elevated_groups

What it does

elevated_groups is a external fact. The fact will be used to give an AD Security Group elevated privileges on a Virtual Machine for both Windows and Linux.

Setup

Facter can parse structured data files stored in the external facts directory and set facts based on their contents.

Structured data files must be valid JSON, have the correct file extension, and must be located in the correct directory.

On Unix, Linux, or Mac OS X: `/etc/puppetlabs/facter/facts.d/`

On Windows: `C:\ProgramData\PuppetLabs\facter\facts.d\`

> Note: These directories don't necessarily exist by default; you may need to create them. If you create the directory, make sure to restrict access so that only administrators can write to the directory.

On Windows

A file named `elevated_groups.json` will need to be created in the external facts directory: `C:\ProgramData\PuppetLabs\facter\facts.d\`.

The content of the file needs to be valid JSON, and follows this pattern.

```
{
    "elevated_groups": {
        "Valid Local Windows Group": [
            "domain\\sg name",
            "domain\\sg name2"
        ],
        "Valid Local Windows Group 2": [
            "domain\\sg name",
            "domain\\sg name2"
        ]
    }
}
```

Each property of the elevated_groups object must be a valid local windows group.

The property can be any number of windows groups provided the group exists, and is spelled exactly how it is shown in Windows.

Example of a valid JSON file:

```
{
    "elevated_groups": {
        "Administrators": [
            "us\\sg-us test",
            "us\\sg-us test2"
        ],
        "Remote Desktop Users": [
            "us\\sg-us test3",
            "us\\sg-us test4"
        ],
        "Power Users": [
            "us\\sg-us test3",
            "us\\sg-us test4"
        ],
        "Custom Made Group": [
            "us\\sg-us test3",
            "us\\sg-us test4"
        ],
    }
}
```

> **IMPORTANT**: Inserting invalid security groups will cause Puppet to **fail** leading the server to not be fully configured. To resolve the failure update the `elevated_groups.json` file with the correct values.
>
> If elevated_groups.json is invalid JSON the file will be skipped, and any Security Groups specified will not be added. e.g. SG-US Example_Group)

On Unix, Linux, or Mac OS X

A file named `elevated_groups.json` needs to be created in the external facts directory: `/etc/puppetlabs/facter/facts.d/`.

NOTE: Create the directory If it does not already exist with the command, mkdir -/etc/puppetlabs/facter/facts.d.

- The elevated_groups object consists of two arrays `sudo_groups` and `access_groups`.
- `sudo_groups`: Manages the sudo access to a server. Groups or userid that needs sudo access to the virtual machine will be added here.
- `access_groups`: Manages login.group.allowed. Groups that need remote access to the server will be added here.

Apply these rules when creating the JSON file on *nix servers.

- All entries are lower case.
- Convert spaces in the original name to an underscore.

Using the rules above, the original name **SG-Example Group** is converted to **sg-example_group**.

- Do not use an AD service group which contains an underscore(_) in its original name as a Linux access group. It will not be recognized and users in the group will be unable to login to the server. For example, the original Active Directory group name **SG-US Example Group** is valid; however, **SG-US Example_Group** is invalid because is contains an underline.

- **For sudo_groups:** Either userid's or groups can be added to the "sudo_groups" list. To differentiate between the two, add a percent sign "%" to each group entry. (e.g. %sg-example_group)

- **For access_groups:** Only groups may be added to the "access_groups" list. Unlike the sudo_group entries, do not add a "%" sign to the beginning of each name.

Example of a valid JSON file:

```
{
    "elevated_groups": {
        "sudo_groups": [
            "%sg-us_example_group1",
            "adminowner"
        ],
        "access_groups": [
            "sg-us_example_group3",
            "sg-us_example_group4"
        ]
    }
}
```

IMPORTANT: Inserting invalid security entries will cause Puppet to add invalid entries to sudo and login.group.allowed. To fix the issue update the `elevated_groups.json` file with the correct entries, manually remove the invalid access_groups entries from /etc/login.group.allowed, and the files in /etc/sudoers.d that are named the same as the invalid sudo_groups entries. (Note, filenames for sudo groups will have a percent (%) sign at the beginning of the file name.)

If elevated_groups.json is invalid JSON, an error displays during the Puppet run, the file will be skipped, and any entries specified will not be added.