Following are some CWE nodes described in terms of vulnerability theory.

Each description has one status:

Stable - the CWE team has produced a definition that should be considered for review by the public.

Draft - the CWE team has an initial definition, but it needs additional work.

Incomplete - the CWE team has not investigated this node closely.

==========

80 - Basic XSS

A behavior supports the mixing of data and directives in a single stream using special elements to separate them.

A Data/Directive Boundary Error occurs when data contains special elements that cause portions to be inadvertently interpreted as directives, or vice versa.

A Restricted Directive Policy is a policy in which a user is only allowed to provide a subset of directives.

A Directive Injection vulnerability occurs when an attacker can exploit a Data/Directive Boundary Error to introduce directives that violate the intended Restrictive Directive Policy into a stream.

Basic XSS occurs when the data stream is HTML and when the intended Restricted Directive Policy prohibits any directives related to scripting, the intended policy allows an actor (such as a user or outsider) to introduce data into a stream that is processed by a user, but there is no protection scheme to prevent a Data/Directive Boundary Error.

Note: is Argument Injection covered under Directive Injection? Should it be?

==========

99 - Resource Injection

An intended policy defines a limited set of resources and a limited set of behaviors that manipulate those resources.

A separate behavior allows the user to specify an identifier for a resource that the product manipulates.

Resource injection occurs when the user is able to specify an identifier for a resource that violates the intended policy.

Note: this might need to give more specific examples for KDM such as types of resources and variations in the intended policy, such as file names within a particular subdirectory or by a particular owner

==========

78 - OS Command Injection

A behavior supports the mixing of data and directives in a single stream using special elements to separate them.

A Data/Directive Boundary Error occurs when data contains special elements that cause portions to be inadvertently interpreted as directives, or vice versa.

A Restricted Directive Policy is a policy in which a user is only allowed to provide a subset of directives.

A Directive Injection vulnerability occurs when an attacker can exploit a Data/Directive Boundary Error to introduce directives that violate the intended Restrictive Directive Policy into a stream.

A product has a behavior that executes OS directives using a mixed data/directive stream, let's call it OSDDS. An actor can control or influence a portion of the data in OSDDS. The Restricted Directive Policy prohibits the actor from specifying any directives in OSDDS.

OS Command Injection occurs when an actor can provide data into OSDDS where the data contains a directive separator causing a Data/Directive Boundary Error.

NOTE: Some interpretatoins of OS command injection allow for the ATTACKER to modify the intended DIRECTIVE.
NOTE: Do we mean just "user" or the more general "actor" for a restrictive directive policy as well as product behaviors?

==========

89 - SQL Injection

A behavior supports the mixing of data and directives in a single stream using special elements to separate them.

A Data/Directive Boundary Error occurs when data contains special elements that cause portions to be inadvertently interpreted as directives, or vice versa.

A Restricted Directive Policy is a policy in which a user is only allowed to provide a subset of directives.

A Directive Injection vulnerability occurs when an attacker can exploit a Data/Directive Boundary Error to introduce directives that violate the intended Restrictive Directive Policy into a stream.

A product has a behavior that executes SQL queries using a mixed data/directive stream, let's call it QUERY. An actor can control or influence a portion of the data in QUERY. The Restricted Directive Policy prohibits the actor from modifying any directives or clauses in QUERY.

SQL Injection occurs when an actor can provide data into QUERY where the data contains a directive or clause separator causing a Data/Directive Boundary Error that violates the Restricted Directive Policy.

Note: We haven't quite covered the addition of new directives as well as clauses (' OR 1=1')

==========

22 - Path Traversal

A CONTAINER is a subset of all resources accessible to the product that should be accessible or mutable to an actor.

A product's behavior constructs a filename that is at least partially based on user input.

The product’s behavior performs directives on a file or directory resource on behalf of an actor. It uses a container to identify valid resources for that actor. The product also has a behavior that constructs a filename for the desired resource in which the filename is at least partially based on user input. The filename is valid if it identifies a resource within the container.

The intended policy only allows valid resources to be identified.

A Path Traversal Vulnerability occurs when the product does not enforce the validity property for a constructed filename.

Note: A product is not limited to user-level applications. For example, a web server only has access to a subset of the OS resources and thus is restricted to a container with respect to the OS.

Terminology Issue: Need a parent term implying both accessibility and mutability.

Draft Notes: A product's behavior constructs a filename that is at least partially based on user input. The intended policy is that a filename is VALID only if it is CONTAINED under a particular directory or set of directories. A path traversal vulnerability occurs if a user can use SYNTACTIC MANIPULATIONS (such as ".." or "/a/b") that cause the generated filename to violate the CONTAINMENT property.

==========

59 - Link Following

A CONTAINER is a subset of all resources accessible to the product that should be accessible or mutable to an actor.

The product performs manipulations on a file or directory resource that is within some container. The product constructs a pathname for that resource. A generated pathname is valid only if it identifies a resource within the container.

The intended policy requires that only valid pathnames be processed.

A Symbolic Link Following Vulnerability occurs when the product generates a pathname that is associated with a symbolic link resource without ensuring that the link resolves to an identifier for a resource within the container.

Draft Notes:

Currently only talking about symbolic link following, further investigation might allow this to be adapted to cover hard links as well as windows .lnk but do these also fall under the same identifier and reference resolution problems?

Does POLICY include the definitions of what the PROPERTIES are as well as which PROPERTIES are allowed, or is it only which PROPERTIES are allowed?

This is an instantiation of a general case where you have identifiers to references to resources instead of identifiers to resources directly.

can borrow from path traversal, except SEMANTIC manipulations are used to violate IMMUTABILITY.

==========

120 - Unbounded Transfer

A behavior receives an input buffer resource on which it performs a manipulation, possibly the identity manipulation, which is then stored in an output buffer resource. The intended policy defines a valid output buffer resource to be large enough to store the result of the manipulation. A Buffer Overflow vulnerability occurs when an attacker can influence behavior such that this validity property is violated.

An Unbounded Transfer is a Buffer Overflow which only involves the identity manipulation, the attacker does not have control over the size of the output buffer resource, and there is no protection scheme that ensures the validity property of the output buffer resource holds.

NOTE: some people may include expansion overflows in "classic overflows"

==========

190 - Integer Overflow

A behavior, B, uses an integer to control or influence operation on a resource.

A separate behavior, G, generates this integer using some manipulation.

T_MAX is defined as the maximum allowable value for an integer given the product’s representation of integers.

An integer overflow occurs when a manipulation would produce a value that exceeds T_MAX, but the behavior returns a result that is less than or equal to T_MAX.

The intended policy is that behavior B does not use the result from behavior G if G produces an integer overflow.

An Integer Overflow Vulnerability occurs when behavior G produces an integer overflow, and behavior B uses the result.

Draft Notes:
The intended policy effectively excludes products which use integer overflows as part of a valid behavior.

Might want to express this in terms of representation and equivalence;
Representation in the product and equivalence to the real world result.

==========

134 - Format string vulnerability

A product has a behavior that formats output to either a MEMORY or STREAM resource.

The intended policy is that the user cannot control or influence the format string, i.e. the format string is IMMUTABLE by all actors except the product itself.

A Format String Vulnerability exists if an attacker can modify the format string which violates the intended policy

Note: This does not account for cases when the intended policy would allow user control of format strings to some degree, such as with internationalization.

A common consequence of a Format String Vulnerability is that properties of other resources could be violated such as immutability or inaccessibility of memory.

==========

170 - Improper Null Termination

The product's environment defines a buffer resource to be valid if it is null terminated.

The product has a behavior, M, that manipulates or creates a buffer resource. The product also has a separate behavior, U, that uses the buffer manipulated by M.

The intended policy is that the buffer resource must be valid after M has modified or created it. This creates an expectation for U that it will be using a valid buffer.

An Improper Null Termination Vulnerability occurs when an attacker can control or influence M, such that M either fails to write a null terminator or overwrites the null terminator with non-null data. This violates U's expectation that the buffer resource is valid.

Note: U is resultant, but it is also necessary for a vulnerability to exist.

Draft Notes:
Is "use" the parent of modify and access?

Might have similar distinction issues for actor vs. attacker and bug vs. vulnerability to TocTou.

==========

244 - Heap Inspection

SENSITIVE INFORMATION is information that should not be accessible to all actors.

AUTHORIZED actors for a given resource are the set of actors allowed to access that resource.

A behavior stores sensitive information in memory on the heap. Another behavior relinquishes the same resource back to the system, either directly using free(), or indirectly using realloc(), or similar functions.

A Heap Inspection Vulnerability exists when a buffer contains sensitive data when it is relinquished, and the buffer's relinquished memory can be accessed by an unauthorized actor.

==========

259 - Hard-Coded Password

The UNIVERSAL POLICY includes the following: only the administrator for a product, or an actor designated by the admin, can access and modify security features of the product.

The product provides a security feature (such as authentication or authorization) that limits which actors can access or modify the product's other features.

A Hard-Coded Password vulnerability occurs when the product has an implemented policy (possibly intended) in which a special user and password grants access to the product, but this user/password is not mutable by the administrator. This violates the universal policy.

==========

367 - Time-of-check Time-of-use race condition

The product has a behavior that performs a manipulation on a resource. The intended policy requires the resource to have a given property, P.

The product implements a protection scheme in which the behavior checks to make sure the resource has the property P before it attempts to manipulate the resource. The product assumes that P is immutable between the check and the manipulation.

A Time-of-check Time-of-use Race Condition Vulnerability occurs when an actor can directly or indirectly modify the assumed-immutable property, P, between the protection scheme check and the actual manipulation.

Note: This doesn't distinguish between bugs and vulnerabilities unless we change "actor" to "attacker" in the last paragraph. However, if the bug becomes a primary weakness, then the actor that caused the bug becomes an accomplice to the attacker that exploits the resultant vulnerability.

Additionally, unlike other issues, this weakness may need to formalize the concepts of time or multiple threads more completely.

==========

391 - Unchecked Error Condition

A product has a behavior, M, that manipulates a resource to achieve or enforce a required property, P. The manipulation can generate an error condition in which P is not present. A subsequent behavior, B, has expectations that P is present.

An Unchecked Error Condition occurs when an attacker can control or influence behavior M such that required property P is not present, and the product does not check the generated error condition that indicates that P is not present. This violates B's expectations and leads to resultant vulnerabilities.

Note: B is resultant, but also necessary for a vulnerability to exist.

==========

401 - Memory leak

The product has a behavior, B, that uses dynamically allocated memory that has no other uses outside of B.

The product has an intended policy in which a dynamically allocated memory resource should be relinquished soon after its last possible use.

A Memory Leak occurs when an attacker can cause B to complete without relinquishing that memory.

Draft Notes:

"no other uses outside of B" is needed so that, when B is complete, the memory should be relinquished. If there are other uses, then the memory would not need to be relinquished.

Consider a massive memleak that causes 50meg to be leaked in a single action (assume the original allocation was valid for a given behavior). Is this a vuln? Many might disagree Also – typical issue is in a behavior that's repeated over and over.

==========

412 - Unrestricted Critical Resource Lock

A PRODUCT has either an INCORRECT BEHAVIOR or false EXPECTATIONS and as a result an ATTACKER is not forced to relinquish a critical RESOURCE. There is either no POLICY in place that requires an ATTACKER to relinquish the RESOURCE or the POLICY exists but is not enforced.

==========

415 - Double Free

Draft Notes:
The intended policy is such that a resource or reference is not accessed or mutated after it has been released.

A valid resource is a resource that has not yet been released.

A Double Free vulnerability exists when an actor can invoke a behavior that will release an invalid resource.

free() is a mutation?

==========

416 - Use After Free

Draft Notes:
The intended policy is such that a resource or reference is not accessed or mutated after it has been released.

A valid resource is a resource that has not yet been released.

A Use After Free vulnerability exists when an actor can invoke a behavior that accesses or manipulates an invalid resource.

==========

457 - Uninitialized Variable

Draft Notes:
The intended policy is such that a resource is not accessed until it has been initialized / assigned a value by the product.

A valid resource is a resource that has been initialized / assigned a value.

An Uninitialized Variable Vulnerability occurs when an actor can invoke a behavior that accesses an invalid resource.

==========

466 - Illegal Pointer Value

My latest attempt:

An INCORRECT BEHAVIOR exists allowing the VALIDITY PROPERTY of a pointer RESOURCE to be violated.

My first attempt:
#(ptr assignment controlled by attacker)
#If a pointer RESOURCE can be MANIPULATED (semantic or syntactic?) to have the PROPERTY of pointing outside of the allocated #buffer RESOURCE, an ATTACKER can trigger unintended BEHAVIORS or other CONSEQUENCES.

#Alternate (bug, ptr assignment not controlled by attacker):

#If a BEHAVIOR violates the PRODUCTS POLICY by assigning a pointer RESOURCE to point outside of the allocated buffer #RESOURCE, any BEHAVIOR that gives an ATTACK access to that pointer RESOURCE may result in a Vulnerability.

==========

467 - Use of sizeof() on a pointer type

==========

468 - Unintentional pointer scaling

==========

469 - Improper pointer subtraction

==========

470 - Unsafe Reflection

==========

476 - Null Dereference

==========

495 - Private Array-Typed Field Returned From A Public Method

==========

496 - Public Data Assigned to Private Array-Typed Field

==========

489 - Leftover Debug Code

==========

121 - Stack Overflow

A behavior receives an input buffer resource on which it performs a manipulation, possibly the identity manipulation, which is then stored in an output buffer resource. The intended policy defines a valid output buffer resource to be large enough to store the result of the manipulation. A Buffer Overflow vulnerability occurs when an attacker can influence behavior such that this validity property is violated.

A Stack Overflow is a Buffer Overflow in which the output buffer resource is allocated on the stack.

+++OLD+++
A Buffer Overflow VULNERABILITY exists if a buffer RESOURCE has a VALIDITY PROPERTY stating that the contents of the buffer must be less than or equal to the allocated size of the buffer RESOURCE and a BEHAVIOR exists that violates said PROPERTY.

If the PRODUCT BEHAVES in a stack / heap environment (Is there a specific vuln theory term to refer to the products environment?) and the Buffer Overflow VULNERABILITY exists in a buffer RESOURCE that exists on the stack, then this VULNERABILITY is called a Stack Overflow.

==========

122 - Heap Overflow

A behavior receives an input buffer resource on which it performs a manipulation, possibly the identity manipulation, which is then stored in an output buffer resource. The intended policy defines a valid output buffer resource to be large enough to store the result of the manipulation. A Buffer Overflow vulnerability occurs when an attacker can influence behavior such that this validity property is violated.

A Heap Overflow is a Buffer Overflow in which the output buffer resource is allocated on the heap.

+++OLD+++
A Buffer Overflow VULNERABILITY exists if a buffer RESO- CVE-2006-4541 - DoS in IDS via NULL argumentURCE has a VALIDITY PROPERTY stating that the contents of the buffer must be less than or equal to the allocated size of the buffer RESOURCE and a BEHAVIOR exists that violates said PROPERTY.

If the PRODUCT BEHAVES in a stack / heap environment (Is there a specific vuln theory term to refer to the products environment?) and the Buffer Overflow VULNERABILITY exists in a buffer RESOURCE that exists on the heap, then this VULNERABILITY is called a
Heap Overflow.

==========

249 - Often Misused: Path Manipulation

A BEHAVIOR, in this case a Path Manipulation BEHAVIOR, receives an input buffer RESOURCE, in this case the path, on which it performs a MANIPULATION which is then stored in an output buffer RESOURCE. The PRODUCT should have a POLICY in place requiring the output buffer RESOURCE to have a VALIDITY PROPERTY such that the RESOURCE must be large enough to store the result of the MANIPULATION.

If the PRODUCT does not enforce this POLICY, an ATTACKER can provide a path that will be too large for the output buffer RESOURCE and may cause a buffer overflow ATTACK.

==========

425 - Direct Request

A product has behaviors that are separated into multiple executable programs. Intended policy defines a valid path in which the user can only access one program, X, through another program, P. P's behavior uses security features or protection schemes to restrict access to X. A "direct request" issues occurs if the user can directly access X without navigating through P first, violating execution-path validity and bypassing the security features.

could mention: alternate path, interaction points, accessibility.