UiPath: Guidance for Credentialing Robots and Managing their Access

The UiPath platform leverages software agents to perform automated tasks. These software agents called robots come in two varieties, attended and unattended. The attended robots are installed on a user machine and work alongside or interactively with the user. The user must trigger the robot in some way to have the robot start an automation. This is very similar to a user running a macro or a pre-programmed script. Unattended robots do not need to run on an end user machine. They are typically installed on server or virtual desktop environments, though they can operate on nearly any Windows operating system. Unattended robots must have their own user credentials as they must login to Windows to perform automations. Although not limited to what a user is able to do, the core of Robotic Process Automation (RPA) is based on the ability to execute tasks similar to a user. Given this principle, a robot must have a user session. Both attended and unattended robot operations prompt various questions around identity, credentialing, and access management (ICAM).

Attended Robots

A common debate amongst customers new to RPA is whether attended robots should be given their own credentials or should attended robots hand off to the user for authentication. By default, an attended robot runs under the context of a Windows user, as such it inherits the authorization of the user. This can be very useful when automating against common desktop applications such as Microsoft Office or Adobe Reader as permissions to files are simply inherited. When automating against more complex or sensitive applications however, there is often a concern in being able to differentiate

between actions that are performed via the robot software compared to those performed by the actual human user. If the robot either hands off to the user to perform the login or takes advantage of single-sign-on from running under the user context, then the application will see all actions as coming from the users. If the robot leverages a separate set of credentials, then the application will be able to separate the actions but a different problem arises. Both options require some further examination.

Option 1, Inherit Credentials:

It is trivial to develop an automation that pauses and displays a notification to ask a human user to perform an action such as performing a login. The robot can perform some work, display a message box asking the user to take an action and then simply wait for some trigger such as the webpage that is displayed after a successful login to continue working. This methodology can be repeated as many times as necessary. The advantage is that no new credentials need to be created or managed. Also, because robot automations can be paused, there is no risk of the user leveraging a separate set of credentials to perform some nefarious act though the robot. The disadvantage is that there is no way for the end application to understand if the actions were performed by a robot or the human user. Audit logs from the UiPath platform that record robot actions would have to be correlated against the application logs to verify that.

Option 2, Issue Separate Credentials

The UiPath platform allows for the storage of credential assets in Orchestrator. Retrieval of stored credentials over TLS and subsequently entering those credentials is supported out of the box. This means that a robot could perform an automation entirely on its own, potentially in the background while a user works. The advantage here is that when the robot performs a login, it does so with a distinct set of credentials. Log correlation between logs generated by the UiPath platform and application logs is easier by using this methodology. However, there is a downside, since this set of credentials needs to be managed either by the user or programmatically to ensure regular password rotation. Depending on the exact automation details, it may be possible for the user to pause the robot and perform actions under this separate identity or even copy the credentials depending on how those are stored, retrieved and entered.

Recommendation

Option 1 is preferred. While real-time access auditing between the actions of attended robots and humans may be appealing, it introduces unneeded complexity by requiring additional credentials. Should an audit need to be performed, the UiPath platform is capable of logging every action performed by the robot down to individual mouse clicks with accurate timestamps. As well these logs can be sent to a customer's log collector of choice so long as it is nLog compatible.

Unattended Robots

Typically an organization only issues three types of accounts:

1. User Accounts – issued to human users. These can be privileged or not, but often the basic user account is tied to some HR process.
2. Service Accounts – issued to allow one application to connect to another application such as a web application connecting to a database.
3. Computer/Device Accounts – issued to computers or other devices to identify and possibly authenticate to the network and retrieve policy.

Given that an unattended robot will most often perform actions similar to that of a human user, but operates on its own similar to a service account, there is often some confusion as to how to credential an unattended robot. Computer or device accounts can generally be ruled out as they do not have access to applications or systems required for automation.

Option 1, Use a Service Account

It is very common to think of a service account as the first option for robots. In many organizational models, any account that is not assigned to a human is considered a service account. There is usually a model or mechanism to request and provision such an account, so a new process does not need to be created. To security teams, service accounts often have a very negative connotation though as they often require some exceptions to normal account policies such as separate OU structures or relaxed password rotation requirements. Because of their nature, they can often be forgotten about if not well documented and regularly audited too.

Option 2, Use a Regular User Account

Often the robots in RPA are referred to as a digital workforce. Given that, it makes some sense to provision accounts to them like you would for a human user. Creating an account and assigning it similar roles and permissions as that of a human will allow a robot to perform work in the same systems as a human. There is an inherent problem though as most organizations have their identity provisioning and possibly even identity management tied to their HR processes. This is fine for a human who needs benefits, pay roll, and other such services, but it adds unneeded complexity for robots.

Option 3, Use a User Account but treat it like a Contractor

This could be referred to as option 2B as it is similar to the option above. Robots need similar access to that of humans without the pay or benefits. Many organizations have policies to address the account creation for contractors or provisional workers. These accounts typically do not include the same provisions of pay, benefits, and other HR related processes as they are handled by a third party. It is common for contractor accounts to be reviewed on a regular basis to ensure that their account and their various application permissions are still required. Additionally, it is common practice to separate out contractor accounts from others to provide a logical organization. These policies and procedures used to better manage contractor accounts typically provide a good starting point for handling robot accounts.

Recommendation

Option 3 is preferred, but not always available. It is easy to over think the creation of a robot account. Most of the time a robot needs to access the same applications as a human in a similar role. If there are already policies in place that allow for the creation of an account without the normal benefits, HR, and pay then that is preferred regardless of what the account type is actually called.

Organization and Maintenance

Beyond the account type created, there are a few best practices to follow that will help with managing robot accounts.

1. Name your robots with an indicator. Some organizations put a "R-" or a "B-" or "BOT_" in front of their robot accounts, this also makes searching for robot accounts much easier in the directory and identity management system.
2. Group your robots by job function. While many organizations only start off with one business unit leveraging RPA, usage typically expands out to several. Consider organizing your robots according to teams like you would group regular employees whether that be by putting them in a security groups or an OU. Again, giving the groups or OUs a prefix for easier identification and searching is recommended here too.
3. Give robots names. This may seem straight forward as well, but it is useful when auditing, planning, and organizing for the future. Some organizations prefer technical names such as BOT_<business_unit>_<team_name>_<ID_Number>. Other organizations will use the names of popular cartoon or comic book characters. Regardless of which structure is chosen, naming robots in a way that positively identifies individual robots and provides a grouping is useful in the long term.
4. Keep your password complexity. In fact, make your robots passwords longer and more complex. Robots do not have to worry about trying to remember long complex passwords. The maximum limit for most Windows systems is 127 characters. For individual application logins, use the maximum complexity for that application.
5. Rotate robot passwords frequently. Similar to password complexity, if a robot changes its password every week, day, or hour even, it still has no trouble remembering. This is not hard to do either. Password rotation can be done in multiple ways. Automations can be developed to include it at the end, PowerShell scripts are available from UiPath to do scheduled rotation for Orchestrator and AD, a management robot or automation can be used to change the passwords of all the robots periodically, or a credential vault such as CyberArk can be used. Starting with 2019.10, UiPath Orchestrator supports a pluggable credential management architecture, making it possible to connect to many different credential management systems.
6. Robots can leverage single-sign-on. If a human user account can leverage SSO, so too can the robot. Leverage SSO and existing access management systems where it makes sense. In cases where single-sign-on is not available, leverage credential assets within Orchestrator. Starting with 2019.10, credential assets can be stored in 3rd party credential vaults as well such as

CyberArk. They can also be set into folders that allow for fine grain access controls to be restricted from developers when access is no longer required.

7. Group the machine accounts where robots run as well. Whether you are leveraging VDIs, servers, or other machines to host unattended robots, it is a good idea to put them in a separate Security Group or an OU. This allows organizations to apply some security policy such as restricting robot accounts to only login to those machines or restricting access to those machines to only robots and support staff.

Conclusions

When a new disruptive technology such as RPA enters a marketplace, it can seem difficult to incorporate existing IT paradigms. Thankfully, with UiPath technology leveraging systems in a way that human users already do, few exceptions need to be made for how work is done already. Organizations can adopt RPA technology quickly without risking security and see quick return on investment.