duplicate Identifier	888	1523

Category-985: SFP Secondary Cluster: Unrestricted Consumption

Category ID : 985

Summary

This category identifies Software Fault Patterns (SFPs) within the Unrestricted Consumption cluster (SFP13).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	892	SFP Primary Cluster: Resource Management	888	2383
HasMember	C	400	Uncontrolled Resource Consumption	888	964
HasMember	C	674	Uncontrolled Recursion	888	1484
HasMember	B	770	Allocation of Resources Without Limits or Throttling	888	1613
HasMember	V	774	Allocation of File Descriptors or Handles Without Limits or Throttling	888	1630

Category-986: SFP Secondary Cluster: Missing Lock

Category ID : 986

Summary

This category identifies Software Fault Patterns (SFPs) within the Missing Lock cluster (SFP19).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	894	SFP Primary Cluster: Synchronization	888	2384
HasMember	B	364	Signal Handler Race Condition	888	899
HasMember	B	366	Race Condition within a Thread	888	904
HasMember	B	368	Context Switching Race Condition	888	912
HasMember	B	413	Improper Resource Locking	888	1003
HasMember	B	414	Missing Lock Check	888	1007
HasMember	V	543	Use of Singleton Pattern Without Synchronization in a Multithreaded Context	888	1255
HasMember	B	567	Unsynchronized Access to Shared Data in a Multithreaded Context	888	1288
HasMember	B	609	Double-Checked Locking	888	1362
HasMember	C	662	Improper Synchronization	888	1448
HasMember	B	663	Use of a Non-reentrant Function in a Concurrent Context	888	1452
HasMember	C	667	Improper Locking	888	1464

Category-987: SFP Secondary Cluster: Multiple Locks/Unlocks

Category ID : 987

Summary

This category identifies Software Fault Patterns (SFPs) within the Multiple Locks/Unlocks cluster (SFP21).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	894	SFP Primary Cluster: Synchronization	888	2384
HasMember	V	585	Empty Synchronized Block	888	1318
HasMember	B	764	Multiple Locks of a Critical Resource	888	1604
HasMember	B	765	Multiple Unlocks of a Critical Resource	888	1605

Category-988: SFP Secondary Cluster: Race Condition Window

Category ID : 988

Summary

This category identifies Software Fault Patterns (SFPs) within the Race Condition Window cluster (SFP20).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	894	SFP Primary Cluster: Synchronization	888	2384

Nature	Type	ID	Name	V	Page
HasMember	C	362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	888	888
HasMember	B	363	Race Condition Enabling Link Following	888	897
HasMember	B	367	Time-of-check Time-of-use (TOCTOU) Race Condition	888	906
HasMember	V	370	Missing Check for Certificate Revocation after Initial Check	888	917
HasMember	C	638	Not Using Complete Mediation	888	1404

Category-989: SFP Secondary Cluster: Unrestricted Lock

Category ID : 989

Summary

This category identifies Software Fault Patterns (SFPs) within the Unrestricted Lock cluster (SFP22).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	894	SFP Primary Cluster: Synchronization	888	2384
HasMember	B	412	Unrestricted Externally Accessible Lock	888	1000

Category-990: SFP Secondary Cluster: Tainted Input to Command

Category ID : 990

Summary

This category identifies Software Fault Patterns (SFPs) within the Tainted Input to Command cluster (SFP24).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	896	SFP Primary Cluster: Tainted Input	888	2385
HasMember	C	74	Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')	888	137
HasMember	C	75	Failure to Sanitize Special Elements into a Different Plane (Special Element Injection)	888	142
HasMember	B	76	Improper Neutralization of Equivalent Special Elements	888	144
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	888	145
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	888	151
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	888	163
HasMember	V	80	Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)	888	177
HasMember	V	81	Improper Neutralization of Script in an Error Message Web Page	888	179
HasMember	V	82	Improper Neutralization of Script in Attributes of IMG Tags in a Web Page	888	182

Nature	Type	ID	Name	V	Page
HasMember	V	83	Improper Neutralization of Script in Attributes in a Web Page	888	183
HasMember	V	84	Improper Neutralization of Encoded URI Schemes in a Web Page	888	186
HasMember	V	85	Doubled Character XSS Manipulations	888	188
HasMember	V	86	Improper Neutralization of Invalid Characters in Identifiers in Web Pages	888	190
HasMember	V	87	Improper Neutralization of Alternate XSS Syntax	888	192
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	888	194
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	888	201
HasMember	B	90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	888	212
HasMember	B	91	XML Injection (aka Blind XPath Injection)	888	215
HasMember	B	93	Improper Neutralization of CRLF Sequences ('CRLF Injection')	888	217
HasMember	V	95	Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')	888	226
HasMember	B	96	Improper Neutralization of Directives in Statically Saved Code ('Static Code Injection')	888	232
HasMember	V	97	Improper Neutralization of Server-Side Includes (SSI) Within a Web Page	888	235
HasMember	C	99	Improper Control of Resource Identifiers ('Resource Injection')	888	243
HasMember	V	102	Struts: Duplicate Validation Forms	888	246
HasMember	V	103	Struts: Incomplete validate() Method Definition	888	248
HasMember	V	104	Struts: Form Bean Does Not Extend Validation Class	888	251
HasMember	V	105	Struts: Form Field Without Validator	888	253
HasMember	V	106	Struts: Plug-in Framework not in Use	888	256
HasMember	V	107	Struts: Unused Validation Form	888	259
HasMember	V	108	Struts: Unvalidated Action Form	888	261
HasMember	V	109	Struts: Validator Turned Off	888	263
HasMember	V	110	Struts: Validator Without Form Field	888	264
HasMember	B	112	Missing XML Validation	888	269
HasMember	V	113	Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Request/Response Splitting')	888	271
HasMember	B	130	Improper Handling of Length Parameter Inconsistency	888	351
HasMember	B	134	Use of Externally-Controlled Format String	888	365
HasMember	C	138	Improper Neutralization of Special Elements	888	373
HasMember	B	140	Improper Neutralization of Delimiters	888	376
HasMember	V	141	Improper Neutralization of Parameter/Argument Delimiters	888	378
HasMember	V	142	Improper Neutralization of Value Delimiters	888	380
HasMember	V	143	Improper Neutralization of Record Delimiters	888	381
HasMember	V	144	Improper Neutralization of Line Delimiters	888	383
HasMember	V	145	Improper Neutralization of Section Delimiters	888	385
HasMember	V	146	Improper Neutralization of Expression/Command Delimiters	888	387
HasMember	V	147	Improper Neutralization of Input Terminators	888	389
HasMember	V	148	Improper Neutralization of Input Leaders	888	391

Nature	Type	ID	Name	V	Page
HasMember	V	149	Improper Neutralization of Quoting Syntax	888	392
HasMember	V	150	Improper Neutralization of Escape, Meta, or Control Sequences	888	394
HasMember	V	151	Improper Neutralization of Comment Delimiters	888	396
HasMember	V	152	Improper Neutralization of Macro Symbols	888	398
HasMember	V	153	Improper Neutralization of Substitution Characters	888	400
HasMember	V	154	Improper Neutralization of Variable Name Delimiters	888	401
HasMember	V	155	Improper Neutralization of Wildcards or Matching Symbols	888	403
HasMember	V	156	Improper Neutralization of Whitespace	888	405
HasMember	V	157	Failure to Sanitize Paired Delimiters	888	407
HasMember	V	158	Improper Neutralization of Null Byte or NUL Character	888	409
HasMember	C	159	Improper Handling of Invalid Use of Special Elements	888	411
HasMember	V	160	Improper Neutralization of Leading Special Elements	888	413
HasMember	V	161	Improper Neutralization of Multiple Leading Special Elements	888	415
HasMember	V	162	Improper Neutralization of Trailing Special Elements	888	417
HasMember	V	163	Improper Neutralization of Multiple Trailing Special Elements	888	418
HasMember	V	164	Improper Neutralization of Internal Special Elements	888	420
HasMember	V	165	Improper Neutralization of Multiple Internal Special Elements	888	422
HasMember	B	183	Permissive List of Allowed Inputs	888	458
HasMember	B	184	Incomplete List of Disallowed Inputs	888	459
HasMember	C	185	Incorrect Regular Expression	888	463
HasMember	B	186	Overly Restrictive Regular Expression	888	466
HasMember	B	444	Inconsistent Interpretation of HTTP Requests ('HTTP Request/Response Smuggling')	888	1068
HasMember	V	553	Command Shell in Externally Accessible Directory	888	1269
HasMember	V	554	ASP.NET Misconfiguration: Not Using Input Validation Framework	888	1269
HasMember	V	564	SQL Injection: Hibernate	888	1282
HasMember	B	601	URL Redirection to Untrusted Site ('Open Redirect')	888	1345
HasMember	B	611	Improper Restriction of XML External Entity Reference	888	1367
HasMember	B	619	Dangling Database Cursor ('Cursor Injection')	888	1382
HasMember	V	621	Variable Extraction Error	888	1385
HasMember	B	624	Executable Regular Expression Error	888	1390
HasMember	B	625	Permissive Regular Expression	888	1392
HasMember	V	626	Null Byte Interaction Error (Poison Null Byte)	888	1394
HasMember	V	627	Dynamic Variable Evaluation	888	1396
HasMember	B	641	Improper Restriction of Names for Files and Other Resources	888	1412
HasMember	B	643	Improper Neutralization of Data within XPath Expressions ('XPath Injection')	888	1419
HasMember	V	644	Improper Neutralization of HTTP Headers for Scripting Syntax	888	1422
HasMember	V	646	Reliance on File Name or Extension of Externally-Supplied File	888	1425
HasMember	B	652	Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')	888	1435

Nature	Type	ID	Name	V	Page
HasMember	V	687	Function Call With Incorrectly Specified Argument Value	888	1510
HasMember	P	707	Improper Neutralization	888	1546

Category-991: SFP Secondary Cluster: Tainted Input to Environment

Category ID : 991

Summary

This category identifies Software Fault Patterns (SFPs) within the Tainted Input to Environment cluster (SFP27).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	896	SFP Primary Cluster: Tainted Input	888	2385
HasMember	B	94	Improper Control of Generation of Code ('Code Injection')	888	219
HasMember	C	114	Process Control	888	277
HasMember	B	427	Uncontrolled Search Path Element	888	1033
HasMember	B	470	Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')	888	1118
HasMember	B	471	Modification of Assumed-Immutable Data (MAID)	888	1121
HasMember	B	472	External Control of Assumed-Immutable Web Parameter	888	1123
HasMember	V	473	PHP External Variable Modification	888	1127
HasMember	B	494	Download of Code Without Integrity Check	888	1185
HasMember	V	622	Improper Validation of Function Hook Arguments	888	1387
HasMember	C	673	External Influence of Sphere Definition	888	1483

Category-992: SFP Secondary Cluster: Faulty Input Transformation

Category ID : 992

Summary

This category identifies Software Fault Patterns (SFPs) within the Faulty Input Transformation cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	896	SFP Primary Cluster: Tainted Input	888	2385
HasMember	C	116	Improper Encoding or Escaping of Output	888	281
HasMember	B	166	Improper Handling of Missing Special Element	888	423
HasMember	B	167	Improper Handling of Additional Special Element	888	425
HasMember	B	168	Improper Handling of Inconsistent Special Elements	888	426
HasMember	C	172	Encoding Error	888	433
HasMember	V	173	Improper Handling of Alternate Encoding	888	435
HasMember	V	174	Double Decoding of the Same Data	888	437
HasMember	V	175	Improper Handling of Mixed Encoding	888	439
HasMember	V	176	Improper Handling of Unicode Encoding	888	440

Nature	Type	ID	Name	V	Page
HasMember	V	177	Improper Handling of URL Encoding (Hex Encoding)	888	442
HasMember	B	178	Improper Handling of Case Sensitivity	888	445
HasMember	B	179	Incorrect Behavior Order: Early Validation	888	448
HasMember	V	180	Incorrect Behavior Order: Validate Before Canonicalize	888	451
HasMember	V	181	Incorrect Behavior Order: Validate Before Filter	888	453
HasMember	B	182	Collapse of Data into Unsafe Value	888	455

Category-993: SFP Secondary Cluster: Incorrect Input Handling

Category ID : 993

Summary

This category identifies Software Fault Patterns (SFPs) within the Incorrect Input Handling cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	896	SFP Primary Cluster: Tainted Input	888	2385
HasMember	V	198	Use of Incorrect Byte Ordering	888	503
HasMember	C	228	Improper Handling of Syntactically Invalid Structure	888	568
HasMember	B	229	Improper Handling of Values	888	570
HasMember	V	230	Improper Handling of Missing Values	888	570
HasMember	V	231	Improper Handling of Extra Values	888	572
HasMember	V	232	Improper Handling of Undefined Values	888	573
HasMember	B	233	Improper Handling of Parameters	888	574
HasMember	V	234	Failure to Handle Missing Parameter	888	576
HasMember	V	235	Improper Handling of Extra Parameters	888	578
HasMember	V	236	Improper Handling of Undefined Parameters	888	579
HasMember	B	237	Improper Handling of Structural Elements	888	580
HasMember	V	238	Improper Handling of Incomplete Structural Elements	888	581
HasMember	V	239	Failure to Handle Incomplete Element	888	582
HasMember	B	240	Improper Handling of Inconsistent Structural Elements	888	583
HasMember	B	241	Improper Handling of Unexpected Data Type	888	584
HasMember	B	351	Insufficient Type Distinction	888	866
HasMember	B	354	Improper Validation of Integrity Check Value	888	876

Category-994: SFP Secondary Cluster: Tainted Input to Variable

Category ID : 994

Summary

This category identifies Software Fault Patterns (SFPs) within the Tainted Input to Variable cluster (SFP25).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	896	SFP Primary Cluster: Tainted Input	888	2385
HasMember	B	15	External Control of System or Configuration Setting	888	17
HasMember	C	20	Improper Input Validation	888	20

Nature	Type	ID	Name	V	Page
HasMember	B	454	External Initialization of Trusted Variables or Data Stores	888	1085
HasMember	V	496	Public Data Assigned to Private Array-Typed Field	888	1192
HasMember	B	502	Deserialization of Untrusted Data	888	1204
HasMember	V	566	Authorization Bypass Through User-Controlled SQL Primary Key	888	1286
HasMember	B	606	Unchecked Input for Loop Condition	888	1357
HasMember	V	616	Incomplete Identification of Uploaded File Variables (PHP)	888	1376

Category-995: SFP Secondary Cluster: Feature

Category ID : 995

Summary

This category identifies Software Fault Patterns (SFPs) within the Feature cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	906	SFP Primary Cluster: UI	888	2388
HasMember	B	447	Unimplemented or Unsupported Feature in UI	888	1075
HasMember	B	448	Obsolete Feature in UI	888	1076
HasMember	B	449	The UI Performs the Wrong Action	888	1077
HasMember	B	450	Multiple Interpretations of UI Input	888	1078
HasMember	C	451	User Interface (UI) Misrepresentation of Critical Information	888	1079
HasMember	B	549	Missing Password Field Masking	888	1262
HasMember	C	655	Insufficient Psychological Acceptability	888	1442

Category-996: SFP Secondary Cluster: Security

Category ID : 996

Summary

This category identifies Software Fault Patterns (SFPs) within the Security cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	906	SFP Primary Cluster: UI	888	2388
HasMember	B	356	Product UI does not Warn User of Unsafe Actions	888	879
HasMember	B	357	Insufficient UI Warning of Dangerous Operations	888	880
HasMember	C	446	UI Discrepancy for Security Feature	888	1073

Category-997: SFP Secondary Cluster: Information Loss

Category ID : 997

Summary

This category identifies Software Fault Patterns (SFPs) within the Information Loss cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	906	SFP Primary Cluster: UI	888	2388
HasMember	C	221	Information Loss or Omission	888	556
HasMember	B	222	Truncation of Security-relevant Information	888	557
HasMember	B	223	Omission of Security-relevant Information	888	559
HasMember	B	224	Obscured Security-relevant Information by Alternate Name	888	561

Category-998: SFP Secondary Cluster: Glitch in Computation

Category ID : 998

Summary

This category identifies Software Fault Patterns (SFPs) within the Glitch in Computation cluster (SFP1).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	885	SFP Primary Cluster: Risky Values	888	2382
HasMember	B	128	Wrap-around Error	888	339
HasMember	B	190	Integer Overflow or Wraparound	888	472
HasMember	B	191	Integer Underflow (Wrap or Wraparound)	888	480
HasMember	V	194	Unexpected Sign Extension	888	491
HasMember	V	195	Signed to Unsigned Conversion Error	888	494
HasMember	V	196	Unsigned to Signed Conversion Error	888	498
HasMember	B	197	Numeric Truncation Error	888	500
HasMember	B	369	Divide By Zero	888	913
HasMember	V	456	Missing Initialization of a Variable	888	1089
HasMember	V	457	Use of Uninitialized Variable	888	1094
HasMember	B	466	Return of Pointer Value Outside of Expected Range	888	1109
HasMember	B	468	Incorrect Pointer Scaling	888	1114
HasMember	B	475	Undefined Behavior for Input to API	888	1130
HasMember	B	480	Use of Incorrect Operator	888	1150
HasMember	V	481	Assigning instead of Comparing	888	1154
HasMember	V	486	Comparison of Classes by Name	888	1164
HasMember	B	562	Return of Stack Variable Address	888	1278
HasMember	B	570	Expression is Always False	888	1292
HasMember	B	571	Expression is Always True	888	1295
HasMember	V	579	J2EE Bad Practices: Non-serializable Object Stored in Session	888	1309
HasMember	V	587	Assignment of a Fixed Address to a Pointer	888	1322
HasMember	V	594	J2EE Framework: Saving Unserializable Objects to Disk	888	1332
HasMember	V	597	Use of Wrong Operator in String Comparison	888	1337
HasMember	B	628	Function Call with Incorrectly Specified Arguments	888	1398
HasMember	B	681	Incorrect Conversion between Numeric Types	888	1495
HasMember	V	683	Function Call With Incorrect Order of Arguments	888	1504

Nature	Type	ID	Name	V	Page
HasMember	V	685	Function Call With Incorrect Number of Arguments	888	1507
HasMember	V	686	Function Call With Incorrect Argument Type	888	1508
HasMember	V	688	Function Call With Incorrect Variable or Reference as Argument	888	1511
HasMember	C	704	Incorrect Type Conversion or Cast	888	1538
HasMember	V	768	Incorrect Short Circuit Evaluation	888	1612

Category-1001: SFP Secondary Cluster: Use of an Improper API

Category ID : 1001

Summary

This category identifies Software Fault Patterns (SFPs) within the Use of an Improper API cluster (SFP3).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	887	SFP Primary Cluster: API	888	2382
HasMember	V	111	Direct Use of Unsafe JNI	888	266
HasMember	C	227	7PK - API Abuse	888	2313
HasMember	B	242	Use of Inherently Dangerous Function	888	586
HasMember	V	245	J2EE Bad Practices: Direct Management of Connections	888	592
HasMember	V	246	J2EE Bad Practices: Direct Use of Sockets	888	594
HasMember	V	382	J2EE Bad Practices: Use of System.exit()	888	933
HasMember	V	383	J2EE Bad Practices: Direct Use of Threads	888	935
HasMember	B	432	Dangerous Signal Handler not Disabled During Sensitive Operations	888	1045
HasMember	B	439	Behavioral Change in New Version or Environment	888	1061
HasMember	B	440	Expected Behavior Violation	888	1062
HasMember	B	474	Use of Function with Inconsistent Implementations	888	1128
HasMember	B	477	Use of Obsolete Function	888	1138
HasMember	V	479	Signal Handler Use of a Non-reentrant Function	888	1147
HasMember	V	558	Use of getlogin() in Multithreaded Application	888	1272
HasMember	V	572	Call to Thread run() instead of start()	888	1296
HasMember	C	573	Improper Following of Specification by Caller	888	1298
HasMember	V	574	EJB Bad Practices: Use of Synchronization Primitives	888	1300
HasMember	V	575	EJB Bad Practices: Use of AWT Swing	888	1301
HasMember	V	576	EJB Bad Practices: Use of Java I/O	888	1304
HasMember	V	577	EJB Bad Practices: Use of Sockets	888	1305
HasMember	V	578	EJB Bad Practices: Use of Class Loader	888	1307
HasMember	B	586	Explicit Call to Finalize()	888	1320
HasMember	V	589	Call to Non-ubiquitous API	888	1325
HasMember	B	617	Reachable Assertion	888	1378
HasMember	B	676	Use of Potentially Dangerous Function	888	1489
HasMember	C	684	Incorrect Provision of Specified Functionality	888	1505
HasMember	B	695	Use of Low-Level Functionality	888	1524

Nature	Type	ID	Name	V	Page
HasMember	C	758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	888	1582

Category-1002: SFP Secondary Cluster: Unexpected Entry Points

Category ID : 1002

Summary

This category identifies Software Fault Patterns (SFPs) within the Unexpected Entry Points cluster.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	C	897	SFP Primary Cluster: Entry Points	888	2385
HasMember	B	489	Active Debug Code	888	1171
HasMember	V	491	Public cloneable() Method Without Final ('Object Hijack')	888	1174
HasMember	V	493	Critical Public Variable Without Final Modifier	888	1182
HasMember	V	500	Public Static Field Not Marked Final	888	1200
HasMember	V	531	Inclusion of Sensitive Information in Test Code	888	1240
HasMember	V	568	finalize() Method Without super.finalize()	888	1290
HasMember	V	580	clone() Method Without super.clone()	888	1311
HasMember	V	582	Array Declared Public, Final, and Static	888	1314
HasMember	V	583	finalize() Method Declared Public	888	1315
HasMember	V	608	Struts: Non-private Field in ActionForm Class	888	1361
HasMember	B	766	Critical Data Element Declared Public	888	1607

Category-1005: 7PK - Input Validation and Representation

Category ID : 1005

Summary

This category represents one of the phyla in the Seven Pernicious Kingdoms vulnerability classification. It includes weaknesses that exist when an application does not properly validate or represent input. According to the authors of the Seven Pernicious Kingdoms, "Input validation and representation problems are caused by metacharacters, alternate encodings and numeric representations. Security problems result from trusting input."

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	700	Seven Pernicious Kingdoms	700	2557
HasMember	C	20	Improper Input Validation	700	20
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	700	145
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	700	163
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	700	201
HasMember	C	99	Improper Control of Resource Identifiers ('Resource Injection')	700	243

References

[REF-6]Katrina Tsipenyuk, Brian Chess and Gary McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". NIST Workshop on Software Security Assurance Tools Techniques and Metrics. 2005 November 7. NIST. < https://samate.nist.gov/SSATTM_Content/papers/Seven%20Pernicious%20Kingdoms%20-%20Taxonomy%20of%20Sw%20Security%20Errors%20-%20Tsipenyuk%20-%20Chess%20-%20McGraw.pdf >.

Category-1006: Bad Coding Practices

Category ID : 1006

Summary

Weaknesses in this category are related to coding practices that are deemed unsafe and increase the chances that an exploitable vulnerability will be present in the application. These weaknesses do not directly introduce a vulnerability, but indicate that the product has not been carefully developed or maintained. If a program is complex, difficult to maintain, not portable, or shows evidence of neglect, then there is a higher likelihood that weaknesses are buried in the code.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	358	Improperly Implemented Security Check for Standard	699	881
HasMember	B	360	Trust of System Event Data	699	887
HasMember	B	478	Missing Default Case in Multiple Condition Expression	699	1142
HasMember	B	487	Reliance on Package-level Scope	699	1167
HasMember	B	489	Active Debug Code	699	1171
HasMember	B	547	Use of Hard-coded, Security-relevant Constants	699	1259
HasMember	B	561	Dead Code	699	1275
HasMember	B	562	Return of Stack Variable Address	699	1278
HasMember	B	563	Assignment to Variable without Use	699	1280
HasMember	V	581	Object Model Violation: Just One of Equals and Hashcode Defined	699	1312
HasMember	B	586	Explicit Call to Finalize()	699	1320
HasMember	V	605	Multiple Binds to the Same Port	699	1356
HasMember	B	628	Function Call with Incorrectly Specified Arguments	699	1398
HasMember	B	654	Reliance on a Single Factor in a Security Decision	699	1439
HasMember	C	656	Reliance on Security Through Obscurity	699	1444
HasMember	B	694	Use of Multiple Resources with Duplicate Identifier	699	1523
HasMember	B	807	Reliance on Untrusted Inputs in a Security Decision	699	1714
HasMember	B	1041	Use of Redundant Code	699	1875
HasMember	B	1043	Data Element Aggregating an Excessively Large Number of Non-Primitive Elements	699	1877
HasMember	B	1044	Architecture with Number of Horizontal Layers Outside of Expected Range	699	1879
HasMember	B	1045	Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor	699	1880
HasMember	B	1046	Creation of Immutable Text Using String Concatenation	699	1881
HasMember	B	1048	Invokable Control Element with Large Number of Outward Calls	699	1883
HasMember	B	1049	Excessive Data Query Operations in a Large Data Table	699	1884

Nature	Type	ID	Name	V	Page
HasMember	B	1050	Excessive Platform Resource Consumption within a Loop	699	1885
HasMember	B	1063	Creation of Class Instance within a Static Code Block	699	1901
HasMember	B	1065	Runtime Resource Management Control Element in a Component Built to Run on Application Servers	699	1903
HasMember	B	1066	Missing Serialization Control Element	699	1904
HasMember	B	1067	Excessive Execution of Sequential Searches of Data Resource	699	1905
HasMember	B	1070	Serializable Data Element Containing non-Serializable Item Elements	699	1909
HasMember	B	1071	Empty Code Block	699	1910
HasMember	B	1072	Data Resource Access without Use of Connection Pooling	699	1912
HasMember	B	1073	Non-SQL Invokable Control Element with Excessive Number of Data Resource Accesses	699	1913
HasMember	B	1079	Parent Class without Virtual Destructor Method	699	1919
HasMember	B	1082	Class Instance Self Destruction Control Element	699	1921
HasMember	B	1084	Invokable Control Element with Excessive File or Data Access Operations	699	1924
HasMember	B	1085	Invokable Control Element with Excessive Volume of Commented-out Code	699	1925
HasMember	B	1087	Class with Virtual Method without a Virtual Destructor	699	1927
HasMember	B	1089	Large Data Table with Excessive Number of Indices	699	1929
HasMember	B	1092	Use of Same Invokable Control Element in Multiple Architectural Layers	699	1932
HasMember	B	1094	Excessive Index Range Scan for a Data Resource	699	1934
HasMember	B	1097	Persistent Storable Data Element without Associated Comparison Control Element	699	1937
HasMember	B	1098	Data Element containing Pointer Item without Proper Copy Control Element	699	1938
HasMember	B	1099	Inconsistent Naming Conventions for Identifiers	699	1939
HasMember	B	1101	Reliance on Runtime Component in Generated Code	699	1941
HasMember	B	1102	Reliance on Machine-Dependent Data Representation	699	1942
HasMember	B	1103	Use of Platform-Dependent Third Party Components	699	1943
HasMember	B	1104	Use of Unmaintained Third Party Components	699	1944
HasMember	B	1106	Insufficient Use of Symbolic Constants	699	1946
HasMember	B	1107	Insufficient Isolation of Symbolic Constant Definitions	699	1947
HasMember	B	1108	Excessive Reliance on Global Variables	699	1948
HasMember	B	1109	Use of Same Variable for Multiple Purposes	699	1949
HasMember	B	1113	Inappropriate Comment Style	699	1953
HasMember	B	1114	Inappropriate Whitespace Style	699	1953
HasMember	B	1115	Source Code Element without Standard Prologue	699	1954
HasMember	B	1116	Inaccurate Comments	699	1955
HasMember	B	1117	Callable with Insufficient Behavioral Summary	699	1957
HasMember	B	1126	Declaration of Variable with Unnecessarily Wide Scope	699	1966
HasMember	B	1127	Compilation with Insufficient Warnings or Errors	699	1966
HasMember	B	1235	Incorrect Use of Autoboxing and Unboxing for Performance Critical Operations	699	2017

Category-1009: Audit

Category ID : 1009

Summary

Weaknesses in this category are related to the design and architecture of audit-based components of the system. Frequently these deal with logging user activities in order to identify attackers and modifications to the system. The weaknesses in this category could lead to a degradation of the quality of the audit capability if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	B	117	Improper Output Neutralization for Logs	1008	288
HasMember	B	223	Omission of Security-relevant Information	1008	559
HasMember	B	224	Obscured Security-relevant Information by Alternate Name	1008	561
HasMember	B	532	Insertion of Sensitive Information into Log File	1008	1241
HasMember	B	778	Insufficient Logging	1008	1638
HasMember	B	779	Logging of Excessive Data	1008	1642

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1010: Authenticate Actors

Category ID : 1010

Summary

Weaknesses in this category are related to the design and architecture of authentication components of the system. Frequently these deal with verifying the entity is indeed who it claims to be. The weaknesses in this category could lead to a degradation of the quality of authentication if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	V	258	Empty Password in Configuration File	1008	621
HasMember	V	259	Use of Hard-coded Password	1008	623
HasMember	B	262	Not Using Password Aging	1008	633
HasMember	B	263	Password Aging with Long Expiration	1008	636
HasMember	C	287	Improper Authentication	1008	692
HasMember	B	288	Authentication Bypass Using an Alternate Path or Channel	1008	700

Nature	Type	ID	Name	V	Page
HasMember	B	289	Authentication Bypass by Alternate Name	1008	703
HasMember	B	290	Authentication Bypass by Spoofing	1008	705
HasMember	V	291	Reliance on IP Address for Authentication	1008	708
HasMember	V	293	Using Referer Field for Authentication	1008	710
HasMember	B	294	Authentication Bypass by Capture-replay	1008	712
HasMember	B	301	Reflection Attack in an Authentication Protocol	1008	733
HasMember	B	302	Authentication Bypass by Assumed-Immutable Data	1008	735
HasMember	B	303	Incorrect Implementation of Authentication Algorithm	1008	737
HasMember	B	304	Missing Critical Step in Authentication	1008	738
HasMember	B	305	Authentication Bypass by Primary Weakness	1008	740
HasMember	B	306	Missing Authentication for Critical Function	1008	741
HasMember	B	307	Improper Restriction of Excessive Authentication Attempts	1008	747
HasMember	B	308	Use of Single-factor Authentication	1008	752
HasMember	B	322	Key Exchange without Entity Authentication	1008	788
HasMember	B	521	Weak Password Requirements	1008	1223
HasMember	V	593	Authentication Bypass: OpenSSL CTX Object Modified after SSL Objects are Created	1008	1331
HasMember	B	603	Use of Client-Side Authentication	1008	1354
HasMember	B	620	Unverified Password Change	1008	1383
HasMember	B	640	Weak Password Recovery Mechanism for Forgotten Password	1008	1409
HasMember	B	798	Use of Hard-coded Credentials	1008	1690
HasMember	B	836	Use of Password Hash Instead of Password for Authentication	1008	1761
HasMember	B	916	Use of Password Hash With Insufficient Computational Effort	1008	1813

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1011: Authorize Actors

Category ID : 1011

Summary

Weaknesses in this category are related to the design and architecture of a system's authorization components. Frequently these deal with enforcing that agents have the required permissions before performing certain operations, such as modifying data. The weaknesses in this category could lead to a degradation of quality of the authorization capability if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	B	15	External Control of System or Configuration Setting	1008	17
HasMember	C	114	Process Control	1008	277
HasMember	V	219	Storage of File with Sensitive Data Under Web Root	1008	553
HasMember	V	220	Storage of File With Sensitive Data Under FTP Root	1008	555
HasMember	B	266	Incorrect Privilege Assignment	1008	638
HasMember	B	267	Privilege Defined With Unsafe Actions	1008	641
HasMember	B	268	Privilege Chaining	1008	644
HasMember	C	269	Improper Privilege Management	1008	646
HasMember	B	270	Privilege Context Switching Error	1008	651
HasMember	C	271	Privilege Dropping / Lowering Errors	1008	653
HasMember	B	272	Least Privilege Violation	1008	656
HasMember	B	273	Improper Check for Dropped Privileges	1008	660
HasMember	B	274	Improper Handling of Insufficient Privileges	1008	663
HasMember	B	276	Incorrect Default Permissions	1008	665
HasMember	V	277	Insecure Inherited Permissions	1008	668
HasMember	V	279	Incorrect Execution-Assigned Permissions	1008	671
HasMember	B	280	Improper Handling of Insufficient Permissions or Privileges	1008	672
HasMember	B	281	Improper Preservation of Permissions	1008	674
HasMember	C	282	Improper Ownership Management	1008	676
HasMember	B	283	Unverified Ownership	1008	678
HasMember	P	284	Improper Access Control	1008	680
HasMember	C	285	Improper Authorization	1008	684
HasMember	C	286	Incorrect User Management	1008	691
HasMember	C	300	Channel Accessible by Non-Endpoint	1008	730
HasMember	B	341	Predictable from Observable State	1008	843
HasMember	B	359	Exposure of Private Personal Information to an Unauthorized Actor	1008	882
HasMember	B	403	Exposure of File Descriptor to Unintended Control Sphere ('File Descriptor Leak')	1008	978
HasMember	B	419	Unprotected Primary Channel	1008	1017
HasMember	B	420	Unprotected Alternate Channel	1008	1018
HasMember	B	425	Direct Request ('Forced Browsing')	1008	1025
HasMember	B	426	Untrusted Search Path	1008	1028
HasMember	B	434	Unrestricted Upload of File with Dangerous Type	1008	1048
HasMember	V	527	Exposure of Version-Control Repository to an Unauthorized Control Sphere	1008	1236
HasMember	V	528	Exposure of Core Dump File to an Unauthorized Control Sphere	1008	1237
HasMember	V	529	Exposure of Access Control List Files to an Unauthorized Control Sphere	1008	1238
HasMember	V	530	Exposure of Backup File to an Unauthorized Control Sphere	1008	1239
HasMember	B	538	Insertion of Sensitive Information into Externally-Accessible File or Directory	1008	1248
HasMember	B	551	Incorrect Behavior Order: Authorization Before Parsing and Canonicalization	1008	1264
HasMember	B	552	Files or Directories Accessible to External Parties	1008	1265

Nature	Type	ID	Name	V	Page
HasMember	V	566	Authorization Bypass Through User-Controlled SQL Primary Key	1008	1286
HasMember	B	639	Authorization Bypass Through User-Controlled Key	1008	1406
HasMember	C	642	External Control of Critical State Data	1008	1414
HasMember	V	647	Use of Non-Canonical URL Paths for Authorization Decisions	1008	1426
HasMember	C	653	Improper Isolation or Compartmentalization	1008	1437
HasMember	C	656	Reliance on Security Through Obscurity	1008	1444
HasMember	C	668	Exposure of Resource to Wrong Sphere	1008	1469
HasMember	C	669	Incorrect Resource Transfer Between Spheres	1008	1471
HasMember	C	671	Lack of Administrator Control over Security	1008	1478
HasMember	C	673	External Influence of Sphere Definition	1008	1483
HasMember	B	708	Incorrect Ownership Assignment	1008	1548
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	1008	1551
HasMember	B	770	Allocation of Resources Without Limits or Throttling	1008	1613
HasMember	V	782	Exposed IOCTL with Insufficient Access Control	1008	1648
HasMember	V	827	Improper Control of Document Type Definition	1008	1736
HasMember	C	862	Missing Authorization	1008	1780
HasMember	C	863	Incorrect Authorization	1008	1787
HasMember	B	921	Storage of Sensitive Data in a Mechanism without Access Control	1008	1824
HasMember	C	923	Improper Restriction of Communication Channel to Intended Endpoints	1008	1827
HasMember	B	939	Improper Authorization in Handler for Custom URL Scheme	1008	1840
HasMember	V	942	Permissive Cross-domain Policy with Untrusted Domains	1008	1847

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1012: Cross Cutting

Category ID : 1012

Summary

Weaknesses in this category are related to the design and architecture of multiple security tactics and how they affect a system. For example, information exposure can impact the Limit Access and Limit Exposure security tactics. The weaknesses in this category could lead to a degradation of the quality of many capabilities if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	B	208	Observable Timing Discrepancy	1008	529
HasMember	B	392	Missing Report of Error Condition	1008	951
HasMember	B	460	Improper Cleanup on Thrown Exception	1008	1102
HasMember	B	544	Missing Standardized Error Handling Mechanism	1008	1256
HasMember	C	602	Client-Side Enforcement of Server-Side Security	1008	1350
HasMember	P	703	Improper Check or Handling of Exceptional Conditions	1008	1535
HasMember	C	754	Improper Check for Unusual or Exceptional Conditions	1008	1568
HasMember	V	784	Reliance on Cookies without Validation and Integrity Checking in a Security Decision	1008	1653
HasMember	B	807	Reliance on Untrusted Inputs in a Security Decision	1008	1714

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1013: Encrypt Data

Category ID : 1013

Summary

Weaknesses in this category are related to the design and architecture of data confidentiality in a system. Frequently these deal with the use of encryption libraries. The weaknesses in this category could lead to a degradation of the quality data encryption if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	B	256	Plaintext Storage of a Password	1008	615
HasMember	B	257	Storing Passwords in a Recoverable Format	1008	618
HasMember	B	260	Password in Configuration File	1008	629
HasMember	B	261	Weak Encoding for Password	1008	631
HasMember	C	311	Missing Encryption of Sensitive Data	1008	757
HasMember	B	312	Cleartext Storage of Sensitive Information	1008	764
HasMember	V	313	Cleartext Storage in a File or on Disk	1008	770
HasMember	V	314	Cleartext Storage in the Registry	1008	772
HasMember	V	315	Cleartext Storage of Sensitive Information in a Cookie	1008	774
HasMember	V	316	Cleartext Storage of Sensitive Information in Memory	1008	775
HasMember	V	317	Cleartext Storage of Sensitive Information in GUI	1008	777
HasMember	V	318	Cleartext Storage of Sensitive Information in Executable	1008	778
HasMember	B	319	Cleartext Transmission of Sensitive Information	1008	779
HasMember	V	321	Use of Hard-coded Cryptographic Key	1008	785
HasMember	B	323	Reusing a Nonce, Key Pair in Encryption	1008	790

Nature	Type	ID	Name	V	Page
HasMember	B	324	Use of a Key Past its Expiration Date	1008	792
HasMember	B	325	Missing Cryptographic Step	1008	794
HasMember	C	326	Inadequate Encryption Strength	1008	796
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	1008	799
HasMember	B	328	Use of Weak Hash	1008	806
HasMember	C	330	Use of Insufficiently Random Values	1008	814
HasMember	B	331	Insufficient Entropy	1008	821
HasMember	V	332	Insufficient Entropy in PRNG	1008	823
HasMember	V	333	Improper Handling of Insufficient Entropy in TRNG	1008	825
HasMember	B	334	Small Space of Random Values	1008	827
HasMember	B	335	Incorrect Usage of Seeds in Pseudo-Random Number Generator (PRNG)	1008	829
HasMember	V	336	Same Seed in Pseudo-Random Number Generator (PRNG)	1008	832
HasMember	V	337	Predictable Seed in Pseudo-Random Number Generator (PRNG)	1008	834
HasMember	B	338	Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)	1008	837
HasMember	V	339	Small Seed Space in PRNG	1008	840
HasMember	B	347	Improper Verification of Cryptographic Signature	1008	857
HasMember	C	522	Insufficiently Protected Credentials	1008	1225
HasMember	B	523	Unprotected Transport of Credentials	1008	1230
HasMember	B	757	Selection of Less-Secure Algorithm During Negotiation ('Algorithm Downgrade')	1008	1581
HasMember	V	759	Use of a One-Way Hash without a Salt	1008	1585
HasMember	V	760	Use of a One-Way Hash with a Predictable Salt	1008	1589
HasMember	V	780	Use of RSA Algorithm without OAEP	1008	1644
HasMember	C	922	Insecure Storage of Sensitive Information	1008	1825

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1014: Identify Actors

Category ID : 1014

Summary

Weaknesses in this category are related to the design and architecture of a system's identification management components. Frequently these deal with verifying that external agents provide inputs into the system. The weaknesses in this category could lead to a degradation of the quality of identification management if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	B	295	Improper Certificate Validation	1008	714
HasMember	B	296	Improper Following of a Certificate's Chain of Trust	1008	719
HasMember	V	297	Improper Validation of Certificate with Host Mismatch	1008	722
HasMember	V	298	Improper Validation of Certificate Expiration	1008	726
HasMember	B	299	Improper Check for Certificate Revocation	1008	727
HasMember	C	345	Insufficient Verification of Data Authenticity	1008	851
HasMember	C	346	Origin Validation Error	1008	853
HasMember	V	370	Missing Check for Certificate Revocation after Initial Check	1008	917
HasMember	C	441	Unintended Proxy or Intermediary ('Confused Deputy')	1008	1064
HasMember	V	599	Missing Validation of OpenSSL Certificate	1008	1341
HasMember	B	940	Improper Verification of Source of a Communication Channel	1008	1842
HasMember	B	941	Incorrectly Specified Destination in a Communication Channel	1008	1845

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1015: Limit Access

Category ID : 1015

Summary

Weaknesses in this category are related to the design and architecture of system resources. Frequently these deal with restricting the amount of resources that are accessed by actors, such as memory, network connections, CPU or access points. The weaknesses in this category could lead to a degradation of the quality of authentication if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	B	73	External Control of File Name or Path	1008	132
HasMember	B	201	Insertion of Sensitive Information Into Sent Data	1008	514
HasMember	B	209	Generation of Error Message Containing Sensitive Information	1008	533
HasMember	B	212	Improper Removal of Sensitive Information Before Storage or Transfer	1008	544
HasMember	V	243	Creation of chroot Jail Without Changing Working Directory	1008	589
HasMember	B	250	Execution with Unnecessary Privileges	1008	599

Nature	Type	ID	Name	V	Page
HasMember	C	610	Externally Controlled Reference to a Resource in Another Sphere	1008	1364
HasMember	B	611	Improper Restriction of XML External Entity Reference	1008	1367

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1016: Limit Exposure

Category ID : 1016

Summary

Weaknesses in this category are related to the design and architecture of the entry points to a system. Frequently these deal with minimizing the attack surface through designing the system with the least needed amount of entry points. The weaknesses in this category could lead to a degradation of a system's defenses if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	B	210	Self-generated Error Message Containing Sensitive Information	1008	539
HasMember	B	211	Externally-Generated Error Message Containing Sensitive Information	1008	541
HasMember	B	214	Invocation of Process Using Visible Sensitive Information	1008	549
HasMember	V	550	Server-generated Error Message Containing Sensitive Information	1008	1263
HasMember	B	829	Inclusion of Functionality from Untrusted Control Sphere	1008	1741
HasMember	V	830	Inclusion of Web Functionality from an Untrusted Source	1008	1747

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1017: Lock Computer

Category ID : 1017

Summary

Weaknesses in this category are related to the design and architecture of a system's lockout mechanism. Frequently these deal with scenarios that take effect in case of multiple failed attempts to access a given resource. The weaknesses in this category could lead to a degradation of access to system assets if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	B	645	Overly Restrictive Account Lockout Mechanism	1008	1423

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1018: Manage User Sessions

Category ID : 1018

Summary

Weaknesses in this category are related to the design and architecture of session management. Frequently these deal with the information or status about each user and their access rights for the duration of multiple requests. The weaknesses in this category could lead to a degradation of the quality of session management if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	V	6	J2EE Misconfiguration: Insufficient Session-ID Length	1008	2
HasMember	C	384	Session Fixation	1008	936
HasMember	B	488	Exposure of Data Element to Wrong Session	1008	1169
HasMember	V	579	J2EE Bad Practices: Non-serializable Object Stored in Session	1008	1309
HasMember	B	613	Insufficient Session Expiration	1008	1371
HasMember	B	841	Improper Enforcement of Behavioral Workflow	1008	1772

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of

Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1019: Validate Inputs

Category ID : 1019

Summary

Weaknesses in this category are related to the design and architecture of a system's input validation components. Frequently these deal with sanitizing, neutralizing and validating any externally provided inputs to minimize malformed data from entering the system and preventing code injection in the input data. The weaknesses in this category could lead to a degradation of the quality of data flow in a system if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	C	20	Improper Input Validation	1008	20
HasMember	B	59	Improper Link Resolution Before File Access ('Link Following')	1008	111
HasMember	C	74	Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')	1008	137
HasMember	C	75	Failure to Sanitize Special Elements into a Different Plane (Special Element Injection)	1008	142
HasMember	B	76	Improper Neutralization of Equivalent Special Elements	1008	144
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	1008	145
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	1008	151
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	1008	163
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	1008	194
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	1008	201
HasMember	B	90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	1008	212
HasMember	B	91	XML Injection (aka Blind XPath Injection)	1008	215
HasMember	B	93	Improper Neutralization of CRLF Sequences ('CRLF Injection')	1008	217
HasMember	B	94	Improper Control of Generation of Code ('Code Injection')	1008	219
HasMember	V	95	Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')	1008	226
HasMember	B	96	Improper Neutralization of Directives in Statically Saved Code ('Static Code Injection')	1008	232
HasMember	V	97	Improper Neutralization of Server-Side Includes (SSI) Within a Web Page	1008	235
HasMember	V	98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')	1008	236

Nature	Type	ID	Name	V	Page
HasMember	C	99	Improper Control of Resource Identifiers ('Resource Injection')	1008	243
HasMember	C	138	Improper Neutralization of Special Elements	1008	373
HasMember	V	150	Improper Neutralization of Escape, Meta, or Control Sequences	1008	394
HasMember	B	349	Acceptance of Extraneous Untrusted Data With Trusted Data	1008	861
HasMember		352	Cross-Site Request Forgery (CSRF)	1008	868
HasMember	B	472	External Control of Assumed-Immutable Web Parameter	1008	1123
HasMember	V	473	PHP External Variable Modification	1008	1127
HasMember	B	502	Deserialization of Untrusted Data	1008	1204
HasMember	B	601	URL Redirection to Untrusted Site ('Open Redirect')	1008	1345
HasMember	B	641	Improper Restriction of Names for Files and Other Resources	1008	1412
HasMember	B	643	Improper Neutralization of Data within XPath Expressions ('XPath Injection')	1008	1419
HasMember	B	652	Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')	1008	1435
HasMember	C	790	Improper Filtering of Special Elements	1008	1678
HasMember	B	791	Incomplete Filtering of Special Elements	1008	1680
HasMember	V	792	Incomplete Filtering of One or More Instances of Special Elements	1008	1681
HasMember	V	793	Only Filtering One Instance of a Special Element	1008	1683
HasMember	V	794	Incomplete Filtering of Multiple Instances of Special Elements	1008	1684
HasMember	B	795	Only Filtering Special Elements at a Specified Location	1008	1685
HasMember	V	796	Only Filtering Special Elements Relative to a Marker	1008	1687
HasMember	V	797	Only Filtering Special Elements at an Absolute Position	1008	1689
HasMember	C	943	Improper Neutralization of Special Elements in Data Query Logic	1008	1850

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1020: Verify Message Integrity

Category ID : 1020

Summary

Weaknesses in this category are related to the design and architecture of a system's data integrity components. Frequently these deal with ensuring integrity of data, such as messages, resource files, deployment files, and configuration files. The weaknesses in this category could lead to a

degradation of data integrity quality if they are not addressed when designing or implementing a secure architecture.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1008	Architectural Concepts	1008	2577
HasMember	B	353	Missing Support for Integrity Check	1008	874
HasMember	B	354	Improper Validation of Integrity Check Value	1008	876
HasMember	B	390	Detection of Error Condition Without Action	1008	943
HasMember	B	391	Unchecked Error Condition	1008	948
HasMember	B	494	Download of Code Without Integrity Check	1008	1185
HasMember	B	565	Reliance on Cookies without Validation and Integrity Checking	1008	1283
HasMember	B	649	Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking	1008	1430
HasMember	P	707	Improper Neutralization	1008	1546
HasMember	C	755	Improper Handling of Exceptional Conditions	1008	1576
HasMember	B	924	Improper Enforcement of Message Integrity During Transmission in a Communication Channel	1008	1830

References

[REF-9]Santos, J. C. S., Tarrit, K. and Mirakhorli, M.. "A Catalog of Security Architecture Weaknesses.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/cawe-paper.pdf >.

[REF-10]Santos, J. C. S., Peruma, A., Mirakhorli, M., Galster, M. and Sejfia, A.. "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, PHP and Thunderbird.". 2017 IEEE International Conference on Software Architecture (ICSA). 2017. < https://design.se.rit.edu/papers/TacticalVulnerabilities.pdf >.

Category-1027: OWASP Top Ten 2017 Category A1 - Injection

Category ID : 1027

Summary

Weaknesses in this category are related to the A1 category in the OWASP Top Ten 2017.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1026	Weaknesses in OWASP Top Ten (2017)	1026	2578
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	1026	145
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	1026	151
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	1026	194
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	1026	201
HasMember	B	90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	1026	212
HasMember	B	91	XML Injection (aka Blind XPath Injection)	1026	215
HasMember	V	564	SQL Injection: Hibernate	1026	1282

Nature	Type	ID	Name	V	Page
HasMember	B	917	Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')	1026	1818
HasMember	C	943	Improper Neutralization of Special Elements in Data Query Logic	1026	1850

References

[REF-957]"Top 10 2017". 2017 April 2. OWASP. < https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf >.

Category-1028: OWASP Top Ten 2017 Category A2 - Broken Authentication

Category ID : 1028

Summary

Weaknesses in this category are related to the A2 category in the OWASP Top Ten 2017.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1026	Weaknesses in OWASP Top Ten (2017)	1026	2578
HasMember	B	256	Plaintext Storage of a Password	1026	615
HasMember	C	287	Improper Authentication	1026	692
HasMember	B	308	Use of Single-factor Authentication	1026	752
HasMember		384	Session Fixation	1026	936
HasMember	C	522	Insufficiently Protected Credentials	1026	1225
HasMember	B	523	Unprotected Transport of Credentials	1026	1230
HasMember	B	613	Insufficient Session Expiration	1026	1371
HasMember	B	620	Unverified Password Change	1026	1383
HasMember	B	640	Weak Password Recovery Mechanism for Forgotten Password	1026	1409

References

[REF-957]"Top 10 2017". 2017 April 2. OWASP. < https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf >.

Category-1029: OWASP Top Ten 2017 Category A3 - Sensitive Data Exposure

Category ID : 1029

Summary

Weaknesses in this category are related to the A3 category in the OWASP Top Ten 2017.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1026	Weaknesses in OWASP Top Ten (2017)	1026	2578
HasMember	V	220	Storage of File With Sensitive Data Under FTP Root	1026	555
HasMember	B	295	Improper Certificate Validation	1026	714
HasMember	C	311	Missing Encryption of Sensitive Data	1026	757
HasMember	B	312	Cleartext Storage of Sensitive Information	1026	764

Nature	Type	ID	Name	V	Page
HasMember	B	319	Cleartext Transmission of Sensitive Information	1026	779
HasMember	C	320	Key Management Errors	1026	2319
HasMember	B	325	Missing Cryptographic Step	1026	794
HasMember	C	326	Inadequate Encryption Strength	1026	796
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	1026	799
HasMember	B	328	Use of Weak Hash	1026	806
HasMember	B	359	Exposure of Private Personal Information to an Unauthorized Actor	1026	882

References

[REF-957]"Top 10 2017". 2017 April 2. OWASP. < https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf >.

Category-1030: OWASP Top Ten 2017 Category A4 - XML External Entities (XXE)

Category ID : 1030

Summary

Weaknesses in this category are related to the A4 category in the OWASP Top Ten 2017.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1026	Weaknesses in OWASP Top Ten (2017)	1026	2578
HasMember	B	611	Improper Restriction of XML External Entity Reference	1026	1367
HasMember	B	776	Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')	1026	1633

References

[REF-957]"Top 10 2017". 2017 April 2. OWASP. < https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf >.

Category-1031: OWASP Top Ten 2017 Category A5 - Broken Access Control

Category ID : 1031

Summary

Weaknesses in this category are related to the A5 category in the OWASP Top Ten 2017.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1026	Weaknesses in OWASP Top Ten (2017)	1026	2578
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	1026	33
HasMember	P	284	Improper Access Control	1026	680
HasMember	C	285	Improper Authorization	1026	684
HasMember	B	425	Direct Request ('Forced Browsing')	1026	1025
HasMember	B	639	Authorization Bypass Through User-Controlled Key	1026	1406

References

[REF-957]"Top 10 2017". 2017 April 2. OWASP. < https://owasp.org/www-pdf-archive/
OWASP_Top_10-2017_%28en%29.pdf >.

Category-1032: OWASP Top Ten 2017 Category A6 - Security Misconfiguration

Category ID : 1032

Summary

Weaknesses in this category are related to the A6 category in the OWASP Top Ten 2017.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1026	Weaknesses in OWASP Top Ten (2017)	1026	2578
MemberOf	C	1349	OWASP Top Ten 2021 Category A05:2021 - Security Misconfiguration	1344	2493
HasMember	C	16	Configuration	1026	2309
HasMember	B	209	Generation of Error Message Containing Sensitive Information	1026	533
HasMember	V	548	Exposure of Information Through Directory Listing	1026	1261

Notes

Relationship

While the OWASP document maps to CWE-2 and CWE-388, these are not appropriate for mapping, as they are high-level categories that are only intended for the Seven Pernicious Kingdoms view (CWE-700).

References

[REF-957]"Top 10 2017". 2017 April 2. OWASP. < https://owasp.org/www-pdf-archive/
OWASP_Top_10-2017_%28en%29.pdf >.

Category-1033: OWASP Top Ten 2017 Category A7 - Cross-Site Scripting (XSS)

Category ID : 1033

Summary

Weaknesses in this category are related to the A7 category in the OWASP Top Ten 2017.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1026	Weaknesses in OWASP Top Ten (2017)	1026	2578
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	1026	163

References

[REF-957]"Top 10 2017". 2017 April 2. OWASP. < https://owasp.org/www-pdf-archive/
OWASP_Top_10-2017_%28en%29.pdf >.

Category-1034: OWASP Top Ten 2017 Category A8 - Insecure Deserialization

Category ID : 1034

Summary

Weaknesses in this category are related to the A8 category in the OWASP Top Ten 2017.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1026	Weaknesses in OWASP Top Ten (2017)	1026	2578
HasMember	B	502	Deserialization of Untrusted Data	1026	1204

References

[REF-957]"Top 10 2017". 2017 April 2. OWASP. < https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf >.

Category-1035: OWASP Top Ten 2017 Category A9 - Using Components with Known Vulnerabilities

Category ID : 1035

Summary

Weaknesses in this category are related to the A9 category in the OWASP Top Ten 2017.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1026	Weaknesses in OWASP Top Ten (2017)	1026	2578
MemberOf	C	1352	OWASP Top Ten 2021 Category A06:2021 - Vulnerable and Outdated Components	1344	2494

Notes

Relationship

This is an unusual category. CWE does not cover the limitations of human processes and procedures that cannot be described in terms of a specific technical weakness as resident in the code, architecture, or configuration of the software. Since "known vulnerabilities" can arise from any kind of weakness, it is not possible to map this OWASP category to other CWE entries, since it would effectively require mapping this category to ALL weaknesses.

References

[REF-957]"Top 10 2017". 2017 April 2. OWASP. < https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf >.

Category-1036: OWASP Top Ten 2017 Category A10 - Insufficient Logging & Monitoring

Category ID : 1036

Summary

Weaknesses in this category are related to the A10 category in the OWASP Top Ten 2017.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1026	Weaknesses in OWASP Top Ten (2017)	1026	2578
HasMember	B	223	Omission of Security-relevant Information	1026	559
HasMember	B	778	Insufficient Logging	1026	1638

References

[REF-957]"Top 10 2017". 2017 April 2. OWASP. < https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf >.

Category-1129: CISQ Quality Measures (2016) - Reliability

Category ID : 1129

Summary

Weaknesses in this category are related to the CISQ Quality Measures for Reliability, as documented in 2016 with the Automated Source Code CISQ Reliability Measure (ASCRM) Specification 1.0. Presence of these weaknesses could reduce the reliability of the software.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1128	CISQ Quality Measures (2016)	1128	2581
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	1128	304
HasMember	B	252	Unchecked Return Value	1128	606
HasMember	B	396	Declaration of Catch for Generic Exception	1128	959
HasMember	B	397	Declaration of Throws for Generic Exception	1128	961
HasMember	V	456	Missing Initialization of a Variable	1128	1089
HasMember	C	674	Uncontrolled Recursion	1128	1484
HasMember	C	704	Incorrect Type Conversion or Cast	1128	1538
HasMember	B	772	Missing Release of Resource after Effective Lifetime	1128	1624
HasMember	B	788	Access of Memory Location After End of Buffer	1128	1669
HasMember	B	1045	Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor	1128	1880
HasMember	B	1047	Modules with Circular Dependencies	1128	1882
HasMember	B	1051	Initialization with Hard-Coded Network Resource Configuration Data	1128	1886
HasMember	B	1056	Invokable Control Element with Variadic Parameters	1128	1891
HasMember	B	1058	Invokable Control Element in Multi-Thread Context with non-Final Static Storable or Member Element	1128	1893
HasMember	B	1062	Parent Class with References to Child Class	1128	1900
HasMember	B	1065	Runtime Resource Management Control Element in a Component Built to Run on Application Servers	1128	1903
HasMember	B	1066	Missing Serialization Control Element	1128	1904
HasMember	V	1069	Empty Exception Block	1128	1907
HasMember	B	1070	Serializable Data Element Containing non-Serializable Item Elements	1128	1909
HasMember	V	1077	Floating Point Comparison with Incorrect Operator	1128	1917
HasMember	B	1079	Parent Class without Virtual Destructor Method	1128	1919
HasMember	B	1082	Class Instance Self Destruction Control Element	1128	1921
HasMember	B	1083	Data Access from Outside Expected Data Manager Component	1128	1922

Nature	Type	ID	Name	V	Page
HasMember	B	1087	Class with Virtual Method without a Virtual Destructor	1128	1927
HasMember	B	1088	Synchronous Access of Remote Resource without Timeout	1128	1928
HasMember	V	1096	Singleton Class Instance Creation without Proper Locking or Synchronization	1128	1936
HasMember	B	1097	Persistent Storable Data Element without Associated Comparison Control Element	1128	1937
HasMember	B	1098	Data Element containing Pointer Item without Proper Copy Control Element	1128	1938

References

[REF-961]Object Management Group (OMG). "Automated Source Code Reliability Measure (ASCRM)". 2016 January. < http://www.omg.org/spec/ASCRM/1.0/ >.

[REF-968]Consortium for Information & Software Quality (CISQ). "Automated Quality Characteristic Measures". 2016. < http://it-cisq.org/standards/automated-quality-characteristic-measures/ >.

Category-1130: CISQ Quality Measures (2016) - Maintainability

Category ID : 1130

Summary

Weaknesses in this category are related to the CISQ Quality Measures for Maintainability, as documented in 2016 with the Automated Source Code Maintainability Measure (ASCMM) Specification 1.0. Presence of these weaknesses could reduce the maintainability of the software.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1128	CISQ Quality Measures (2016)	1128	2581
HasMember	B	561	Dead Code	1128	1275
HasMember	B	766	Critical Data Element Declared Public	1128	1607
HasMember	B	1041	Use of Redundant Code	1128	1875
HasMember	B	1044	Architecture with Number of Horizontal Layers Outside of Expected Range	1128	1879
HasMember	B	1047	Modules with Circular Dependencies	1128	1882
HasMember	B	1048	Invokable Control Element with Large Number of Outward Calls	1128	1883
HasMember	B	1052	Excessive Use of Hard-Coded Literals in Initialization	1128	1887
HasMember	B	1054	Invocation of a Control Element at an Unnecessarily Deep Horizontal Layer	1128	1889
HasMember	B	1055	Multiple Inheritance from Concrete Classes	1128	1890
HasMember	B	1064	Invokable Control Element with Signature Containing an Excessive Number of Parameters	1128	1902
HasMember	B	1074	Class with Excessively Deep Inheritance	1128	1914
HasMember	B	1075	Unconditional Control Flow Transfer outside of Switch Block	1128	1915
HasMember	B	1080	Source Code File with Excessive Number of Lines of Code	1128	1920
HasMember	B	1084	Invokable Control Element with Excessive File or Data Access Operations	1128	1924

Nature	Type	ID	Name	V	Page
HasMember	B	1085	Invokable Control Element with Excessive Volume of Commented-out Code	1128	1925
HasMember	B	1086	Class with Excessive Number of Child Classes	1128	1926
HasMember	B	1090	Method Containing Access of a Member Element from Another Class	1128	1930
HasMember	B	1092	Use of Same Invokable Control Element in Multiple Architectural Layers	1128	1932
HasMember	B	1095	Loop Condition Value Update within the Loop	1128	1935
HasMember	B	1121	Excessive McCabe Cyclomatic Complexity	1128	1961

References

[REF-960]Object Management Group (OMG). "Automated Source Code Maintainability Measure (ASCMM)". 2016 January. < https://www.omg.org/spec/ASCMM/ >.2023-04-07.

[REF-968]Consortium for Information & Software Quality (CISQ). "Automated Quality Characteristic Measures". 2016. < http://it-cisq.org/standards/automated-quality-characteristic-measures/ >.

Category-1131: CISQ Quality Measures (2016) - Security

Category ID : 1131

Summary

Weaknesses in this category are related to the CISQ Quality Measures for Security, as documented in 2016 with the Automated Source Code Security Measure (ASCSM) Specification 1.0. Presence of these weaknesses could reduce the security of the software.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1128	CISQ Quality Measures (2016)	1128	2581
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	1128	33
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	1128	151
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	1128	163
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	1128	201
HasMember	C	99	Improper Control of Resource Identifiers ('Resource Injection')	1128	243
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	1128	304
HasMember	V	129	Improper Validation of Array Index	1128	341
HasMember	B	134	Use of Externally-Controlled Format String	1128	365
HasMember	B	252	Unchecked Return Value	1128	606
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	1128	799
HasMember	B	396	Declaration of Catch for Generic Exception	1128	959
HasMember	B	397	Declaration of Throws for Generic Exception	1128	961
HasMember	B	434	Unrestricted Upload of File with Dangerous Type	1128	1048
HasMember	V	456	Missing Initialization of a Variable	1128	1089
HasMember	B	606	Unchecked Input for Loop Condition	1128	1357
HasMember	C	667	Improper Locking	1128	1464

Nature	Type	ID	Name	V	Page
HasMember	C	672	Operation on a Resource after Expiration or Release	1128	1479
HasMember	B	681	Incorrect Conversion between Numeric Types	1128	1495
HasMember	B	772	Missing Release of Resource after Effective Lifetime	1128	1624
HasMember	V	789	Memory Allocation with Excessive Size Value	1128	1674
HasMember	B	798	Use of Hard-coded Credentials	1128	1690
HasMember	B	835	Loop with Unreachable Exit Condition ('Infinite Loop')	1128	1757

References

[REF-962]Object Management Group (OMG). "Automated Source Code Security Measure (ASCSM)". 2016 January. < http://www.omg.org/spec/ASCSM/1.0/ >.

[REF-968]Consortium for Information & Software Quality (CISQ). "Automated Quality Characteristic Measures". 2016. < http://it-cisq.org/standards/automated-quality-characteristic-measures/ >.

Category-1132: CISQ Quality Measures (2016) - Performance Efficiency

Category ID : 1132

Summary

Weaknesses in this category are related to the CISQ Quality Measures for Performance Efficiency, as documented in 2016 with the Automated Source Code Performance Efficiency Measure (ASCPEM) Specification 1.0. Presence of these weaknesses could reduce the performance efficiency of the software.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1128	CISQ Quality Measures (2016)	1128	2581
HasMember	V	1042	Static Member Data Element outside of a Singleton Class Element	1128	1876
HasMember	B	1043	Data Element Aggregating an Excessively Large Number of Non-Primitive Elements	1128	1877
HasMember	B	1046	Creation of Immutable Text Using String Concatenation	1128	1881
HasMember	B	1049	Excessive Data Query Operations in a Large Data Table	1128	1884
HasMember	B	1050	Excessive Platform Resource Consumption within a Loop	1128	1885
HasMember	B	1057	Data Access Operations Outside of Expected Data Manager Component	1128	1892
HasMember	B	1060	Excessive Number of Inefficient Server-Side Data Accesses	1128	1897
HasMember	B	1063	Creation of Class Instance within a Static Code Block	1128	1901
HasMember	B	1067	Excessive Execution of Sequential Searches of Data Resource	1128	1905
HasMember	B	1072	Data Resource Access without Use of Connection Pooling	1128	1912
HasMember	B	1073	Non-SQL Invokable Control Element with Excessive Number of Data Resource Accesses	1128	1913
HasMember	B	1089	Large Data Table with Excessive Number of Indices	1128	1929
HasMember	B	1091	Use of Object without Invoking Destructor Method	1128	1931
HasMember	B	1094	Excessive Index Range Scan for a Data Resource	1128	1934

References

[REF-959]Object Management Group (OMG). "Automated Source Code Performance Efficiency Measure (ASCPEM)". 2016 January. < https://www.omg.org/spec/ASCPEM/ >.2023-04-07.

[REF-968]Consortium for Information & Software Quality (CISQ). "Automated Quality Characteristic Measures". 2016. < http://it-cisq.org/standards/automated-quality-characteristic-measures/ >.

Category-1134: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 00. Input Validation and Data Sanitization (IDS)

Category ID : 1134

Summary

Weaknesses in this category are related to the rules and recommendations in the Input Validation and Data Sanitization (IDS) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	1133	151
HasMember	C	116	Improper Encoding or Escaping of Output	1133	281
HasMember	B	117	Improper Output Neutralization for Logs	1133	288
HasMember	B	134	Use of Externally-Controlled Format String	1133	365
HasMember	V	144	Improper Neutralization of Line Delimiters	1133	383
HasMember	V	150	Improper Neutralization of Escape, Meta, or Control Sequences	1133	394
HasMember	V	180	Incorrect Behavior Order: Validate Before Canonicalize	1133	451
HasMember	B	182	Collapse of Data into Unsafe Value	1133	455
HasMember	B	289	Authentication Bypass by Alternate Name	1133	703
HasMember	B	409	Improper Handling of Highly Compressed Data (Data Amplification)	1133	996

References

[REF-814]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 00. Input Validation and Data Sanitization (IDS)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487865 >.

[REF-996]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 00. Input Validation and Data Sanitization (IDS)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487337 >.

Category-1135: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 01. Declarations and Initialization (DCL)

Category ID : 1135

Summary

Weaknesses in this category are related to the rules and recommendations in the Declarations and Initialization (DCL) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	C	665	Improper Initialization	1133	1456

References

[REF-815]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 01. Declarations and Initialization (DCL)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487858 >.

[REF-997]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 01. Declarations and Initialization (DCL)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487329 >.

Category-1136: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 02. Expressions (EXP)

Category ID : 1136

Summary

Weaknesses in this category are related to the rules and recommendations in the Expressions (EXP) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	252	Unchecked Return Value	1133	606
HasMember	B	476	NULL Pointer Dereference	1133	1132
HasMember	V	595	Comparison of Object References Instead of Object Contents	1133	1334
HasMember	V	597	Use of Wrong Operator in String Comparison	1133	1337

References

[REF-816]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 02. Expressions (EXP)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487704 >.

[REF-998]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 02. Expressions (EXP)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487331 >.

Category-1137: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 03. Numeric Types and Operations (NUM)

Category ID : 1137

Summary

Weaknesses in this category are related to the rules and recommendations in the Numeric Types and Operations (NUM) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	190	Integer Overflow or Wraparound	1133	472
HasMember	B	191	Integer Underflow (Wrap or Wraparound)	1133	480
HasMember	B	197	Numeric Truncation Error	1133	500
HasMember	B	369	Divide By Zero	1133	913
HasMember	B	681	Incorrect Conversion between Numeric Types	1133	1495
HasMember	P	682	Incorrect Calculation	1133	1499

References

[REF-817]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 03. Numeric Types and Operations (NUM)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487628 >.

[REF-999]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 03. Numeric Types and Operations (NUM)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487335 >.

Category-1138: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 04. Characters and Strings (STR)

Category ID : 1138

Summary

Weaknesses in this category are related to the rules and recommendations in the Characters and Strings (STR) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	838	Inappropriate Encoding for Output Context	1133	1764

References

[REF-971]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 04. Characters and Strings (STR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487607 >.

[REF-1000]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 04. Characters and Strings (STR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487333 >.

Category-1139: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 05. Object Orientation (OBJ)

Category ID : 1139

Summary

Weaknesses in this category are related to the rules and recommendations in the Object Orientation (OBJ) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	374	Passing Mutable Objects to an Untrusted Method	1133	920
HasMember	B	375	Returning a Mutable Object to an Untrusted Caller	1133	923
HasMember	V	486	Comparison of Classes by Name	1133	1164
HasMember	V	491	Public cloneable() Method Without Final ('Object Hijack')	1133	1174
HasMember	V	492	Use of Inner Class Containing Sensitive Data	1133	1175
HasMember	V	498	Cloneable Class Containing Sensitive Information	1133	1196
HasMember	V	500	Public Static Field Not Marked Final	1133	1200
HasMember	B	766	Critical Data Element Declared Public	1133	1607

References

[REF-818]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 05. Object Orientation (OBJ)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487715 >.

[REF-1001]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 05. Object Orientation (OBJ)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487353 >.

Category-1140: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 06. Methods (MET)

Category ID : 1140

Summary

Weaknesses in this category are related to the rules and recommendations in the Methods (MET) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	V	568	finalize() Method Without super.finalize()	1133	1290
HasMember	C	573	Improper Following of Specification by Caller	1133	1298
HasMember	V	581	Object Model Violation: Just One of Equals and Hashcode Defined	1133	1312
HasMember	V	583	finalize() Method Declared Public	1133	1315
HasMember	B	586	Explicit Call to Finalize()	1133	1320
HasMember	V	589	Call to Non-ubiquitous API	1133	1325
HasMember	B	617	Reachable Assertion	1133	1378
HasMember	P	697	Incorrect Comparison	1133	1530

References

[REF-819]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 06. Methods (MET)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487441 >.

[REF-1002]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 06. Methods (MET)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487336 >.

Category-1141: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 07. Exceptional Behavior (ERR)

Category ID : 1141

Summary

Weaknesses in this category are related to the rules and recommendations in the Exceptional Behavior (ERR) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	248	Uncaught Exception	1133	596
HasMember	V	382	J2EE Bad Practices: Use of System.exit()	1133	933
HasMember	B	397	Declaration of Throws for Generic Exception	1133	961
HasMember	B	459	Incomplete Cleanup	1133	1099
HasMember	B	460	Improper Cleanup on Thrown Exception	1133	1102
HasMember	B	584	Return Inside Finally Block	1133	1317
HasMember	P	703	Improper Check or Handling of Exceptional Conditions	1133	1535
HasMember	C	705	Incorrect Control Flow Scoping	1133	1542
HasMember	C	754	Improper Check for Unusual or Exceptional Conditions	1133	1568

References

[REF-820]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 07. Exceptional Behavior (ERR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487665 >.

[REF-1003]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 07. Exceptional Behavior (ERR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487338 >.

Category-1142: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 08. Visibility and Atomicity (VNA)

Category ID : 1142

Summary

Weaknesses in this category are related to the rules and recommendations in the Visibility and Atomicity (VNA) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	C	362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	1133	888

Nature	Type	ID	Name	V	Page
HasMember	B	366	Race Condition within a Thread	1133	904
HasMember	B	413	Improper Resource Locking	1133	1003
HasMember	B	567	Unsynchronized Access to Shared Data in a Multithreaded Context	1133	1288
HasMember	C	662	Improper Synchronization	1133	1448
HasMember	C	667	Improper Locking	1133	1464

References

[REF-821]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 08. Visibility and Atomicity (VNA)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487824 >.

Category-1143: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 09. Locking (LCK)

Category ID : 1143

Summary

Weaknesses in this category are related to the rules and recommendations in the Locking (LCK) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	412	Unrestricted Externally Accessible Lock	1133	1000
HasMember	B	609	Double-Checked Locking	1133	1362
HasMember	C	667	Improper Locking	1133	1464
HasMember	B	820	Missing Synchronization	1133	1720

References

[REF-822]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 09. Locking (LCK)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487666 >.

Category-1144: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 10. Thread APIs (THI)

Category ID : 1144

Summary

Weaknesses in this category are related to the rules and recommendations in the Thread APIs (THI) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	V	572	Call to Thread run() instead of start()	1133	1296

References

[REF-823]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 10. Thread APIs (THI)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487735 >.

Category-1145: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 11. Thread Pools (TPS)

Category ID : 1145

Summary

Weaknesses in this category are related to the rules and recommendations in the Thread Pools (TPS) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	392	Missing Report of Error Condition	1133	951
HasMember	C	405	Asymmetric Resource Consumption (Amplification)	1133	986
HasMember	B	410	Insufficient Resource Pool	1133	998

References

[REF-824]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 11. Thread Pools (TPS)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487728 >.

Category-1146: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 12. Thread-Safety Miscellaneous (TSM)

Category ID : 1146

Summary

Weaknesses in this category are related to the rules and recommendations in the Thread-Safety Miscellaneous (TSM) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582

References

[REF-825]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 12. Thread-Safety Miscellaneous (TSM)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487731 >.

Category-1147: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 13. Input Output (FIO)

Category ID : 1147

Summary

Weaknesses in this category are related to the rules and recommendations in the Input Output (FIO) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	V	67	Improper Handling of Windows Device Names	1133	126
HasMember	V	180	Incorrect Behavior Order: Validate Before Canonicalize	1133	451
HasMember	V	198	Use of Incorrect Byte Ordering	1133	503
HasMember	B	276	Incorrect Default Permissions	1133	665
HasMember	V	279	Incorrect Execution-Assigned Permissions	1133	671
HasMember	B	359	Exposure of Private Personal Information to an Unauthorized Actor	1133	882
HasMember	C	377	Insecure Temporary File	1133	925
HasMember	C	404	Improper Resource Shutdown or Release	1133	980
HasMember	C	405	Asymmetric Resource Consumption (Amplification)	1133	986
HasMember	B	459	Incomplete Cleanup	1133	1099
HasMember	B	532	Insertion of Sensitive Information into Log File	1133	1241
HasMember	V	647	Use of Non-Canonical URL Paths for Authorization Decisions	1133	1426
HasMember	C	705	Incorrect Control Flow Scoping	1133	1542
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	1133	1551
HasMember	B	770	Allocation of Resources Without Limits or Throttling	1133	1613

References

[REF-826]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 13. Input Output (FIO)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487725 >.

[REF-1004]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 13. Input Output (FIO)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487330 >.

Category-1148: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 14. Serialization (SER)

Category ID : 1148

Summary

Weaknesses in this category are related to the rules and recommendations in the Serialization (SER) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	319	Cleartext Transmission of Sensitive Information	1133	779
HasMember	C	400	Uncontrolled Resource Consumption	1133	964

Nature	Type	ID	Name	V	Page
HasMember	V	499	Serializable Class Containing Sensitive Data	1133	1198
HasMember	B	502	Deserialization of Untrusted Data	1133	1204
HasMember	B	770	Allocation of Resources Without Limits or Throttling	1133	1613

References

[REF-827]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 14. Serialization (SER)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487787 >.

Category-1149: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 15. Platform Security (SEC)

Category ID : 1149

Summary

Weaknesses in this category are related to the rules and recommendations in the Platform Security (SEC) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	266	Incorrect Privilege Assignment	1133	638
HasMember	B	272	Least Privilege Violation	1133	656
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	1133	1551

References

[REF-828]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 15. Platform Security (SEC)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487683 >.

[REF-1005]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 15. Platform Security (SEC)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487332 >.

Category-1150: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 16. Runtime Environment (ENV)

Category ID : 1150

Summary

Weaknesses in this category are related to the rules and recommendations in the Runtime Environment (ENV) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	B	349	Acceptance of Extraneous Untrusted Data With Trusted Data	1133	861

Nature	Type	ID	Name	V	Page
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	1133	1551

References

[REF-829]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 16. Runtime Environment (ENV)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487764 >.

Category-1151: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 17. Java Native Interface (JNI)

Category ID : 1151

Summary

Weaknesses in this category are related to the rules and recommendations in the Java Native Interface (JNI) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	V	111	Direct Use of Unsafe JNI	1133	266

References

[REF-972]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 17. Java Native Interface (JNI)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487346 >.

Category-1152: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 49. Miscellaneous (MSC)

Category ID : 1152

Summary

Weaknesses in this category are related to the rules and recommendations in the Miscellaneous (MSC) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582
HasMember	V	259	Use of Hard-coded Password	1133	623
HasMember	C	311	Missing Encryption of Sensitive Data	1133	757
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	1133	799
HasMember	C	330	Use of Insufficiently Random Values	1133	814
HasMember	V	332	Insufficient Entropy in PRNG	1133	823
HasMember	V	336	Same Seed in Pseudo-Random Number Generator (PRNG)	1133	832
HasMember	V	337	Predictable Seed in Pseudo-Random Number Generator (PRNG)	1133	834

Nature	Type	ID	Name	V	Page
HasMember	C	400	Uncontrolled Resource Consumption	1133	964
HasMember	V	401	Missing Release of Memory after Effective Lifetime	1133	973
HasMember	B	770	Allocation of Resources Without Limits or Throttling	1133	1613
HasMember	B	798	Use of Hard-coded Credentials	1133	1690

References

[REF-830]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 49. Miscellaneous (MSC)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487686 >.

[REF-1006]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 49. Miscellaneous (MSC)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487351 >.

Category-1153: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 50. Android (DRD)

Category ID : 1153

Summary

Weaknesses in this category are related to the rules and recommendations in the Android (DRD) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582

References

[REF-973]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rule 50. Android (DRD)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487375 >.

Category-1155: SEI CERT C Coding Standard - Guidelines 01. Preprocessor (PRE)

Category ID : 1155

Summary

Weaknesses in this category are related to the rules and recommendations in the Preprocessor (PRE) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583

References

[REF-599]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 01. Preprocessor (PRE)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152276 >.

[REF-979]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 01. Preprocessor (PRE)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151965 >.

Category-1156: SEI CERT C Coding Standard - Guidelines 02. Declarations and Initialization (DCL)

Category ID : 1156

Summary

Weaknesses in this category are related to the rules and recommendations in the Declarations and Initialization (DCL) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	B	562	Return of Stack Variable Address	1154	1278

References

[REF-600]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 02. Declarations and Initialization (DCL)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152215 >.

[REF-980]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 02. Declarations and Initialization (DCL)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151966 >.

Category-1157: SEI CERT C Coding Standard - Guidelines 03. Expressions (EXP)

Category ID : 1157

Summary

Weaknesses in this category are related to the rules and recommendations in the Expressions (EXP) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	1154	293
HasMember	B	125	Out-of-bounds Read	1154	330
HasMember	B	476	NULL Pointer Dereference	1154	1132
HasMember	B	480	Use of Incorrect Operator	1154	1150
HasMember	V	481	Assigning instead of Comparing	1154	1154
HasMember	B	628	Function Call with Incorrectly Specified Arguments	1154	1398
HasMember	V	685	Function Call With Incorrect Number of Arguments	1154	1507
HasMember	V	686	Function Call With Incorrect Argument Type	1154	1508
HasMember		690	Unchecked Return Value to NULL Pointer Dereference	1154	1514

Nature	Type	ID	Name	V	Page
HasMember	C	704	Incorrect Type Conversion or Cast	1154	1538
HasMember	C	758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	1154	1582
HasMember	B	843	Access of Resource Using Incompatible Type ('Type Confusion')	1154	1776
HasMember	B	908	Use of Uninitialized Resource	1154	1792

References

[REF-601]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 03. Expressions (EXP)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152200 >.

[REF-981]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 03. Expressions (EXP)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151976 >.

Category-1158: SEI CERT C Coding Standard - Guidelines 04. Integers (INT)

Category ID : 1158

Summary

Weaknesses in this category are related to the rules and recommendations in the Integers (INT) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	B	131	Incorrect Calculation of Buffer Size	1154	355
HasMember	B	190	Integer Overflow or Wraparound	1154	472
HasMember	B	191	Integer Underflow (Wrap or Wraparound)	1154	480
HasMember	V	192	Integer Coercion Error	1154	482
HasMember	V	194	Unexpected Sign Extension	1154	491
HasMember	V	195	Signed to Unsigned Conversion Error	1154	494
HasMember	B	197	Numeric Truncation Error	1154	500
HasMember	B	369	Divide By Zero	1154	913
HasMember	V	587	Assignment of a Fixed Address to a Pointer	1154	1322
HasMember	CH	680	Integer Overflow to Buffer Overflow	1154	1493
HasMember	B	681	Incorrect Conversion between Numeric Types	1154	1495
HasMember	P	682	Incorrect Calculation	1154	1499
HasMember	C	704	Incorrect Type Conversion or Cast	1154	1538
HasMember	C	758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	1154	1582

References

[REF-602]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 04. Integers (INT)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152052 >.

[REF-982]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec. 04. Integers (INT)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151979 >.

Category-1159: SEI CERT C Coding Standard - Guidelines 05. Floating Point (FLP)

Category ID : 1159

Summary

Weaknesses in this category are related to the rules and recommendations in the Floating Point (FLP) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	B	197	Numeric Truncation Error	1154	500
HasMember	B	391	Unchecked Error Condition	1154	948
HasMember	B	681	Incorrect Conversion between Numeric Types	1154	1495
HasMember	P	682	Incorrect Calculation	1154	1499

References

[REF-603]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 05. Floating Point (FLP)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152181 >.

[REF-983]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 05. Floating Point (FLP)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151969 >.

Category-1160: SEI CERT C Coding Standard - Guidelines 06. Arrays (ARR)

Category ID : 1160

Summary

Weaknesses in this category are related to the rules and recommendations in the Arrays (ARR) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	1154	293
HasMember	V	121	Stack-based Buffer Overflow	1154	314
HasMember	B	123	Write-what-where Condition	1154	323
HasMember	B	125	Out-of-bounds Read	1154	330
HasMember	V	129	Improper Validation of Array Index	1154	341
HasMember	B	468	Incorrect Pointer Scaling	1154	1114
HasMember	B	469	Use of Pointer Subtraction to Determine Size	1154	1115
HasMember	C	758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	1154	1582
HasMember	B	786	Access of Memory Location Before Start of Buffer	1154	1658
HasMember	B	805	Buffer Access with Incorrect Length Value	1154	1702

References

[REF-604]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 06. Arrays (ARR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152051 >.

[REF-984]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 06. Arrays (ARR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151972 >.

Category-1161: SEI CERT C Coding Standard - Guidelines 07. Characters and Strings (STR)

Category ID : 1161

Summary

Weaknesses in this category are related to the rules and recommendations in the Characters and Strings (STR) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	1154	293
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	1154	304
HasMember	V	121	Stack-based Buffer Overflow	1154	314
HasMember	V	122	Heap-based Buffer Overflow	1154	318
HasMember	B	123	Write-what-where Condition	1154	323
HasMember	B	125	Out-of-bounds Read	1154	330
HasMember	B	170	Improper Null Termination	1154	428
HasMember	B	676	Use of Potentially Dangerous Function	1154	1489
HasMember	C	704	Incorrect Type Conversion or Cast	1154	1538

References

[REF-605]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 07. Characters and Strings (STR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152038 >.

[REF-985]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 07. Characters and Strings (STR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151974 >.

Category-1162: SEI CERT C Coding Standard - Guidelines 08. Memory Management (MEM)

Category ID : 1162

Summary

Weaknesses in this category are related to the rules and recommendations in the Memory Management (MEM) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	B	131	Incorrect Calculation of Buffer Size	1154	355
HasMember	B	190	Integer Overflow or Wraparound	1154	472
HasMember	V	401	Missing Release of Memory after Effective Lifetime	1154	973
HasMember	C	404	Improper Resource Shutdown or Release	1154	980
HasMember	V	415	Double Free	1154	1008
HasMember	V	416	Use After Free	1154	1012
HasMember	B	459	Incomplete Cleanup	1154	1099
HasMember	V	467	Use of sizeof() on a Pointer Type	1154	1110
HasMember	V	590	Free of Memory not on the Heap	1154	1326
HasMember	C	666	Operation on Resource in Wrong Phase of Lifetime	1154	1462
HasMember	C	672	Operation on a Resource after Expiration or Release	1154	1479
HasMember	Chain	680	Integer Overflow to Buffer Overflow	1154	1493
HasMember	C	758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	1154	1582
HasMember	B	771	Missing Reference to Active Allocated Resource	1154	1622
HasMember	B	772	Missing Release of Resource after Effective Lifetime	1154	1624
HasMember	V	789	Memory Allocation with Excessive Size Value	1154	1674

References

[REF-606]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 08. Memory Management (MEM)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152142 >.

[REF-986]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec. 08. Memory Management (MEM)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151930 >.

Category-1163: SEI CERT C Coding Standard - Guidelines 09. Input Output (FIO)

Category ID : 1163

Summary

Weaknesses in this category are related to the rules and recommendations in the Input Output (FIO) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	C	20	Improper Input Validation	1154	20
HasMember	V	67	Improper Handling of Windows Device Names	1154	126
HasMember	B	134	Use of Externally-Controlled Format String	1154	365
HasMember	B	197	Numeric Truncation Error	1154	500
HasMember	B	241	Improper Handling of Unexpected Data Type	1154	584
HasMember	C	404	Improper Resource Shutdown or Release	1154	980
HasMember	B	459	Incomplete Cleanup	1154	1099
HasMember	P	664	Improper Control of a Resource Through its Lifetime	1154	1454

Nature	Type	ID	Name	V	Page
HasMember	C	666	Operation on Resource in Wrong Phase of Lifetime	1154	1462
HasMember	C	672	Operation on a Resource after Expiration or Release	1154	1479
HasMember	V	685	Function Call With Incorrect Number of Arguments	1154	1507
HasMember	V	686	Function Call With Incorrect Argument Type	1154	1508
HasMember	C	758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	1154	1582
HasMember	B	771	Missing Reference to Active Allocated Resource	1154	1622
HasMember	B	772	Missing Release of Resource after Effective Lifetime	1154	1624
HasMember	V	773	Missing Reference to Active File Descriptor or Handle	1154	1629
HasMember	V	775	Missing Release of File Descriptor or Handle after Effective Lifetime	1154	1631
HasMember	B	910	Use of Expired File Descriptor	1154	1800

References

[REF-607]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 09. Input Output (FIO)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152270 >.

[REF-987]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 09. Input Output (FIO)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151932 >.

Category-1165: SEI CERT C Coding Standard - Guidelines 10. Environment (ENV)

Category ID : 1165

Summary

Weaknesses in this category are related to the rules and recommendations in the Environment (ENV) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	1154	151
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	1154	194
HasMember	B	676	Use of Potentially Dangerous Function	1154	1489
HasMember	C	705	Incorrect Control Flow Scoping	1154	1542

References

[REF-608]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 10. Environment (ENV)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152421 >.

[REF-988]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec. 10. Environment (ENV)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151968 >.

Category-1166: SEI CERT C Coding Standard - Guidelines 11. Signals (SIG)

Category ID : 1166

Summary

Weaknesses in this category are related to the rules and recommendations in the Signals (SIG) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	V	479	Signal Handler Use of a Non-reentrant Function	1154	1147
HasMember	C	662	Improper Synchronization	1154	1448

References

[REF-609]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 11. Signals (SIG)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152469 >.

[REF-989]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 11. Signals (SIG)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151975 >.

Category-1167: SEI CERT C Coding Standard - Guidelines 12. Error Handling (ERR)

Category ID : 1167

Summary

Weaknesses in this category are related to the rules and recommendations in the Error Handling (ERR) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	B	252	Unchecked Return Value	1154	606
HasMember	B	253	Incorrect Check of Function Return Value	1154	613
HasMember	B	391	Unchecked Error Condition	1154	948
HasMember	V	456	Missing Initialization of a Variable	1154	1089
HasMember	B	676	Use of Potentially Dangerous Function	1154	1489
HasMember	C	758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	1154	1582

References

[REF-610]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 12. Error Handling (ERR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152345 >.

[REF-990]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 12. Error Handling (ERR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151977 >.

Category-1168: SEI CERT C Coding Standard - Guidelines 13. Application Programming Interfaces (API)

Category ID : 1168

Summary

Weaknesses in this category are related to the rules and recommendations in the Application Programming Interfaces (API) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583

References

[REF-611]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 13. Application Programming Interfaces (API)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152242 >.

[REF-991]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 13. Application Programming Interfaces (API)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151980 >.

Category-1169: SEI CERT C Coding Standard - Guidelines 14. Concurrency (CON)

Category ID : 1169

Summary

Weaknesses in this category are related to the rules and recommendations in the Concurrency (CON) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	C	330	Use of Insufficiently Random Values	1154	814
HasMember	B	366	Race Condition within a Thread	1154	904
HasMember	C	377	Insecure Temporary File	1154	925
HasMember	C	667	Improper Locking	1154	1464
HasMember	B	676	Use of Potentially Dangerous Function	1154	1489

References

[REF-612]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 14. Concurrency (CON)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152257 >.

[REF-992]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 14. Concurrency (CON)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151970 >.

Category-1170: SEI CERT C Coding Standard - Guidelines 48. Miscellaneous (MSC)

Category ID : 1170

Summary

Weaknesses in this category are related to the rules and recommendations in the Miscellaneous (MSC) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	1154	799
HasMember	C	330	Use of Insufficiently Random Values	1154	814
HasMember	B	331	Insufficient Entropy	1154	821
HasMember	B	338	Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)	1154	837
HasMember	B	676	Use of Potentially Dangerous Function	1154	1489
HasMember	C	758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	1154	1582

References

[REF-613]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 48. Miscellaneous (MSC)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152201 >.

[REF-993]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 48. Miscellaneous (MSC)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151973 >.

Category-1171: SEI CERT C Coding Standard - Guidelines 50. POSIX (POS)

Category ID : 1171

Summary

Weaknesses in this category are related to the rules and recommendations in the POSIX (POS) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	B	170	Improper Null Termination	1154	428
HasMember	B	242	Use of Inherently Dangerous Function	1154	586
HasMember	B	252	Unchecked Return Value	1154	606
HasMember	B	253	Incorrect Check of Function Return Value	1154	613
HasMember	B	273	Improper Check for Dropped Privileges	1154	660
HasMember	B	363	Race Condition Enabling Link Following	1154	897
HasMember	B	391	Unchecked Error Condition	1154	948
HasMember	C	667	Improper Locking	1154	1464
HasMember	C	696	Incorrect Behavior Order	1154	1527

References

[REF-614]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 50. POSIX (POS)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152405 >.

[REF-994]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 50. POSIX (POS)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151931 >.

Category-1172: SEI CERT C Coding Standard - Guidelines 51. Microsoft Windows (WIN)

Category ID : 1172

Summary

Weaknesses in this category are related to the rules and recommendations in the Microsoft Windows (WIN) section of the SEI CERT C Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1154	Weaknesses Addressed by the SEI CERT C Coding Standard	1154	2583
HasMember	V	590	Free of Memory not on the Heap	1154	1326
HasMember	V	762	Mismatched Memory Management Routines	1154	1596

References

[REF-617]The Software Engineering Institute. "SEI CERT C Coding Standard : Rule 51. Microsoft Windows (WIN)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151925 >.

[REF-995]The Software Engineering Institute. "SEI CERT C Coding Standard : Rec 51. Microsoft Windows (WIN)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151933 >.

Category-1175: SEI CERT Oracle Secure Coding Standard for Java - Guidelines 18. Concurrency (CON)

Category ID : 1175

Summary

Weaknesses in this category are related to the rules and recommendations in the Concurrency (CON) section of the SEI CERT Oracle Secure Coding Standard for Java.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1133	Weaknesses Addressed by the SEI CERT Oracle Coding Standard for Java	1133	2582

References

[REF-1007]The Software Engineering Institute. "SEI CERT Oracle Coding Standard for Java : Rec 18. Concurrency (CON)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88487352 >.

Category-1179: SEI CERT Perl Coding Standard - Guidelines 01. Input Validation and Data Sanitization (IDS)

Category ID : 1179

Summary

Weaknesses in this category are related to the rules and recommendations in the Input Validation and Data Sanitization (IDS) section of the SEI CERT Perl Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1178	Weaknesses Addressed by the SEI CERT Perl Coding Standard	1178	2585
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	1178	33
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	1178	145
HasMember	V	95	Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')	1178	226
HasMember	C	116	Improper Encoding or Escaping of Output	1178	281
HasMember	V	129	Improper Validation of Array Index	1178	341
HasMember	B	134	Use of Externally-Controlled Format String	1178	365
HasMember	V	789	Memory Allocation with Excessive Size Value	1178	1674

References

[REF-1012]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rule 01. Input Validation and Data Sanitization (IDS)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890533 >.

[REF-1020]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rec. 01. Input Validation and Data Sanitization (IDS)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890568 >.

Category-1180: SEI CERT Perl Coding Standard - Guidelines 02. Declarations and Initialization (DCL)

Category ID : 1180

Summary

Weaknesses in this category are related to the rules and recommendations in the Declarations and Initialization (DCL) section of the SEI CERT Perl Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1178	Weaknesses Addressed by the SEI CERT Perl Coding Standard	1178	2585
HasMember	V	456	Missing Initialization of a Variable	1178	1089
HasMember	V	457	Use of Uninitialized Variable	1178	1094
HasMember	B	477	Use of Obsolete Function	1178	1138
HasMember	B	628	Function Call with Incorrectly Specified Arguments	1178	1398

References

[REF-1013]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rule 02. Declarations and Initialization (DCL)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890509 >.

[REF-1021]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rec. 02. Declarations and Initialization (DCL)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890569 >.

Category-1181: SEI CERT Perl Coding Standard - Guidelines 03. Expressions (EXP)

Category ID : 1181

Summary

Weaknesses in this category are related to the rules and recommendations in the Expressions (EXP) section of the SEI CERT Perl Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1178	Weaknesses Addressed by the SEI CERT Perl Coding Standard	1178	2585
HasMember	B	248	Uncaught Exception	1178	596
HasMember	B	252	Unchecked Return Value	1178	606
HasMember	B	375	Returning a Mutable Object to an Untrusted Caller	1178	923
HasMember	B	391	Unchecked Error Condition	1178	948
HasMember	B	394	Unexpected Status Code or Return Value	1178	955
HasMember	B	460	Improper Cleanup on Thrown Exception	1178	1102
HasMember	B	477	Use of Obsolete Function	1178	1138
HasMember	V	597	Use of Wrong Operator in String Comparison	1178	1337
HasMember	B	628	Function Call with Incorrectly Specified Arguments	1178	1398
HasMember	C	690	Unchecked Return Value to NULL Pointer Dereference	1178	1514
HasMember	C	705	Incorrect Control Flow Scoping	1178	1542
HasMember	C	754	Improper Check for Unusual or Exceptional Conditions	1178	1568
HasMember	B	783	Operator Precedence Logic Error	1178	1650

References

[REF-1014]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rule 03. Expressions (EXP)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890504 >.

[REF-1022]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rec. 03. Expressions (EXP)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890559 >.

Category-1182: SEI CERT Perl Coding Standard - Guidelines 04. Integers (INT)

Category ID : 1182

Summary

Weaknesses in this category are related to the rules and recommendations in the Integers (INT) section of the SEI CERT Perl Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1178	Weaknesses Addressed by the SEI CERT Perl Coding Standard	1178	2585
HasMember	C	189	Numeric Errors	1178	2312

References

[REF-1015]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rule 04. Integers (INT)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890508 >.

[REF-1023]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rec. 04. Integers (INT)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890560 >.

Category-1183: SEI CERT Perl Coding Standard - Guidelines 05. Strings (STR)

Category ID : 1183

Summary

Weaknesses in this category are related to the rules and recommendations in the Strings (STR) section of the SEI CERT Perl Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1178	Weaknesses Addressed by the SEI CERT Perl Coding Standard	1178	2585

References

[REF-1016]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rule 05. Strings (STR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890507 >.

[REF-1024]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rec. 05. Strings (STR)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890563 >.

Category-1184: SEI CERT Perl Coding Standard - Guidelines 06. Object-Oriented Programming (OOP)

Category ID : 1184

Summary

Weaknesses in this category are related to the rules and recommendations in the Object-Oriented Programming (OOP) section of the SEI CERT Perl Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1178	Weaknesses Addressed by the SEI CERT Perl Coding Standard	1178	2585
HasMember	B	767	Access to Critical Private Variable via Public Method	1178	1610

References

[REF-1017]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rule 06. Object-Oriented Programming (OOP)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890501 >.

[REF-1025]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rec. 06. Object-Oriented Programming (OOP)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890561 >.

Category-1185: SEI CERT Perl Coding Standard - Guidelines 07. File Input and Output (FIO)

Category ID : 1185

Summary

Weaknesses in this category are related to the rules and recommendations in the File Input and Output (FIO) section of the SEI CERT Perl Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1178	Weaknesses Addressed by the SEI CERT Perl Coding Standard	1178	2585
HasMember	B	59	Improper Link Resolution Before File Access ('Link Following')	1178	111

References

[REF-1018]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rule 07. File Input and Output (FIO)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890499 >.

[REF-1026]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rec. 07. File Input and Output (FIO)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890496 >.

Category-1186: SEI CERT Perl Coding Standard - Guidelines 50. Miscellaneous (MSC)

Category ID : 1186

Summary

Weaknesses in this category are related to the rules and recommendations in the Miscellaneous (MSC) section of the SEI CERT Perl Coding Standard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1178	Weaknesses Addressed by the SEI CERT Perl Coding Standard	1178	2585
HasMember	B	561	Dead Code	1178	1275
HasMember	B	563	Assignment to Variable without Use	1178	1280

References

[REF-1019]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rule 50. Miscellaneous (MSC)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890497 >.

[REF-1027]The Software Engineering Institute. "SEI CERT Perl Coding Standard : Rule 50. Miscellaneous (MSC)". < https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88890502 >.

Category-1195: Manufacturing and Life Cycle Management Concerns

Category ID : 1195

Summary

Weaknesses in this category are root-caused to defects that arise in the semiconductor-manufacturing process or during the life cycle and supply chain.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	C	1059	Insufficient Technical Documentation	1194	1894
HasMember	B	1248	Semiconductor Defects in Hardware Logic with Security-Sensitive Implications	1194	2049
HasMember	B	1266	Improper Scrubbing of Sensitive Data from Decommissioned Device	1194	2091
HasMember	B	1269	Product Released in Non-Release Configuration	1194	2098
HasMember	B	1273	Device Unlock Credential Sharing	1194	2106
HasMember	B	1297	Unprotected Confidential Information on Device is Accessible by OSAT Vendors	1194	2156

Category-1196: Security Flow Issues

Category ID : 1196

Summary

Weaknesses in this category are related to improper design of full-system security flows, including but not limited to secure boot, secure update, and hardware-device attestation.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	B	1190	DMA Device Enabled Too Early in Boot Phase	1194	1978
HasMember	B	1193	Power-On of Untrusted Execution Core Before Enabling Fabric Access Control	1194	1986
HasMember	B	1264	Hardware Logic with Insecure De-Synchronization between Control and Data Channels	1194	2086
HasMember	B	1274	Improper Access Control for Volatile Memory Containing Boot Code	1194	2108
HasMember	B	1283	Mutable Attestation or Measurement Reporting Data	1194	2128
HasMember	B	1310	Missing Ability to Patch ROM Code	1194	2179
HasMember	B	1326	Missing Immutable Root of Trust in Hardware	1194	2212
HasMember	B	1328	Security Version Number Mutable to Older Versions	1194	2217

Category-1197: Integration Issues

Category ID : 1197

Summary

Weaknesses in this category are those that arise due to integration of multiple hardware Intellectual Property (IP) cores, from System-on-a-Chip (SoC) subsystem interactions, or from hardware platform subsystem interactions.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	B	1276	Hardware Child Block Incorrectly Connected to Parent System	1194	2113

Category-1198: Privilege Separation and Access Control Issues

Category ID : 1198

Summary

Weaknesses in this category are related to features and mechanisms providing hardware-based isolation and access control (e.g., identity, policy, locking control) of sensitive shared hardware resources such as registers and fuses.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
MemberOf	C	1372	ICS Supply Chain: OT Counterfeit and Malicious Corruption	1358	2509
HasMember	B	276	Incorrect Default Permissions	1194	665
HasMember	C	441	Unintended Proxy or Intermediary ('Confused Deputy')	1194	1064
HasMember	B	1189	Improper Isolation of Shared Resources on System-on-a-Chip (SoC)	1194	1976
HasMember	B	1192	Improper Identifier for IP Block used in System-On-Chip (SOC)	1194	1985
HasMember	B	1220	Insufficient Granularity of Access Control	1194	1992
HasMember	V	1222	Insufficient Granularity of Address Regions Protected by Register Locks	1194	1999
HasMember	B	1242	Inclusion of Undocumented Features or Chicken Bits	1194	2033
HasMember	B	1260	Improper Handling of Overlap Between Protected Memory Ranges	1194	2075
HasMember	B	1262	Improper Access Control for Register Interface	1194	2081
HasMember	B	1267	Policy Uses Obsolete Encoding	1194	2093
HasMember	B	1268	Policy Privileges are not Assigned Consistently Between Control and Data Agents	1194	2095
HasMember	B	1280	Access Control Check Implemented After Asset is Accessed	1194	2122
HasMember	C	1294	Insecure Security Identifier Mechanism	1194	2150
HasMember	B	1299	Missing Protection Mechanism for Alternate Hardware Interface	1194	2162
HasMember	B	1302	Missing Source Identifier in Entity Transactions on a System-On-Chip (SOC)	1194	2172

Nature	Type	ID	Name	V	Page
HasMember	B	1303	Non-Transparent Sharing of Microarchitectural Resources	1194	2174
HasMember	B	1314	Missing Write Protection for Parametric Data Values	1194	2187
HasMember	B	1318	Missing Support for Security Features in On-chip Fabrics or Buses	1194	2197
HasMember	B	1334	Unauthorized Error Injection Can Degrade Hardware Redundancy	1194	2234
HasMember	B	1420	Exposure of Sensitive Information during Transient Execution	1194	2284

Category-1199: General Circuit and Logic Design Concerns

Category ID : 1199

Summary

Weaknesses in this category are related to hardware-circuit design and logic (e.g., CMOS transistors, finite state machines, and registers) as well as issues related to hardware description languages such as System Verilog and VHDL.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	B	1209	Failure to Disable Reserved Bits	1194	1991
HasMember	B	1221	Incorrect Register Defaults or Module Parameters	1194	1996
HasMember	B	1223	Race Condition for Write-Once Attributes	1194	2001
HasMember	B	1224	Improper Restriction of Write-Once Bit Fields	1194	2003
HasMember	B	1231	Improper Prevention of Lock Bit Modification	1194	2007
HasMember	B	1232	Improper Lock Behavior After Power State Transition	1194	2010
HasMember	B	1233	Security-Sensitive Hardware Controls with Missing Lock Bit Protection	1194	2012
HasMember	B	1234	Hardware Internal or Debug Modes Allow Override of Locks	1194	2014
HasMember	B	1245	Improper Finite State Machines (FSMs) in Hardware Logic	1194	2041
HasMember	B	1250	Improper Preservation of Consistency Between Independent Representations of Shared State	1194	2052
HasMember	B	1253	Incorrect Selection of Fuse Values	1194	2058
HasMember	B	1254	Incorrect Comparison Logic Granularity	1194	2060
HasMember	B	1261	Improper Handling of Single Event Upsets	1194	2079
HasMember	B	1298	Hardware Logic Contains Race Conditions	1194	2158

Category-1201: Core and Compute Issues

Category ID : 1201

Summary

Weaknesses in this category are typically associated with CPUs, Graphics, Vision, AI, FPGA, and microcontrollers.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	B	1252	CPU Hardware Not Configured to Support Exclusivity of Write and Execute Operations	1194	2056
HasMember	B	1281	Sequence of Processor Instructions Leads to Unexpected Behavior	1194	2124
HasMember	B	1342	Information Exposure through Microarchitectural State after Transient Execution	1194	2250
HasMember	B	1420	Exposure of Sensitive Information during Transient Execution	1194	2284

Category-1202: Memory and Storage Issues

Category ID : 1202

Summary

Weaknesses in this category are typically associated with memory (e.g., DRAM, SRAM) and storage technologies (e.g., NAND Flash, OTP, EEPROM, and eMMC).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	B	226	Sensitive Information in Resource Not Removed Before Reuse	1194	562
HasMember	B	1246	Improper Write Handling in Limited-write Non-Volatile Memories	1194	2043
HasMember	B	1251	Mirrored Regions with Different Values	1194	2054
HasMember	B	1257	Improper Access Control Applied to Mirrored or Aliased Memory Regions	1194	2068
HasMember	B	1282	Assumed-Immutable Data is Stored in Writable Memory	1194	2127
HasMember	B	1420	Exposure of Sensitive Information during Transient Execution	1194	2284

Category-1203: Peripherals, On-chip Fabric, and Interface/IO Problems

Category ID : 1203

Summary

Weaknesses in this category are related to hardware security problems that apply to peripheral devices, IO interfaces, on-chip interconnects, network-on-chip (NoC), and buses. For example, this category includes issues related to design of hardware interconnect and/or protocols such as PCIe, USB, SMBUS, general-purpose IO pins, and user-input peripherals such as mouse and keyboard.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	B	1311	Improper Translation of Security Attributes by Fabric Bridge	1194	2182

Nature	Type	ID	Name	V	Page
HasMember	B	1312	Missing Protection for Mirrored Regions in On-Chip Fabric Firewall	1194	2184
HasMember	B	1315	Improper Setting of Bus Controlling Capability in Fabric End-point	1194	2190
HasMember	B	1316	Fabric-Address Map Allows Programming of Unwarranted Overlaps of Protected and Unprotected Ranges	1194	2192
HasMember	B	1317	Improper Access Control in Fabric Bridge	1194	2194
HasMember	B	1331	Improper Isolation of Shared Resources in Network On Chip (NoC)	1194	2225

Category-1205: Security Primitives and Cryptography Issues

Category ID : 1205

Summary

Weaknesses in this category are related to hardware implementations of cryptographic protocols and other hardware-security primitives such as physical unclonable functions (PUFs) and random number generators (RNGs).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	B	203	Observable Discrepancy	1194	518
HasMember	B	325	Missing Cryptographic Step	1194	794
HasMember	B	1240	Use of a Cryptographic Primitive with a Risky Implementation	1194	2025
HasMember	B	1241	Use of Predictable Algorithm in Random Number Generator	1194	2030
HasMember	B	1279	Cryptographic Operations are run Before Supporting Units are Ready	1194	2120
HasMember	B	1351	Improper Handling of Hardware Behavior in Exceptionally Cold Environments	1194	2252

Category-1206: Power, Clock, Thermal, and Reset Concerns

Category ID : 1206

Summary

Weaknesses in this category are related to system power, voltage, current, temperature, clocks, system state saving/restoring, and resets at the platform and SoC level.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	B	1232	Improper Lock Behavior After Power State Transition	1194	2010
HasMember	B	1247	Improper Protection Against Voltage and Clock Glitches	1194	2044
HasMember	B	1248	Semiconductor Defects in Hardware Logic with Security-Sensitive Implications	1194	2049

Nature	Type	ID	Name	V	Page
HasMember	V	1255	Comparison Logic is Vulnerable to Power Side-Channel Attacks	1194	2062
HasMember	B	1256	Improper Restriction of Software Interfaces to Hardware Features	1194	2065
HasMember	B	1271	Uninitialized Value on Reset for Registers Holding Security Settings	1194	2102
HasMember	B	1304	Improperly Preserved Integrity of Hardware Configuration State During a Power Save/Restore Operation	1194	2176
HasMember	B	1314	Missing Write Protection for Parametric Data Values	1194	2187
HasMember	B	1320	Improper Protection for Outbound Error Messages and Alert Signals	1194	2202
HasMember	B	1332	Improper Handling of Faults that Lead to Instruction Skips	1194	2227
HasMember	B	1338	Improper Protections Against Hardware Overheating	1194	2240

Category-1207: Debug and Test Problems

Category ID : 1207

Summary

Weaknesses in this category are related to hardware debug and test interfaces such as JTAG and scan chain.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	B	319	Cleartext Transmission of Sensitive Information	1194	779
HasMember	B	1191	On-Chip Debug and Test Interface With Improper Access Control	1194	1980
HasMember	B	1234	Hardware Internal or Debug Modes Allow Override of Locks	1194	2014
HasMember	B	1243	Sensitive Non-Volatile Information Not Protected During Debug	1194	2035
HasMember	B	1244	Internal Asset Exposed to Unsafe Debug Access Level or State	1194	2037
HasMember	B	1258	Exposure of Sensitive System Information Due to Uncleared Debug Information	1194	2071
HasMember	B	1272	Sensitive Information Uncleared Before Debug/Power State Transition	1194	2104
HasMember	B	1291	Public Key Re-Use for Signing both Debug and Production Code	1194	2145
HasMember	B	1295	Debug Messages Revealing Unnecessary Information	1194	2152
HasMember	B	1296	Incorrect Chaining or Granularity of Debug Components	1194	2153
HasMember	B	1313	Hardware Allows Activation of Test or Debug Logic at Runtime	1194	2185
HasMember	B	1323	Improper Management of Sensitive Trace Data	1194	2208

Category-1208: Cross-Cutting Problems

Category ID : 1208

Summary

Weaknesses in this category can arise in multiple areas of hardware design or can apply to a wide cross-section of components.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1194	Hardware Design	1194	2586
HasMember	B	440	Expected Behavior Violation	1194	1062
HasMember	B	1053	Missing Documentation for Design	1194	1888
HasMember	C	1059	Insufficient Technical Documentation	1194	1894
HasMember	C	1263	Improper Physical Access Control	1194	2085
HasMember	B	1277	Firmware Not Updateable	1194	2116
HasMember	B	1301	Insufficient or Incomplete Data Removal within Hardware Component	1194	2170
HasMember	B	1329	Reliance on Component That is Not Updateable	1194	2219
HasMember	C	1357	Reliance on Insufficiently Trustworthy Component	1194	2254

Category-1210: Audit / Logging Errors

Category ID : 1210

Summary

Weaknesses in this category are related to audit-based components of a software system. Frequently these deal with logging user activities in order to identify undesired access and modifications to the system. The weaknesses in this category could lead to a degradation of the quality of the audit capability if they are not addressed.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	117	Improper Output Neutralization for Logs	699	288
HasMember	B	222	Truncation of Security-relevant Information	699	557
HasMember	B	223	Omission of Security-relevant Information	699	559
HasMember	B	224	Obscured Security-relevant Information by Alternate Name	699	561
HasMember	B	778	Insufficient Logging	699	1638
HasMember	B	779	Logging of Excessive Data	699	1642

Category-1211: Authentication Errors

Category ID : 1211

Summary

Weaknesses in this category are related to authentication components of a system. Frequently these deal with the ability to verify that an entity is indeed who it claims to be. If not addressed when designing or implementing a software system, these weaknesses could lead to a degradation of the quality of the authentication capability.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	289	Authentication Bypass by Alternate Name	699	703
HasMember	B	290	Authentication Bypass by Spoofing	699	705
HasMember	B	294	Authentication Bypass by Capture-replay	699	712
HasMember	B	295	Improper Certificate Validation	699	714
HasMember	B	301	Reflection Attack in an Authentication Protocol	699	733
HasMember	B	303	Incorrect Implementation of Authentication Algorithm	699	737
HasMember	B	305	Authentication Bypass by Primary Weakness	699	740
HasMember	B	306	Missing Authentication for Critical Function	699	741
HasMember	B	307	Improper Restriction of Excessive Authentication Attempts	699	747
HasMember	B	308	Use of Single-factor Authentication	699	752
HasMember	B	309	Use of Password System for Primary Authentication	699	754
HasMember	B	322	Key Exchange without Entity Authentication	699	788
HasMember	B	603	Use of Client-Side Authentication	699	1354
HasMember	B	645	Overly Restrictive Account Lockout Mechanism	699	1423
HasMember	B	804	Guessable CAPTCHA	699	1701
HasMember	B	836	Use of Password Hash Instead of Password for Authentication	699	1761

Category-1212: Authorization Errors

Category ID : 1212

Summary

Weaknesses in this category are related to authorization components of a system. Frequently these deal with the ability to enforce that agents have the required permissions before performing certain operations, such as modifying data. If not addressed when designing or implementing a software system, these weaknesses could lead to a degradation of the quality of the authorization capability.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	425	Direct Request ('Forced Browsing')	699	1025
HasMember	B	551	Incorrect Behavior Order: Authorization Before Parsing and Canonicalization	699	1264
HasMember	B	552	Files or Directories Accessible to External Parties	699	1265
HasMember	B	639	Authorization Bypass Through User-Controlled Key	699	1406
HasMember	C	653	Improper Isolation or Compartmentalization	699	1437
HasMember	B	842	Placement of User into Incorrect Group	699	1775
HasMember	B	939	Improper Authorization in Handler for Custom URL Scheme	699	1840
HasMember	B	1220	Insufficient Granularity of Access Control	699	1992
HasMember	B	1230	Exposure of Sensitive Information Through Metadata	699	2006

Category-1213: Random Number Issues

Category ID : 1213

Summary

Weaknesses in this category are related to a software system's random number generation.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	331	Insufficient Entropy	699	821
HasMember	B	334	Small Space of Random Values	699	827
HasMember	B	335	Incorrect Usage of Seeds in Pseudo-Random Number Generator (PRNG)	699	829
HasMember	B	338	Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)	699	837
HasMember	B	341	Predictable from Observable State	699	843
HasMember	B	342	Predictable Exact Value from Previous Values	699	845
HasMember	B	343	Predictable Value Range from Previous Values	699	847
HasMember	B	344	Use of Invariant Value in Dynamically Changing Context	699	849
HasMember	B	1241	Use of Predictable Algorithm in Random Number Generator	699	2030

Category-1214: Data Integrity Issues

Category ID : 1214

Summary

Weaknesses in this category are related to a software system's data integrity components. Frequently these deal with the ability to ensure the integrity of data, such as messages, resource files, deployment files, and configuration files. The weaknesses in this category could lead to a degradation of data integrity quality if they are not addressed.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	322	Key Exchange without Entity Authentication	699	788
HasMember	C	346	Origin Validation Error	699	853
HasMember	B	347	Improper Verification of Cryptographic Signature	699	857
HasMember	B	348	Use of Less Trusted Source	699	859
HasMember	B	349	Acceptance of Extraneous Untrusted Data With Trusted Data	699	861
HasMember	B	351	Insufficient Type Distinction	699	866
HasMember	B	353	Missing Support for Integrity Check	699	874
HasMember	B	354	Improper Validation of Integrity Check Value	699	876
HasMember	B	494	Download of Code Without Integrity Check	699	1185
HasMember	B	565	Reliance on Cookies without Validation and Integrity Checking	699	1283
HasMember	B	649	Reliance on Obfuscation or Encryption of Security-Relevant Inputs without Integrity Checking	699	1430
HasMember	B	829	Inclusion of Functionality from Untrusted Control Sphere	699	1741

Nature	Type	ID	Name	V	Page
HasMember	B	924	Improper Enforcement of Message Integrity During Transmission in a Communication Channel	699	1830

Category-1215: Data Validation Issues

Category ID : 1215

Summary

Weaknesses in this category are related to a software system's components for input validation, output validation, or other kinds of validation. Validation is a frequently-used technique for ensuring that data conforms to expectations before it is further processed as input or output. There are many varieties of validation (see CWE-20, which is just for input validation). Validation is distinct from other techniques that attempt to modify data before processing it, although developers may consider all attempts to product "safe" inputs or outputs as some kind of validation. Regardless, validation is a powerful tool that is often used to minimize malformed data from entering the system, or indirectly avoid code injection or other potentially-malicious patterns when generating output. The weaknesses in this category could lead to a degradation of the quality of data flow in a system if they are not addressed.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	112	Missing XML Validation	699	269
HasMember	B	179	Incorrect Behavior Order: Early Validation	699	448
HasMember	B	183	Permissive List of Allowed Inputs	699	458
HasMember	B	184	Incomplete List of Disallowed Inputs	699	459
HasMember	B	606	Unchecked Input for Loop Condition	699	1357
HasMember	B	641	Improper Restriction of Names for Files and Other Resources	699	1412
HasMember	B	1173	Improper Use of Validation Framework	699	1969
HasMember	B	1284	Improper Validation of Specified Quantity in Input	699	2130
HasMember	B	1285	Improper Validation of Specified Index, Position, or Offset in Input	699	2132
HasMember	B	1286	Improper Validation of Syntactic Correctness of Input	699	2136
HasMember	B	1287	Improper Validation of Specified Type of Input	699	2138
HasMember	B	1288	Improper Validation of Consistency within Input	699	2139
HasMember	B	1289	Improper Validation of Unsafe Equivalence in Input	699	2141

Notes

Relationship

CWE-20 (Improper Input Validation) is not included in this category because it is a Class level, and this category focuses more on Base level weaknesses. Also note that other kinds of weaknesses besides improper validation are included as members of this category.

Category-1216: Lockout Mechanism Errors

Category ID : 1216

Summary

Weaknesses in this category are related to a software system's lockout mechanism. Frequently these deal with scenarios that take effect in case of multiple failed attempts to access a given resource. The weaknesses in this category could lead to a degradation of access to system assets if they are not addressed.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
MemberOf	C	1353	OWASP Top Ten 2021 Category A07:2021 - Identification and Authentication Failures	1344	2494
HasMember	B	645	Overly Restrictive Account Lockout Mechanism	699	1423

Category-1217: User Session Errors

Category ID : 1217

Summary

Weaknesses in this category are related to session management. Frequently these deal with the information or status about each user and their access rights for the duration of multiple requests. The weaknesses in this category could lead to a degradation of the quality of session management if they are not addressed.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	488	Exposure of Data Element to Wrong Session	699	1169
HasMember	B	613	Insufficient Session Expiration	699	1371
HasMember	B	841	Improper Enforcement of Behavioral Workflow	699	1772

Category-1218: Memory Buffer Errors

Category ID : 1218

Summary

Weaknesses in this category are related to the handling of memory buffers within a software system.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	699	304
HasMember	B	124	Buffer Underwrite ('Buffer Underflow')	699	326
HasMember	B	125	Out-of-bounds Read	699	330
HasMember	B	131	Incorrect Calculation of Buffer Size	699	355
HasMember	B	786	Access of Memory Location Before Start of Buffer	699	1658
HasMember	B	787	Out-of-bounds Write	699	1661
HasMember	B	788	Access of Memory Location After End of Buffer	699	1669
HasMember	B	805	Buffer Access with Incorrect Length Value	699	1702

Nature	Type	ID	Name	V	Page
HasMember	B	1284	Improper Validation of Specified Quantity in Input	699	2130

Category-1219: File Handling Issues

Category ID : 1219

Summary

Weaknesses in this category are related to the handling of files within a software system. Files, directories, and folders are so central to information technology that many different weaknesses and variants have been discovered.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	699	33
HasMember	B	41	Improper Resolution of Path Equivalence	699	86
HasMember	B	59	Improper Link Resolution Before File Access ('Link Following')	699	111
HasMember	B	66	Improper Handling of File Names that Identify Virtual Resources	699	124
HasMember	B	378	Creation of Temporary File With Insecure Permissions	699	928
HasMember	B	379	Creation of Temporary File in Directory with Insecure Permissions	699	930
HasMember	B	426	Untrusted Search Path	699	1028
HasMember	B	427	Uncontrolled Search Path Element	699	1033
HasMember	B	428	Unquoted Search Path or Element	699	1039

Category-1225: Documentation Issues

Category ID : 1225

Summary

Weaknesses in this category are related to the documentation provided to support, create, or analyze a product.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	1053	Missing Documentation for Design	699	1888
HasMember	B	1068	Inconsistency Between Implementation and Documented Design	699	1906
HasMember	B	1110	Incomplete Design Documentation	699	1950
HasMember	B	1111	Incomplete I/O Documentation	699	1951
HasMember	B	1112	Incomplete Documentation of Program Execution	699	1952
HasMember	B	1118	Insufficient Documentation of Error Handling Techniques	699	1958

Category-1226: Complexity Issues

Category ID : 1226

Summary

Weaknesses in this category are associated with things being overly complex.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	1043	Data Element Aggregating an Excessively Large Number of Non-Primitive Elements	699	1877
HasMember	B	1047	Modules with Circular Dependencies	699	1882
HasMember	B	1055	Multiple Inheritance from Concrete Classes	699	1890
HasMember	B	1056	Invokable Control Element with Variadic Parameters	699	1891
HasMember	B	1060	Excessive Number of Inefficient Server-Side Data Accesses	699	1897
HasMember	B	1064	Invokable Control Element with Signature Containing an Excessive Number of Parameters	699	1902
HasMember	B	1074	Class with Excessively Deep Inheritance	699	1914
HasMember	B	1075	Unconditional Control Flow Transfer outside of Switch Block	699	1915
HasMember	B	1080	Source Code File with Excessive Number of Lines of Code	699	1920
HasMember	B	1086	Class with Excessive Number of Child Classes	699	1926
HasMember	B	1095	Loop Condition Value Update within the Loop	699	1935
HasMember	B	1119	Excessive Use of Unconditional Branching	699	1959
HasMember	B	1121	Excessive McCabe Cyclomatic Complexity	699	1961
HasMember	B	1122	Excessive Halstead Complexity	699	1962
HasMember	B	1123	Excessive Use of Self-Modifying Code	699	1963
HasMember	B	1124	Excessively Deep Nesting	699	1964
HasMember	B	1125	Excessive Attack Surface	699	1965
HasMember	B	1333	Inefficient Regular Expression Complexity	699	2230

Category-1227: Encapsulation Issues

Category ID : 1227

Summary

Weaknesses in this category are related to issues surrounding the bundling of data with the methods intended to operate on that data.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	1054	Invocation of a Control Element at an Unnecessarily Deep Horizontal Layer	699	1889
HasMember	B	1057	Data Access Operations Outside of Expected Data Manager Component	699	1892
HasMember	B	1062	Parent Class with References to Child Class	699	1900

Nature	Type	ID	Name	V	Page
HasMember	B	1083	Data Access from Outside Expected Data Manager Component	699	1922
HasMember	B	1090	Method Containing Access of a Member Element from Another Class	699	1930
HasMember	B	1100	Insufficient Isolation of System-Dependent Functions	699	1940
HasMember	B	1105	Insufficient Encapsulation of Machine-Dependent Functionality	699	1945

Category-1228: API / Function Errors

Category ID : 1228

Summary

Weaknesses in this category are related to the use of built-in functions or external APIs.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	699	Software Development	699	2555
HasMember	B	242	Use of Inherently Dangerous Function	699	586
HasMember	B	474	Use of Function with Inconsistent Implementations	699	1128
HasMember	B	475	Undefined Behavior for Input to API	699	1130
HasMember	B	477	Use of Obsolete Function	699	1138
HasMember	B	676	Use of Potentially Dangerous Function	699	1489
HasMember	B	695	Use of Low-Level Functionality	699	1524
HasMember	B	749	Exposed Dangerous Method or Function	699	1564

Category-1237: SFP Primary Cluster: Faulty Resource Release

Category ID : 1237

Summary

This category identifies Software Fault Patterns (SFPs) within the Faulty Resource Release cluster (SFP37).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	V	415	Double Free	888	1008
HasMember	V	762	Mismatched Memory Management Routines	888	1596
HasMember	B	763	Release of Invalid Pointer or Reference	888	1599

Category-1238: SFP Primary Cluster: Failure to Release Memory

Category ID : 1238

Summary

This category identifies Software Fault Patterns (SFPs) within the Failure to Release Memory cluster (SFP38).

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	888	Software Fault Pattern (SFP) Clusters	888	2571
HasMember	V	401	Missing Release of Memory after Effective Lifetime	888	973

Category-1306: CISQ Quality Measures - Reliability

Category ID : 1306

Summary

Weaknesses in this category are related to the CISQ Quality Measures for Reliability. Presence of these weaknesses could reduce the reliability of the software.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1305	CISQ Quality Measures (2020)	1305	2588
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	1305	293
HasMember	B	170	Improper Null Termination	1305	428
HasMember	B	252	Unchecked Return Value	1305	606
HasMember	B	390	Detection of Error Condition Without Action	1305	943
HasMember	B	394	Unexpected Status Code or Return Value	1305	955
HasMember	C	404	Improper Resource Shutdown or Release	1305	980
HasMember	C	424	Improper Protection of Alternate Path	1305	1023
HasMember	B	459	Incomplete Cleanup	1305	1099
HasMember	B	476	NULL Pointer Dereference	1305	1132
HasMember	B	480	Use of Incorrect Operator	1305	1150
HasMember	B	484	Omitted Break Statement in Switch	1305	1162
HasMember	B	562	Return of Stack Variable Address	1305	1278
HasMember	V	595	Comparison of Object References Instead of Object Contents	1305	1334
HasMember	C	662	Improper Synchronization	1305	1448
HasMember	C	665	Improper Initialization	1305	1456
HasMember	C	672	Operation on a Resource after Expiration or Release	1305	1479
HasMember	B	681	Incorrect Conversion between Numeric Types	1305	1495
HasMember	P	682	Incorrect Calculation	1305	1499
HasMember	P	703	Improper Check or Handling of Exceptional Conditions	1305	1535
HasMember	C	704	Incorrect Type Conversion or Cast	1305	1538
HasMember	C	758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior	1305	1582
HasMember	B	835	Loop with Unreachable Exit Condition ('Infinite Loop')	1305	1757
HasMember	B	908	Use of Uninitialized Resource	1305	1792
HasMember	B	1045	Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor	1305	1880
HasMember	B	1051	Initialization with Hard-Coded Network Resource Configuration Data	1305	1886
HasMember	B	1066	Missing Serialization Control Element	1305	1904

Nature	Type	ID	Name	V	Page
HasMember	B	1070	Serializable Data Element Containing non-Serializable Item Elements	1305	1909
HasMember	V	1077	Floating Point Comparison with Incorrect Operator	1305	1917
HasMember	B	1079	Parent Class without Virtual Destructor Method	1305	1919
HasMember	B	1082	Class Instance Self Destruction Control Element	1305	1921
HasMember	B	1083	Data Access from Outside Expected Data Manager Component	1305	1922
HasMember	B	1087	Class with Virtual Method without a Virtual Destructor	1305	1927
HasMember	B	1088	Synchronous Access of Remote Resource without Timeout	1305	1928
HasMember	B	1098	Data Element containing Pointer Item without Proper Copy Control Element	1305	1938

References

[REF-1133]Consortium for Information & Software Quality (CISQ). "Automated Source Code Quality Measures". 2020. < https://www.omg.org/spec/ASCQM/ >.

Category-1307: CISQ Quality Measures - Maintainability

Category ID : 1307

Summary

Weaknesses in this category are related to the CISQ Quality Measures for Maintainability. Presence of these weaknesses could reduce the maintainability of the software.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1305	CISQ Quality Measures (2020)	1305	2588
HasMember	C	407	Inefficient Algorithmic Complexity	1305	992
HasMember	B	478	Missing Default Case in Multiple Condition Expression	1305	1142
HasMember	B	480	Use of Incorrect Operator	1305	1150
HasMember	B	484	Omitted Break Statement in Switch	1305	1162
HasMember	B	561	Dead Code	1305	1275
HasMember	B	570	Expression is Always False	1305	1292
HasMember	B	571	Expression is Always True	1305	1295
HasMember	B	783	Operator Precedence Logic Error	1305	1650
HasMember	B	1041	Use of Redundant Code	1305	1875
HasMember	B	1045	Parent Class with a Virtual Destructor and a Child Class without a Virtual Destructor	1305	1880
HasMember	B	1047	Modules with Circular Dependencies	1305	1882
HasMember	B	1048	Invokable Control Element with Large Number of Outward Calls	1305	1883
HasMember	B	1051	Initialization with Hard-Coded Network Resource Configuration Data	1305	1886
HasMember	B	1052	Excessive Use of Hard-Coded Literals in Initialization	1305	1887
HasMember	B	1054	Invocation of a Control Element at an Unnecessarily Deep Horizontal Layer	1305	1889
HasMember	B	1055	Multiple Inheritance from Concrete Classes	1305	1890
HasMember	B	1062	Parent Class with References to Child Class	1305	1900

Nature	Type	ID	Name	V	Page
HasMember	B	1064	Invokable Control Element with Signature Containing an Excessive Number of Parameters	1305	1902
HasMember	B	1074	Class with Excessively Deep Inheritance	1305	1914
HasMember	B	1075	Unconditional Control Flow Transfer outside of Switch Block	1305	1915
HasMember	B	1079	Parent Class without Virtual Destructor Method	1305	1919
HasMember	B	1080	Source Code File with Excessive Number of Lines of Code	1305	1920
HasMember	B	1084	Invokable Control Element with Excessive File or Data Access Operations	1305	1924
HasMember	B	1085	Invokable Control Element with Excessive Volume of Commented-out Code	1305	1925
HasMember	B	1086	Class with Excessive Number of Child Classes	1305	1926
HasMember	B	1087	Class with Virtual Method without a Virtual Destructor	1305	1927
HasMember	B	1090	Method Containing Access of a Member Element from Another Class	1305	1930
HasMember	B	1095	Loop Condition Value Update within the Loop	1305	1935

References

[REF-1133]Consortium for Information & Software Quality (CISQ). "Automated Source Code Quality Measures". 2020. < https://www.omg.org/spec/ASCQM/ >.

Category-1308: CISQ Quality Measures - Security

Category ID : 1308

Summary

Weaknesses in this category are related to the CISQ Quality Measures for Security. Presence of these weaknesses could reduce the security of the software.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1305	CISQ Quality Measures (2020)	1305	2588
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	1305	33
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	1305	145
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	1305	163
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	1305	201
HasMember	B	90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	1305	212
HasMember	B	91	XML Injection (aka Blind XPath Injection)	1305	215
HasMember	C	99	Improper Control of Resource Identifiers ('Resource Injection')	1305	243
HasMember	C	119	Improper Restriction of Operations within the Bounds of a Memory Buffer	1305	293
HasMember	V	129	Improper Validation of Array Index	1305	341
HasMember	B	134	Use of Externally-Controlled Format String	1305	365
HasMember	B	252	Unchecked Return Value	1305	606

Nature	Type	ID	Name	V	Page
HasMember	C	404	Improper Resource Shutdown or Release	1305	980
HasMember	C	424	Improper Protection of Alternate Path	1305	1023
HasMember	B	434	Unrestricted Upload of File with Dangerous Type	1305	1048
HasMember	B	477	Use of Obsolete Function	1305	1138
HasMember	B	480	Use of Incorrect Operator	1305	1150
HasMember	B	502	Deserialization of Untrusted Data	1305	1204
HasMember	B	570	Expression is Always False	1305	1292
HasMember	B	571	Expression is Always True	1305	1295
HasMember	B	606	Unchecked Input for Loop Condition	1305	1357
HasMember	B	611	Improper Restriction of XML External Entity Reference	1305	1367
HasMember	B	643	Improper Neutralization of Data within XPath Expressions ('XPath Injection')	1305	1419
HasMember	B	652	Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')	1305	1435
HasMember	C	662	Improper Synchronization	1305	1448
HasMember	C	665	Improper Initialization	1305	1456
HasMember	C	672	Operation on a Resource after Expiration or Release	1305	1479
HasMember	B	681	Incorrect Conversion between Numeric Types	1305	1495
HasMember	P	682	Incorrect Calculation	1305	1499
HasMember	C	732	Incorrect Permission Assignment for Critical Resource	1305	1551
HasMember	B	778	Insufficient Logging	1305	1638
HasMember	B	783	Operator Precedence Logic Error	1305	1650
HasMember	V	789	Memory Allocation with Excessive Size Value	1305	1674
HasMember	B	798	Use of Hard-coded Credentials	1305	1690
HasMember	B	835	Loop with Unreachable Exit Condition ('Infinite Loop')	1305	1757

References

[REF-1133]Consortium for Information & Software Quality (CISQ). "Automated Source Code Quality Measures". 2020. < https://www.omg.org/spec/ASCQM/ >.

Category-1309: CISQ Quality Measures - Efficiency

Category ID : 1309

Summary

Weaknesses in this category are related to the CISQ Quality Measures for Efficiency. Presence of these weaknesses could reduce the efficiency of the software.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1305	CISQ Quality Measures (2020)	1305	2588
HasMember	C	404	Improper Resource Shutdown or Release	1305	980
HasMember	C	424	Improper Protection of Alternate Path	1305	1023
HasMember	V	1042	Static Member Data Element outside of a Singleton Class Element	1305	1876
HasMember	B	1043	Data Element Aggregating an Excessively Large Number of Non-Primitive Elements	1305	1877
HasMember	B	1046	Creation of Immutable Text Using String Concatenation	1305	1881

Nature	Type	ID	Name	V	Page
HasMember	B	1049	Excessive Data Query Operations in a Large Data Table	1305	1884
HasMember	B	1050	Excessive Platform Resource Consumption within a Loop	1305	1885
HasMember	B	1057	Data Access Operations Outside of Expected Data Manager Component	1305	1892
HasMember	B	1060	Excessive Number of Inefficient Server-Side Data Accesses	1305	1897
HasMember	B	1067	Excessive Execution of Sequential Searches of Data Resource	1305	1905
HasMember	B	1072	Data Resource Access without Use of Connection Pooling	1305	1912
HasMember	B	1073	Non-SQL Invokable Control Element with Excessive Number of Data Resource Accesses	1305	1913
HasMember	B	1089	Large Data Table with Excessive Number of Indices	1305	1929
HasMember	B	1091	Use of Object without Invoking Destructor Method	1305	1931
HasMember	B	1094	Excessive Index Range Scan for a Data Resource	1305	1934

References

[REF-1133]Consortium for Information & Software Quality (CISQ). "Automated Source Code Quality Measures". 2020. < https://www.omg.org/spec/ASCQM/ >.

Category-1345: OWASP Top Ten 2021 Category A01:2021 - Broken Access Control

Category ID : 1345

Summary

Weaknesses in this category are related to the A01 category "Broken Access Control" in the OWASP Top Ten 2021.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1344	Weaknesses in OWASP Top Ten (2021)	1344	2593
HasMember	B	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	1344	33
HasMember	B	23	Relative Path Traversal	1344	46
HasMember	V	35	Path Traversal: '.../...//'	1344	73
HasMember	B	59	Improper Link Resolution Before File Access ('Link Following')	1344	111
HasMember	C	200	Exposure of Sensitive Information to an Unauthorized Actor	1344	504
HasMember	B	201	Insertion of Sensitive Information Into Sent Data	1344	514
HasMember	V	219	Storage of File with Sensitive Data Under Web Root	1344	553
HasMember	C	264	Permissions, Privileges, and Access Controls	1344	2316
HasMember	C	275	Permission Issues	1344	2317
HasMember	B	276	Incorrect Default Permissions	1344	665
HasMember	P	284	Improper Access Control	1344	680
HasMember	C	285	Improper Authorization	1344	684
HasMember		352	Cross-Site Request Forgery (CSRF)	1344	868

Nature	Type	ID	Name	V	Page
HasMember	B	359	Exposure of Private Personal Information to an Unauthorized Actor	1344	882
HasMember	C	377	Insecure Temporary File	1344	925
HasMember	C	402	Transmission of Private Resources into a New Sphere ('Resource Leak')	1344	976
HasMember	B	425	Direct Request ('Forced Browsing')	1344	1025
HasMember	C	441	Unintended Proxy or Intermediary ('Confused Deputy')	1344	1064
HasMember	B	497	Exposure of Sensitive System Information to an Unauthorized Control Sphere	1344	1193
HasMember	B	538	Insertion of Sensitive Information into Externally-Accessible File or Directory	1344	1248
HasMember	B	540	Inclusion of Sensitive Information in Source Code	1344	1251
HasMember	V	548	Exposure of Information Through Directory Listing	1344	1261
HasMember	B	552	Files or Directories Accessible to External Parties	1344	1265
HasMember	V	566	Authorization Bypass Through User-Controlled SQL Primary Key	1344	1286
HasMember	B	601	URL Redirection to Untrusted Site ('Open Redirect')	1344	1345
HasMember	B	639	Authorization Bypass Through User-Controlled Key	1344	1406
HasMember	V	651	Exposure of WSDL File Containing Sensitive Information	1344	1433
HasMember	C	668	Exposure of Resource to Wrong Sphere	1344	1469
HasMember	C	706	Use of Incorrectly-Resolved Name or Reference	1344	1544
HasMember	C	862	Missing Authorization	1344	1780
HasMember	C	863	Incorrect Authorization	1344	1787
HasMember	C	913	Improper Control of Dynamically-Managed Code Resources	1344	1805
HasMember	C	922	Insecure Storage of Sensitive Information	1344	1825
HasMember	V	1275	Sensitive Cookie with Improper SameSite Attribute	1344	2110

Notes

Maintenance

As of CWE 4.6, the relationships in this category were pulled directly from the CWE mappings cited in the 2021 OWASP Top Ten. These mappings include categories, which are discouraged for mapping, as well as high-level weaknesses such as Pillars. The CWE Program will work with OWASP to improve these mappings, possibly requiring modifications to CWE itself.

References

[REF-1207]"A01:2021 - Broken Access Control". 2021 September 4. OWASP. < https://owasp.org/Top10/A01_2021-Broken_Access_Control/ >.

[REF-1206]"OWASP Top 10:2021". 2021 September 4. OWASP. < https://owasp.org/Top10/ >.

Category-1346: OWASP Top Ten 2021 Category A02:2021 - Cryptographic Failures

Category ID : 1346

Summary

Weaknesses in this category are related to the A02 category "Cryptographic Failures" in the OWASP Top Ten 2021.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1344	Weaknesses in OWASP Top Ten (2021)	1344	2593
HasMember	B	261	Weak Encoding for Password	1344	631
HasMember	B	296	Improper Following of a Certificate's Chain of Trust	1344	719
HasMember	C	310	Cryptographic Issues	1344	2318
HasMember	B	319	Cleartext Transmission of Sensitive Information	1344	779
HasMember	V	321	Use of Hard-coded Cryptographic Key	1344	785
HasMember	B	322	Key Exchange without Entity Authentication	1344	788
HasMember	B	323	Reusing a Nonce, Key Pair in Encryption	1344	790
HasMember	B	324	Use of a Key Past its Expiration Date	1344	792
HasMember	B	325	Missing Cryptographic Step	1344	794
HasMember	C	326	Inadequate Encryption Strength	1344	796
HasMember	C	327	Use of a Broken or Risky Cryptographic Algorithm	1344	799
HasMember	B	328	Use of Weak Hash	1344	806
HasMember	V	329	Generation of Predictable IV with CBC Mode	1344	811
HasMember	C	330	Use of Insufficiently Random Values	1344	814
HasMember	B	331	Insufficient Entropy	1344	821
HasMember	B	335	Incorrect Usage of Seeds in Pseudo-Random Number Generator (PRNG)	1344	829
HasMember	V	336	Same Seed in Pseudo-Random Number Generator (PRNG)	1344	832
HasMember	V	337	Predictable Seed in Pseudo-Random Number Generator (PRNG)	1344	834
HasMember	B	338	Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)	1344	837
HasMember	C	340	Generation of Predictable Numbers or Identifiers	1344	842
HasMember	B	347	Improper Verification of Cryptographic Signature	1344	857
HasMember	B	523	Unprotected Transport of Credentials	1344	1230
HasMember	C	720	OWASP Top Ten 2007 Category A9 - Insecure Communications	1344	2333
HasMember	B	757	Selection of Less-Secure Algorithm During Negotiation ('Algorithm Downgrade')	1344	1581
HasMember	V	759	Use of a One-Way Hash without a Salt	1344	1585
HasMember	V	760	Use of a One-Way Hash with a Predictable Salt	1344	1589
HasMember	V	780	Use of RSA Algorithm without OAEP	1344	1644
HasMember	C	818	OWASP Top Ten 2010 Category A9 - Insufficient Transport Layer Protection	1344	2359
HasMember	B	916	Use of Password Hash With Insufficient Computational Effort	1344	1813

Notes

Maintenance

As of CWE 4.6, the relationships in this category were pulled directly from the CWE mappings cited in the 2021 OWASP Top Ten. These mappings include categories, which are discouraged for mapping, as well as high-level weaknesses such as Pillars. The CWE Program will work with OWASP to improve these mappings, possibly requiring modifications to CWE itself.

References

[REF-1208]"A02:2021 - Cryptographic Failures". 2021 September 4. OWASP. < https://owasp.org/Top10/A02_2021-Cryptographic_Failures/ >.

[REF-1206]"OWASP Top 10:2021". 2021 September 4. OWASP. < https://owasp.org/Top10/ >.

Category-1347: OWASP Top Ten 2021 Category A03:2021 - Injection

Category ID : 1347

Summary

Weaknesses in this category are related to the A03 category "Injection" in the OWASP Top Ten 2021.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1344	Weaknesses in OWASP Top Ten (2021)	1344	2593
HasMember	C	20	Improper Input Validation	1344	20
HasMember	C	74	Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')	1344	137
HasMember	C	75	Failure to Sanitize Special Elements into a Different Plane (Special Element Injection)	1344	142
HasMember	C	77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	1344	145
HasMember	B	78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	1344	151
HasMember	B	79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	1344	163
HasMember	V	80	Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)	1344	177
HasMember	V	83	Improper Neutralization of Script in Attributes in a Web Page	1344	183
HasMember	V	87	Improper Neutralization of Alternate XSS Syntax	1344	192
HasMember	B	88	Improper Neutralization of Argument Delimiters in a Command ('Argument Injection')	1344	194
HasMember	B	89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	1344	201
HasMember	B	90	Improper Neutralization of Special Elements used in an LDAP Query ('LDAP Injection')	1344	212
HasMember	B	91	XML Injection (aka Blind XPath Injection)	1344	215
HasMember	B	93	Improper Neutralization of CRLF Sequences ('CRLF Injection')	1344	217
HasMember	B	94	Improper Control of Generation of Code ('Code Injection')	1344	219
HasMember	V	95	Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')	1344	226
HasMember	B	96	Improper Neutralization of Directives in Statically Saved Code ('Static Code Injection')	1344	232
HasMember	V	97	Improper Neutralization of Server-Side Includes (SSI) Within a Web Page	1344	235
HasMember	V	98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')	1344	236
HasMember	C	99	Improper Control of Resource Identifiers ('Resource Injection')	1344	243

Nature	Type	ID	Name	V	Page
HasMember	V	113	Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Request/Response Splitting')	1344	271
HasMember	C	116	Improper Encoding or Escaping of Output	1344	281
HasMember	C	138	Improper Neutralization of Special Elements	1344	373
HasMember	B	184	Incomplete List of Disallowed Inputs	1344	459
HasMember	B	470	Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')	1344	1118
HasMember	B	471	Modification of Assumed-Immutable Data (MAID)	1344	1121
HasMember	V	564	SQL Injection: Hibernate	1344	1282
HasMember	C	610	Externally Controlled Reference to a Resource in Another Sphere	1344	1364
HasMember	B	643	Improper Neutralization of Data within XPath Expressions ('XPath Injection')	1344	1419
HasMember	V	644	Improper Neutralization of HTTP Headers for Scripting Syntax	1344	1422
HasMember	B	652	Improper Neutralization of Data within XQuery Expressions ('XQuery Injection')	1344	1435
HasMember	B	917	Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection')	1344	1818

Notes

Maintenance

As of CWE 4.6, the relationships in this category were pulled directly from the CWE mappings cited in the 2021 OWASP Top Ten. These mappings include high-level Class and/or Pillar weaknesses. The CWE Program will work with OWASP to improve these mappings, possibly including modifications to CWE itself.

References

[REF-1209]"A03:2021 - Injection". 2021 September 4. OWASP. < https://owasp.org/Top10/A03_2021-Injection/ >.

[REF-1206]"OWASP Top 10:2021". 2021 September 4. OWASP. < https://owasp.org/Top10/ >.

Category-1348: OWASP Top Ten 2021 Category A04:2021 - Insecure Design

Category ID : 1348

Summary

Weaknesses in this category are related to the A04 "Insecure Design" category in the OWASP Top Ten 2021.

Membership

Nature	Type	ID	Name	V	Page
MemberOf	V	1344	Weaknesses in OWASP Top Ten (2021)	1344	2593
HasMember	B	73	External Control of File Name or Path	1344	132
HasMember	B	183	Permissive List of Allowed Inputs	1344	458
HasMember	B	209	Generation of Error Message Containing Sensitive Information	1344	533
HasMember	B	213	Exposure of Sensitive Information Due to Incompatible Policies	1344	547

CWE-1348: OWASP Top Ten 2021 Category A04:2021 - Insecure Design

Nature	Type	ID	Name	V	Page
HasMember	V	235	Improper Handling of Extra Parameters	1344	578
HasMember	B	256	Plaintext Storage of a Password	1344	615
HasMember	B	257	Storing Passwords in a Recoverable Format	1344	618
HasMember	B	266	Incorrect Privilege Assignment	1344	638
HasMember	C	269	Improper Privilege Management	1344	646
HasMember	B	280	Improper Handling of Insufficient Permissions or Privileges	1344	672
HasMember	C	311	Missing Encryption of Sensitive Data	1344	757
HasMember	B	312	Cleartext Storage of Sensitive Information	1344	764
HasMember	V	313	Cleartext Storage in a File or on Disk	1344	770
HasMember	V	316	Cleartext Storage of Sensitive Information in Memory	1344	775
HasMember	B	419	Unprotected Primary Channel	1344	1017
HasMember	B	430	Deployment of Wrong Handler	1344	1042
HasMember	B	434	Unrestricted Upload of File with Dangerous Type	1344	1048
HasMember	B	444	Inconsistent Interpretation of HTTP Requests ('HTTP Request/Response Smuggling')	1344	1068
HasMember	C	451	User Interface (UI) Misrepresentation of Critical Information	1344	1079
HasMember	B	472	External Control of Assumed-Immutable Web Parameter	1344	1123
HasMember	B	501	Trust Boundary Violation	1344	1203
HasMember	C	522	Insufficiently Protected Credentials	1344	1225
HasMember	V	525	Use of Web Browser Cache Containing Sensitive Information	1344	1233
HasMember	V	539	Use of Persistent Cookies Containing Sensitive Information	1344	1250
HasMember	V	579	J2EE Bad Practices: Non-serializable Object Stored in Session	1344	1309
HasMember	V	598	Use of GET Request Method With Sensitive Query Strings	1344	1340
HasMember	C	602	Client-Side Enforcement of Server-Side Security	1344	1350
HasMember	C	642	External Control of Critical State Data	1344	1414
HasMember	V	646	Reliance on File Name or Extension of Externally-Supplied File	1344	1425
HasMember	V	650	Trusting HTTP Permission Methods on the Server Side	1344	1432
HasMember	C	653	Improper Isolation or Compartmentalization	1344	1437
HasMember	C	656	Reliance on Security Through Obscurity	1344	1444
HasMember	C	657	Violation of Secure Design Principles	1344	1446
HasMember	C	799	Improper Control of Interaction Frequency	1344	1699
HasMember	B	807	Reliance on Untrusted Inputs in a Security Decision	1344	1714
HasMember	C	840	Business Logic Errors	1344	2360
HasMember	B	841	Improper Enforcement of Behavioral Workflow	1344	1772
HasMember	V	927	Use of Implicit Intent for Sensitive Communication	1344	1836
HasMember	B	1021	Improper Restriction of Rendered UI Layers or Frames	1344	1860
HasMember	B	1173	Improper Use of Validation Framework	1344	1969

Notes

Maintenance

As of CWE 4.6, the relationships in this category were pulled directly from the CWE mappings cited in the 2021 OWASP Top Ten. These mappings include categories, which are discouraged