- Open Vulnerability and Assessment Language - Element Dictionary

- Schema: UNIX Definition
- Version: 5.0 release candidate 3
- Release Date: 26 May 2006

The following is a description of the elements, types, and attributes that compose generic UNIX tests found in Open Vulnerability and Assessment Language (OVAL). Each test is an extension of the standard test element defined in the Core Definition Schema. Through extension, each test inherits a set of elements and attributes that are shared amongst all OVAL tests. Each test is described in detail and should provide the information necessary to understand what each element and attribute represents. This document is intended for developers and assumes some familiarity with XML. A high level description of the interaction between the different tests and their relationship to the Core Definition Schema is not outlined here.

The OVAL Schema is maintained by The Mitre Corporation and developed by the public OVAL Community. For more information, including how to get involved in the project and how to submit change requests, please visit the OVAL website at http://oval.mitre.org.

< file_test >

The file test is used to check metadata associated with UNIX files, of the sort returned by either an ls command, stat command or stat() system call. It extends the standard TestType as defined in the oval-definitions-schema and one should refer to the TestType description for more information. The required object element references a file_object and the optional state element specifies the metadata to check. The evaluation of the test is guided by the check attribute that is inherited from the TestType.

Child Elements	Type	MinOccurs	MaxOccurs
object	oval-def:ObjectRefType	1	1
state	oval-def:StateRefType	0	1

< file_object >

The file_object element is used by a file test to define the specific file(s) to be evaluated. Each object extends the standard ObjectType as definied in the oval-definitions-schema and one should refer to the ObjectType description for more information. The common set element allows complex objects to be created using filters and set logic. Again, please refer to the description of the set element in the oval-definitions-schema.

A file object defines the path and filename of the file(s). In addition, a number of behaviors may be provided that help guide the collection of objects. Please refer to the FileBehaviors complex type for more information about specific behaviors.

Child Elements	Type	MinOccurs	MaxOccurs
behaviors	unix-def:FileBehaviors	0	1
path	oval-def:EntityObjectStringType	1	1
filename	oval-def:EntityObjectStringType	1	1

< file_state >

The file_state element defines the different metadata associate with a UNIX file. This includes the path, filename, type, group id, user id, size, etc. In addition, the permission associated with the file are also included. Please refer to the individual elements in the schema for more details about what each represents.

Child Elements	Type	MinOccurs	MaxOccurs
path	oval-def:EntityStateStringType	0	1
filename	oval-def:EntityStateStringType	0	1
type	oval-def:EntityStateStringType	0	1
group_id	oval-def:EntityStateStringType	0	1
user_id	oval-def:EntityStateStringType	0	1
a_time	oval-def:EntityStateStringType	0	1
c_time	oval-def:EntityStateStringType	0	1
m_time	oval-def:EntityStateStringType	0	1
size	oval-def:EntityStateIntType	0	1
suid	oval-def:EntityStateBoolType	0	1
sgid	oval-def:EntityStateBoolType	0	1
sticky	oval-def:EntityStateBoolType	0	1
uread	oval-def:EntityStateBoolType	0	1
uwrite	oval-def:EntityStateBoolType	0	1
uexec	oval-def:EntityStateBoolType	0	1
gread	oval-def:EntityStateBoolType	0	1
gwrite	oval-def:EntityStateBoolType	0	1
gexec	oval-def:EntityStateBoolType	0	1
oread	oval-def:EntityStateBoolType	0	1
owrite	oval-def:EntityStateBoolType	0	1
oexec	oval-def:EntityStateBoolType	0	1

== FileBehaviors ==

These behaviors allow a more detailed definition of the file objects being specified.

Attributes:

- max_depth	n/a	(optional -- default='-1')
- recurse	n/a	(optional -- default='none')
- recurse_direction	n/a	(optional -- default='none')
- recurse_file_system	n/a	(optional -- default='all')

< inetd_test >

The inetd test is used to check information associated with different Internet services. It extends the standard TestType as defined in the oval-definitions-schema and one should refer to the TestType description for more information. The required object element references an inetd_object and the optional state element specifies the information to check. The evaluation of the test is guided by the check attribute that is inherited from the TestType.

Child Elements	Type	MinOccurs	MaxOccurs
object	oval-def:ObjectRefType	1	1
state	oval-def:StateRefType	0	1

< inetd_object >

The inetd_object element is used by an inetd test to define the specific protocol-service to be evaluated. Each object extends the standard ObjectType as definied in the oval-definitions-schema and one should refer to the ObjectType description for more information. The common set element allows complex objects to be created using filters and set logic. Again, please refer to the description of the set element in the oval-definitions-schema.

An inetd object consists of a single server_program entity that identifies the service to be used.

Child Elements	Type	MinOccurs	MaxOccurs
protocol	oval-def:EntityObjectStringType	1	1
service_name	oval-def:EntityObjectStringType	1	1

< inetd_state >

The inetd_state element defines the different information associated with a specific Internet service.

Please refer to the individual elements in the schema for more details about what each represents.

Child Elements	Type	MinOccurs	MaxOccurs
protocol	oval-def:EntityStateStringType	0	1
service_name	oval-def:EntityStateStringType	0	1
server_program	oval-def:EntityStateStringType	0	1
endpoint_type	unix-def:EntityStateEndpointType	0	1
exec_as_user	oval-def:EntityStateStringType	0	1
wait_status	unix-def:EntityStateWaitStatusType	0	1

< interface_test >

The interface test enumerate various attributes about the interfaces on a system. It extends the standard TestType as defined in the oval-definitions-schema and one should refer to the TestType description for more information. The required object element references an interface_object and the optional state element specifies the interface information to check. The evaluation of the test is guided by the check attribute that is inherited from the TestType.

Child Elements	Type	MinOccurs	MaxOccurs
object	oval-def:ObjectRefType	1	1
state	oval-def:StateRefType	0	1

< interface_object >

The interface_object element is used by an interface test to define the specific interfaces(s) to be evaluated. Each object extends the standard ObjectType as definied in the oval-definitions-schema and one should refer to the ObjectType description for more information. The common set element allows complex objects to be created using filters and set logic. Again, please refer to the description of the set element in the oval-definitions-schema.

An interface object consists of a single name entity that identifies which interface is being specified.

Child Elements	Type	MinOccurs	MaxOccurs
name	oval-def:EntityObjectStringType	1	1

< interface_state >

The interface_state element enumerates the different properties associate with a Unix interface. Please refer to the individual elements in the schema for more details about what each represents.

Child Elements	Type	MinOccurs	MaxOccurs
name	oval-def:EntityStateStringType	0	1
hardware_addr	oval-def:EntityStateStringType	0	1
inet_addr	oval-def:EntityStateStringType	0	1
broadcast_addr	oval-def:EntityStateStringType	0	1
netmask	oval-def:EntityStateStringType	0	1
flag	oval-def:EntityStateStringType	0	1

< password_test >

/etc/passwd. See passwd(4).

The password test is used to check metadata associated with the UNIX password file, of the sort returned by the passwd command. It extends the standard TestType as defined in the oval-definitions-schema and one should refer to the TestType description for more information. The required object element references a password_object and the optional state element specifies the metadata to check. The evaluation of the test is guided by the check attribute that is inherited from the TestType.

Child Elements	Type	MinOccurs	MaxOccurs
object	oval-def:ObjectRefType	1	1
state	oval-def:StateRefType	0	1

< password_object >

The password_object element is used by a password test to define the object to be evaluated. Each object extends the standard ObjectType as definied in the oval-definitions-schema and one should refer to the ObjectType description for more information. The common set element allows complex objects to be created using filters and set logic. Again, please refer to the description of the set element in the oval-definitions-schema.

A password object consists of a single username entity that identifies the user whos passwords are to be evaluated.

Child Elements	Type	MinOccurs	MaxOccurs
username	oval-def:EntityObjectStringType	1	1

< password_state >

The password_state element defines the different information associated with the system passwords. Please refer to the individual elements in the schema for more details about what each represents.

Child Elements	Type	MinOccurs	MaxOccurs
username	oval-def:EntityStateStringType	0	1
password	oval-def:EntityStateStringType	0	1
user_id	oval-def:EntityStateStringType	0	1
group_id	oval-def:EntityStateStringType	0	1
gcos	oval-def:EntityStateStringType	0	1
home_dir	oval-def:EntityStateStringType	0	1
login_shell	oval-def:EntityStateStringType	0	1

< process_test >

The process test is used to check information found in the UNIX processes. It is equivalent to parsing the output of the ps command. It extends the standard TestType as defined in the oval-definitions-schema and one should refer to the TestType description for more information. The required object element references a process_object and the optional state element specifies the process information to check. The evaluation of the test is guided by the check attribute that is inherited from the TestType.

Child Elements	Type	MinOccurs	MaxOccurs
object	oval-def:ObjectRefType	1	1
state	oval-def:StateRefType	0	1

< process_object >

The process_object element is used by a process test to define the specific process(es) to be evaluated. Each object extends the standard ObjectType as definied in the oval-definitions-schema and one should refer to the ObjectType description for more information. The common set element allows complex objects to be created using filters and set logic. Again, please refer to the description of the set element in the oval-definitions-schema.

A process object defines the command line used to start the process(s).

Child Elements	Type	MinOccurs	MaxOccurs

command	oval-def:EntityObjectStringType	1	1

< process_state >

The process_state element defines the different metadata associate with a UNIX process. This includes the command line, pid, ppid, priority, and user id. Please refer to the individual elements in the schema for more details about what each represents.

Child Elements	**Type**	**MinOccurs**	**MaxOccurs**
command	oval-def:EntityStateStringType	0	1
exec_time	oval-def:EntityStateStringType	0	1
pid	oval-def:EntityStateIntType	0	1
ppid	oval-def:EntityStateIntType	0	1
priority	oval-def:EntityStateStringType	0	1
scheduling_class	oval-def:EntityStateStringType	0	1
start_time	oval-def:EntityStateStringType	0	1
tty	oval-def:EntityStateStringType	0	1
user_id	oval-def:EntityStateStringType	0	1

< runlevel_test >

The runlevel test is used to check information about which runlevel specified service are scheduled to exist at. For more information see the output generated by a chkconfig --list. It extends the standard TestType as defined in the oval-definitions-schema and one should refer to the TestType description for more information. The required object element references a runlevel_object and the optional state element specifies the data to check. The evaluation of the test is guided by the check attribute that is inherited from the TestType.

Child Elements	**Type**	**MinOccurs**	**MaxOccurs**
object	oval-def:ObjectRefType	1	1
state	oval-def:StateRefType	0	1

< runlevel_object >

The runlevel_object element is used by a runlevel_test to define the specific service(s)/runlevel combination to be evaluated. Each object extends the standard ObjectType as definied in the oval-definitions-schema and one should refer to the ObjectType description for more information. The common set element allows complex objects to be created using filters and set logic. Again, please refer to the description of the set element in the oval-definitions-schema.

Child Elements	Type	MinOccurs	MaxOccurs
service_name	oval-def:EntityObjectStringType	1	1
runlevel	oval-def:EntityObjectStringType	1	1

< runlevel_state >

The runlevel_state element holds information about whether a specific service is schedule to start or kill at a given runlevel. Please refer to the individual elements in the schema for more details about what each represents.

Child Elements	Type	MinOccurs	MaxOccurs
service_name	oval-def:EntityStateStringType	0	1
runlevel	oval-def:EntityStateStringType	0	1
start	oval-def:EntityStateBoolType	0	1
kill	oval-def:EntityStateBoolType	0	1

< sccs_test >

Child Elements	Type	MinOccurs	MaxOccurs
object	oval-def:ObjectRefType	1	1
state	oval-def:StateRefType	0	1

< sccs_object >

Child Elements	Type	MinOccurs	MaxOccurs
path	oval-def:EntityObjectStringType	1	1
filename	oval-def:EntityObjectStringType	1	1

< sccs_state >

Child Elements	Type	MinOccurs	MaxOccurs

path	oval-def:EntityStateStringType	0	1
filename	oval-def:EntityStateStringType	0	1
module_name	oval-def:EntityStateStringType	0	1
module_type	oval-def:EntityStateStringType	0	1
release	oval-def:EntityStateStringType	0	1
level	oval-def:EntityStateStringType	0	1
branch	oval-def:EntityStateStringType	0	1
sequence	oval-def:EntityStateStringType	0	1
what_string	oval-def:EntityStateStringType	0	1

< shadow_test >

The shadow test is used to check information from the /etc/shadow file for a specific user. This file contains a user's password, but also their password aging and lockout information. It extends the standard TestType as defined in the oval-definitions-schema and one should refer to the TestType description for more information. The required object element references an inetd_object and the optional state element specifies the information to check. The evaluation of the test is guided by the check attribute that is inherited from the TestType.

Child Elements	Type	MinOccurs	MaxOccurs
object	oval-def:ObjectRefType	1	1
state	oval-def:StateRefType	0	1

< shadow_object >

The shadow_object element is used by a shadow test to define the shadow file to be evaluated. Each object extends the standard ObjectType as definied in the oval-definitions-schema and one should refer to the ObjectType description for more information. The common set element allows complex objects to be created using filters and set logic. Again, please refer to the description of the set element in the oval-definitions-schema.

A shdow object consists of a single user entity that identifies the username associted with the shadow file.

Child Elements	Type	MinOccurs	MaxOccurs
username	oval-def:EntityObjectStringType	1	1

< shadow_state >

The shadows_state element defines the different information associated with the system shadow file. Please refer to the individual elements in the schema for more details about what each represents.

Child Elements	Type	MinOccurs	MaxOccurs
username	oval-def:EntityStateStringType	0	1
password	oval-def:EntityStateStringType	0	1
chg_lst	oval-def:EntityStateStringType	0	1
chg_allow	oval-def:EntityStateStringType	0	1
chg_req	oval-def:EntityStateStringType	0	1
exp_warn	oval-def:EntityStateStringType	0	1
exp_inact	oval-def:EntityStateStringType	0	1
exp_date	oval-def:EntityStateStringType	0	1
flag	oval-def:EntityStateStringType	0	1

< uname_test >

The uname test reveals information about the hardware the machine is running on. This information is the parsed equivalent of uname -a. For example: "Linux quark 2.6.5-7.108-default #1 Wed Aug 25 13:34:40 UTC 2004 i686 i686 i386 GNU/Linux" or "Darwin TestHost 7.7.0 Darwin Kernel Version 7.7.0: Sun Nov 7 16:06:51 PST 2004; root:xnu/xnu-517.9.5.obj~1/RELEASE_PPC Power Macintosh powerpc". It extends the standard TestType as defined in the oval-definitions-schema and one should refer to the TestType description for more information. The required object element references a uname_object and the optional state element specifies the metadata to check. The evaluation of the test is guided by the check attribute that is inherited from the TestType.

Child Elements	Type	MinOccurs	MaxOccurs
object	oval-def:ObjectRefType	1	1
state	oval-def:StateRefType	0	1

< uname_object >

The uname_object element is used by an uname test to define those objects to evaluated based on a specified state. There is actually only one object relating to uname and this is the system as a whole. Therefore, there are no child entities defined. Any OVAL Test written to check uname will reference the same uname_object which is basically an empty object element.

< uname_state >

The uname_state element defines the information about the hardware the machine is running one. Please refer to the individual elements in the schema for more details about what each represents.

Child Elements	**Type**	**MinOccurs**	**MaxOccurs**
machine_class	oval-def:EntityStateStringType	0	1
node_name	oval-def:EntityStateStringType	0	1
os_name	oval-def:EntityStateStringType	0	1
os_release	oval-def:EntityStateStringType	0	1
os_version	oval-def:EntityStateStringType	0	1
processor_type	oval-def:EntityStateStringType	0	1

== EntityStateEndpointType ==

The EntityStateEndpointType complex type restricts a string value to a specific set of values that describe endpoint types associated with an Internet service. The empty string is also allowed to support empty emlement associated with variable references.

Value	**Description**
stream	for a stream socket
dgram	for a datagram socket
raw	for a raw socket
seqpacket	for a sequenced packet socket
tli	for all TLI endpoints

== EntityStateWaitStatusType ==

The EntityStateWaitStatusType complex type restricts a string value to two values, either wait or nowait, that specify whether the server that is invoked by inetd will take over the listening socket associated with the service, and whether once launched, inetd will wait for that server to exit, if ever, before it resumes listening for new service requests. The empty string is also allowed to support empty emlement associated with variable references.

Value	**Description**
wait	The value of 'wait' specifies that the server that is invoked by inetd will take over the listening socket associated with the service, and once launched,

	inetd will wait for that server to exit, if ever, before it resumes listening for new service requests.
nowait	The value of 'nowait' specifies that the server that is invoked by inetd will not wait for any existing server to finish before taking over the listening socket associated with the service.