OVAL Specification ntuser\_test, ntuser\_object, ntuser\_state, ntuser\_item

# Overview

Currently there exists content with OVAL Registry tests that require checking information located in the HKEY\_CURRENT\_USER (HKCU) hive. However, unless the tool used to run these tests is written in such a way as to check all of the user accounts on the server or workstation only the user running the scanning program will be tested for compliance. OVAL does not currently include any requirements for determining which users should be evaluated with HKCU checks. This can lead to devices reporting compliance to OVAL tests when in fact they should have failed the tests. This is a well-known limitation in OVAL. To combat this problem, we are proposing a new OVAL NTUser Test that should be used in place of any Registry tests that require checking the HKCU hive. This proposal is consistent with the previous discussion of this issue at security automation developer days[[1]](#footnote-1). It is a standardization of a proprietary checking process that has been in operational use on a federal government network for several years. There are two possible approaches to deploying this test,

* **System Specific**: The NTUser test would use the key/name/value entries located in the ntuser.dat file of each user account. User accounts can be obtained by the registry profile list (HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList). This approach is valid for evaluating all user accounts on a single endpoint. This approach has been built into a special version of SCC for demonstration purposes and represents a standard form of a previously proprietary approach that has been in use for several years.
* **Central Location**: The NTUser test would be given the path to a central location of any number of users’ ntuser.dat files. Each file would be tested for compliance. This approach is designed to evaluate user accounts at the source rather than at each endpoint.

This paper defines the benefits and potential pitfalls of both solutions.

# Technical Discussion:

The premise of the ntuser\_test is to examine data from the ntuser.dat file and gather profile information for a user on a system. The user can be currently logged on locally, logged on from a domain, or logged off. The ntuser.dat file contains key/name/value entries for certain profile information and is the data source that populates the HKEY\_CURRENT\_USER (HKCU) hive in the Windows registry. When a user logs onto a system, there is a folder created under C:\Users\<username> (Windows 7, Server 2008, Server 2008 R2) or C:\Documents and Settings\<username> (Windows XP, Vista, Server 2003). When a user is currently logged onto the system, the ntuser.dat file is loaded into the Windows registry under the HKCU and HKEY\_USERS (HKU) hive (XP allows for a single user to be logged in at a time, while Windows 7 allows for up to two concurrent logins). An OVAL tool implementing this test must be able to determine which users are currently logged on and evaluate those users by inspecting HKU. Users not logged on will be evaluated by inspecting ntuser.dat files directly. Profile files should never be locked or otherwise interfere with a real user logging on.

## Profiles defined on the system

The Window’s registry contains a list of all profiles that have logged onto the system for both local and domain accounts. Each account is assigned a unique System ID (SID), which is used as the registry key containing the user’s profile information. Included in the information is the “ProfileImagePath” name/value pair that contains the path of the user’s ntuser.dat file. The profile list will only contain information for user accounts that have logged onto the system. Any accounts, either local or domain, that have never logged onto the system will not be listed. Thus the profile list will provide an accurate representation of the state of each user account as of the last time that particular user logged onto the system currently under evaluation.

Potentially inaccurate results could be reported if the system being scanned contains accounts that haven’t been logged into in a while since these accounts may contain outdated information that newer policy or computer configurations may have fixed. These accounts could be excluded by using the date\_modified element to be greater than or equal to some date set by the content author. Additionally, other inaccurate results may come from accounts that use roaming profiles. The user may have logged onto a different system since they last logged onto the system being tested resulting in the test failing due to older domain policies that may have been updated and included in the roaming profile from the user’s last logon to the other system. Scanning on multiple systems in a domain where users login to multiple systems will result in findings for a single user on multiple systems. For a given user, only the most recent login will potentially represent the true current compliance if policies are updated or changed.

The following sections describe the elements to remove or add from an existing (v5.10.1) OVAL Registry Test to convert it to an NTUser Test.

### Elements for removal:

**NTUSER\_OBJECT and NTUSER\_STATE:**

*-* **hive:** The hive element is no longer required as all of the data is resident in the .dat file (or HKCU/HKU hive).

### Elements for inclusion:

**NTUSER\_OBJECT:**

The object elements included in the ntuser\_object are almost identical to those specified in the registry\_object. The only change is in the behaviors element.

-**behaviors**: Optional behaviors similar to win-def:RegistryBehaviors will be required to properly support recursion.

-**key** (string): Required element from the original win-def:Registry Object. Defined identical to the same element in the registry\_object.

**-name** (string)**:** Required element from the original win-def:Registry Object. Defined identical to the same element in the registry\_object.

**-filter**: Optional element for reducing the list of objects. Defined identical to the same element in the registry\_object.

**NTUSER\_STATE/ITEM:**

Items marked as new are different from the existing registry\_state. Existing items are unchanged from the registry\_state.

-**key** (string): This element describes a registry key normally found in the HKCU hive to be tested. Optional.

**-name** (string):This element describes the name of a value of a registry key. If the xsi:nil attribute is set to true, then the name element should not be used in analysis. Optional.

***-*sid** (string) (new): SID of the ntuser.dat file owner. Optional. Defined in HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList.

***-*username** (string) (new): The username of the ntuser.dat file owner in the form of domain\user or machine\user. Optional.

***-*account\_type** (enum) (new): determines if user account is local or domain account (local or domain). Optional.

**-logged\_on** (bool) (new): determines if the user is currently logged on or not. Optional.

***-*enabled** (bool) (new): determines if user account is enabled or disabled. Optional.

***-*date\_modified** (int) (new): last modified date of the ntuser.dat file (this is set when the user last logs off, when the current user makes a setting that writes to the registry, or when the current user locks/unlocks the session). Optional.

***-*days\_since\_modified** (int) (new): The number of days since the ntuser.dat file was last modified. The value should be rounded up to the next whole integer as policy generally dictates a value less than or equal to a specific integer. Optional.

***-*filepath**(string) (new)**:**The filepath element specifies the absolute path for the ntuser.dat file on the machine. A directory cannot be specified as a filepath. Optional.

***-*last\_write\_time** (int) (new)**:**This entity is the last time that the key or any of its value entries were modified. The value of this entity represents the FILETIME structure which is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC). Last write time can be queried on a key or name. When collecting only information about a registry key the last write time will be the time the key or any of its entries was written to. When collecting only information about a registry name the last write time will be the time the name was written to. See the RegQueryInfoKey function lpftLastWriteTime. Optional.

***-*type*:*** The type entity allows a test to be written against the registry type associated with the specified registry key(s). Please refer to the documentation on the EntityStateRegistryTypeType for more information about the different valid individual types. Optional.

***-*value*:*** The value entity allows a test to be written against the value held within the specified registry key(s). If the value being tested is of type REG\_BINARY, then the datatype attribute should be set to 'binary' and the data represented by the value entity should follow the xsd:hexBinary form. (each binary octet is encoded as two hex digits) If the value being tested is of type REG\_DWORD or REG\_QWORD, then the datatype attribute should be set to 'int' and the value entity should represent the data as an integer. If the value being tested is of type REG\_EXPAND\_SZ, then the datatype attribute should be set to 'string' and the pre-expanded string should be represented by the value entity. If the value being tested is of type REG\_MULTI\_SZ, then only a single string (one of the multiple strings) should be tested using the value entity with the datatype attribute set to 'string'. In order to test multiple values, multiple OVAL registry tests should be used. If the specified registry key is of type REG\_SZ, then the datatype should be 'string' and the value entity should be a copy of the string. Note that if the intent is to test a version number held in the registry (as a reg\_sz) then instead of setting the datatype to 'string', the datatype can be set to 'version'. This allows tools performing the evaluation to know how to perform less than and greater than operations correctly. Optional.

## Profiles Stored in a Central Repository

In many large networks, roaming profiles are utilized so that users can log into any number of systems on the network without having to setup a local profile on each system. A centralized repository is used to store each user’s account information including the ntuser.dat file. Each time the user logs onto a system, a local copy is stored on the system. Additionally, during the logoff process the local copy is written back to the central repository. Therefore, the most recent copy of the user’s profile is always located in the central repository. This means, however, that if multiple systems were to be scanned and a user had not logged into one of the systems within a certain amount of time the local copy of the user’s account could be outdated and provide a false result for the test being run. Thus, scanning the central repository instead of individual systems for determining if a test is passed or failed may be a more efficient approach. It should be noted that this approach may result in very large results when run on networks with many users. The system approach will have even more results but they will be distributed over many results files on the endpoints.

Possible pitfalls include insufficient rights to read different user’s ntuser.dat files as well as the inability to correctly capture the different elements defined in the schema such as username, SID, and logged\_on. Microsoft has established best practices for roaming profile locations, where by only the user and local system account have read or write access to each profile[[2]](#footnote-2). Therefore, without changing the permissions on the directory and files, a successful scan could only be completed using the system account. Some tools today already run using system account permissions but this may be a problem for other tools. Additionally, a centralized repository may not provide a clear indication of the owner of each ntuser.dat file because this information is not stored in the file. The user account that owns each file can often be determined from the folder in which the file resides but this information may not be reliable. However, a more reliable approach is to query Active Directory (AD) as AD maintains the profile file location for each user. This process is currently external to the OVAL definition. The existing definition would return results for only the file and it would be left to the administrator to determine how to best remediate each user. This approach would also not evaluate local user accounts that may be present on each endpoint.

The primary concern with using the central repository approach is determining applicability of the content being tested. For example, a user may never log onto a system that has Microsoft Office installed and therefore may not have or need the settings required by the Office SCAP/OVAL content. Thus if this user’s ntuser.dat file is scanned in the central repository the user would be reported as non-compliant, and may never be compliant resulting in a potential false positive that must be accounted for with each scan report. Add to this another user that will or has logged onto a system with Microsoft Office installed and should have the appropriate settings configured but does not. The second user will also fail and be reported as non-compliant as expected, but the final report will include both users and will be unable to differentiate between the two users.

Support for the central scanning approach would require changes to the ntuser\_object and ntuser\_state. The object would need the path element included so an author can specify the directory where the target ntuser.dat files will be found. The behaviors element would be used to tell the tool to search the subdirectories through recursion to find additional files. Additionally, elements included in the ntuser\_state (sid, username, account\_type, logged\_on, and enabled) would never be collected as these elements cannot be accurately discerned from a raw ntuser.dat file. The changes required to support the central repository use case, lead to defining a different OVAL Test, Object, State, and Item: NTUserCentral.

The following sections describe the elements to remove or add from an existing (v5.10.1) OVAL Registry Test to convert it to an NTUserCentral Test.

### Elements for removal:

**NTUSERCENTRAL\_OBJECT and NTUSERCENTRAL \_STATE:**

*-* **hive:** The hive element is no longer required as all of the data is resident in the .dat file (or HKCU/HKU hive).

### Elements for inclusion:

**NTUSERCENTRAL \_OBJECT:**

The object elements included in the ntusercentral\_object are almost identical to those specified in the registry\_object with the addition of file behaviors and a path element.

-**behaviors**: Optional behaviors similar to win-def:RegistryBehaviors and win-def:FileBehaviors will be required to properly support recursion in both the registry and file structure.

-**key** (string): Required element from the original win-def:Registry Object. Defined identical to the same element in the registry\_object.

**-name** (string)**:** Required element from the original win-def:Registry Object. Defined identical to the same element in the registry\_object.

**-path** (string)**:** Required element. The path element specifies the directory component of the absolute path to a file on the machine.

**-filter**: Optional element for reducing the list of objects.

**NTUSERCENTRAL\_STATE/ITEM:**

Items marked as new are different from the existing registry\_state. Existing items are unchanged from the registry\_state.

-**key** (string): This element describes a registry key normally found in the HKCU hive to be tested. Optional.

**-name** (string):This element describes the name of a value of a registry key. If the xsi:nil attribute is set to true, then the name element should not be used in analysis. Optional.

**-path** (string)**:** The path element specifies the directory component of the absolute path to a file on the machine. Optional.

***-*date\_modified** (int) (new): last modified date of the ntuser.dat file (this is set when the user last logs off, when the current user makes a setting that writes to the registry, or when the current user locks/unlocks the session). Optional.

***-*filepath**(string) (new)**:**The filepath element specifies the absolute path for the ntuser.dat file on the machine. A directory cannot be specified as a filepath. Optional.

***-*last\_write\_time** (int) (new)**:**This entity is the last time that the key or any of its value entries were modified. The value of this entity represents the FILETIME structure which is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC). Last write time can be queried on a key or name. When collecting only information about a registry key the last write time will be the time the key or any of its entries was written to. When collecting only information about a registry name the last write time will be the time the name was written to. See the RegQueryInfoKey function lpftLastWriteTime. Optional.

***-*type*:*** The type entity allows a test to be written against the registry type associated with the specified registry key(s). Please refer to the documentation on the EntityStateRegistryTypeType for more information about the different valid individual types. Optional.

***-*value*:*** The value entity allows a test to be written against the value held within the specified registry key(s). If the value being tested is of type REG\_BINARY, then the datatype attribute should be set to 'binary' and the data represented by the value entity should follow the xsd:hexBinary form. (each binary octet is encoded as two hex digits) If the value being tested is of type REG\_DWORD or REG\_QWORD, then the datatype attribute should be set to 'int' and the value entity should represent the data as an integer. If the value being tested is of type REG\_EXPAND\_SZ, then the datatype attribute should be set to 'string' and the pre-expanded string should be represented by the value entity. If the value being tested is of type REG\_MULTI\_SZ, then only a single string (one of the multiple strings) should be tested using the value entity with the datatype attribute set to 'string'. In order to test multiple values, multiple OVAL registry tests should be used. If the specified registry key is of type REG\_SZ, then the datatype should be 'string' and the value entity should be a copy of the string. Note that if the intent is to test a version number held in the registry (as a reg\_sz) then instead of setting the datatype to 'string', the datatype can be set to 'version'. This allows tools performing the evaluation to know how to perform less than and greater than operations correctly. Optional.

# Conclusion:

The SCC team developed two separate prototypes, one for profiles defined on a system (NTUSER) and the other for profiles defined in a central repository (NTUSERCENTRAL). Both prototypes demonstrate that the information defined in the schema can be acquired for the separate use cases. The NTUSER use case could replace registry tests in existing content and provide a more complete picture of system compliance. However, when applicability is applied to each use case; NTUSERCENTRAL falls short of being able to directly replace existing content or content that is intended to scan for a specific application that may or may not be installed on the server containing the central repository. Thus while technically possible, it is our opinion that the NTUSERCENTRAL use case should not be used as a replacement for OVAL registry tests that use the HKEY\_CURRENT\_USER hive.

# Appendix A: SCC Implementation - NTUser

1. Gather User Profile information (list of SIDs) from ‘HKEY\_LOCAL\_MACHINE/SOFTWARE/Microsoft/Windows NT/CurrentVersion/ProfileList/’
2. Determine which user or users are currently logged onto the system.
3. Gather information for each profile listed in 1).
   1. Ignore any BUILTIN or NT AUTHORITY users.
   2. Local users will have a username resembling COMPUTER NAME/USERNAME and any account information can be obtained using local system calls. **(account\_type, enabled, sid, username)**
   3. Domain users will have a username resembling DOMAIN NAME/USERNAME and any account information can be obtained using remote system calls on the Domain Controller. **(account\_type, enabled, sid, username)**
   4. Check to see if the profile is included in the currently logged on user list. **(logged\_on)**
   5. Determine the last modification date for the user profile. **(date\_modified)**
   6. Gather ntuser.dat file location from ‘HKEY\_LOCAL\_MACHINE/SOFTWARE/Microsoft/Windows NT/CurrentVersion/ProfileList/[SID]/ProfileImagePath’. **(filePath)**
   7. Return the final list of profiles to gather registry keys from.
4. Create OVAL items for key/name pairs defined in the OVAL Object. **(key, name, last\_write\_time, type, value)**
   1. For any user currently logged on, check the local registry HKEY\_USERS using the user’s SID
   2. For any user not currently logged on, parse the ntuser.dat file.
5. Return collected OVAL Items and compare against any defined OVAL States.

# Appendix B: SCC Implementation - NTUserCentral

1. Using the defined object’s path element, gather all ntuser.dat files using appropriate operations and file behaviors defined in the object. **(path)**
2. Using each ntuser.dat file found in 1), create OVAL items for key/name pairs defined in the OVAL Object. **(key, name, date\_modified, last\_write\_time, type, value)**
3. Return collected OVAL Items and compare against any defined OVAL States.

# Appendix C: INCLUDE\_DEFAULT Behavior

In discussions on the OVAL Developer’s Forum, a suggestion was made to include the ntuser.dat file that is used as the basis for all new local user accounts. To ensure this file could be included in the results, the ‘include\_default’ behavior was added to the schema. This behavior is set to ‘false’ by default so as to not include the default ntuser.dat file since this file is not updated based on security settings applied to the computer.

The file is located in the Default user’s home directory, which can be found by querying the following registry key:

‘HKEY\_LOCAL\_MACHINE/SOFTWARE/Microsoft/Windows NT/CurrentVersion/ProfileList/Default’ for Vista, 7, 2k8, and 2k8r2 (%SystemDrive%\Users\Default)

or combining the registry keys

‘HKEY\_LOCAL\_MACHINE/SOFTWARE/Microsoft/Windows NT/CurrentVersion/ProfileList/ProfilesDirectory’ (%SystemDrive%\Documents and Settings) and ‘HKEY\_LOCAL\_MACHINE/SOFTWARE/Microsoft/Windows NT/CurrentVersion/ProfileList/DefaultUserProfile’ (Default User) for XP and 2k3.

Please note that this Default user is different from the .Default which is an alias for the Local System account. For details, see <http://blogs.msdn.com/b/oldnewthing/archive/2007/03/02/1786493.aspx>.

1. *Automated Checking of Windows User Configuration Settings*, Wed, July 11,2012 http://makingsecuritymeasurable.mitre.org/participation/devdays.html#summer2012 [↑](#footnote-ref-1)
2. http://technet.microsoft.com/en-us/library/cc757013(v=ws.10).aspx [↑](#footnote-ref-2)