Custom Configuration Scripts

# Notes before script-specific annotation

For these scripts to be run properly and enact the necessary device changes, the majority (if not all) require root privileges. For that reason, the scripts ought to be run with maximum privileges from the highest directory within a sandboxed environment. This will ensure the scripts can alter settings without restrictions, find all the necessary files (as files are easily hidden within nested folders and directories) and do not pose a serious security threat to the organization utilizing them.

# Network Configuration

It is possible to create a unique Wi-Fi profile with specific wired settings and subsequently deploy this profile to devices within your organization. This script relies on editing the NetworkManager configuration file that exists within each device and can overwrite user configuration options via internal operations.

The command *nmcli* is a command-line tool that can be used to control NetworkManager and report network status. As such, it is the primary tool to implement a custom Wi-Fi network profile on all member devices. The properties listed below are frequently altered during custom configuration and/or are included in the networking script library.

Connection Settings (General Connection Profile)

* connection.autoconnnect : whether or not the connection should be automatically connected by NetworkManager when the resources for the connection are available. “True” to automatically activate the connection, “False” to require manual intervention to activate the connection. Autoconnect happens when the circumstances are suitable.
  + If multiple profiles are ready to autoconnect on the same device, the one with the better "connection.autoconnect-priority" is chosen.
* connection.autoconnect-priority : a range of priority integers from -999 to 999 with a default priority of zero. If the connection is set to autoconnect, connections with higher priority will be preferred.
  + This property only matters if there is more than one candidate profile available for autoconnect.
  + If priorities are equal, the profile that has been used most recently will take precedence.
* connection.metered : whether or not a given connection is metered, meaning that all data usage is tracked and the network connection has a limited data usage per month or per user.
  + Property changes on currently activated connections take effect immediately.
* connection.auth : a range of integers between 1-3600 that represents the number of seconds between authentication attempts after a failed authentication attempt. The default timeout is 1 second.
* connection.auth-retries : the number of retries for user authentication for a set of credentials.
  + A retry-value of zero means a user can try indefinitely, -1 means to use a global default. If the global default is not set, the authentication retries 3 times before failing the connection.
  + This applies only to 802-1x authentication.
* hidden : a connection option that needs to be set to yes when connecting for the first time to an AP not broadcasting its SSID. Otherwise, the SSID would not be found and the connection attempt would fail.

802-1x Authentication Settings

* 802-1x.eap : this indicates the allowed EAP method that is used when authenticating to the network with 802-1x.
  + Valid methods include "leap", "md5", "tls", "peap", "ttls", "pwd", and "fast". Each method requires different configuration using the properties of this setting - refer to wpa\_supplicant documentation for the allowed combinations.
* 802-1x.identity : the identity string for EAP authentication methods, which is often the user’s username or login name.
* 802-1x.phase2-auth : a string that specifies the allowed “phase2” inner authentication method when an EAP method that uses an inner TLS tunnel is specified in the “eap” property.
  + For TTLS: this property selects one of the supported non-EAP inner methods which include “pap”, “chap”, “mschap”, and "mschapv2" while "phase2-autheap" selects an EAP inner method.
  + For PEAP: this property selects an inner EAP method, which include "gtc", "otp", "md5" and "tls". Each "phase 2" inner method requires specific parameters for successful authentication; see the wpa\_supplicant documentation for more details.
  + Both "phase2-auth" and "phase2-autheap" cannot be specified.
* 802-1x.auth-timeout : an 32-bit integer timeout for authentication with a global default of zero. If the global default is not set, the authentication timeout is 25 seconds.

802-11-Wireless-Security Settings

* 802-11-wireless-security.keym-mgmt : the key management used for the given connection that must be set for any Wi-Fi connection that uses security.
  + The options for key management are "none" (WEP or no password protection), "ieee8021x" (Dynamic WEP), "owe" (Opportunistic Wireless Encryption), "wpa-psk" (WPA2 + WPA3 personal), "sae" (WPA3 personal only), "wpa-eap" (WPA2 + WPA3 enterprise) or "wpa-eap-suite-b-192" (WPA3 enterprise only).

Proxy Settings

* proxy.method : the 32-bit integer method for proxy configuration with a default of NM\_SETTING\_PROXY\_METHOD\_NONE (0)

# Microsoft Defender for Endpoint Configuration

Microsoft Defender for Endpoint (MDE) is an endpoint security platform designed to help enterprise networks prevent, detect, investigate, and respond to advanced threats. With this, MDE configuration allows a client to configure and deploy a unique MDE security protocol to all devices within his or her organization.

A selection of the most commonly configured settings – most of which can be found in the script library – are listed below. Each relies on the *mdatp* command, which is installed through the MDE deployment process.

* Real-time protection: config real-time-protection --value
  + This MDE setting can be enabled or disabled.
* Behavior monitoring: config behavior-monitoring --value
  + This MDE setting can be enabled or disabled.
* Cloud protection: config cloud
  + This MDE setting can be enabled or disabled.
* Product diagnostics: config cloud-diagnostic --value
  + This MDE setting can be enabled or disabled.
* Automatic sample submission: config cloud-automatic-sample-submission --value
  + This MDE setting can be enabled or disabled.
* AV passive mode: config passive-mode --value
  + This MDE setting can be enabled or disabled.
* Antivirus exclusion for a directory: exclusion file [add/remove] --path
* List all antivirus exclusions: exclusion list
* Manage threats: threat allowed [add/remove] --name
* PUA protection: threat policy set --type --action
  + Possible types include potentially\_unwanted\_application or archive\_bomb.
  + Possible actions include block, off, and audit.
* Degree of parallelism for on-demand scans: config maximum-on-demand-scan-threads --value
  + The value can be any numerical value between 1 and 64.
* Scans after security intelligence updates: config sacn-after-definition-update --value
  + This MDE setting can be enabled or disabled.
* Archive Scanning (on-demand scans only): config scan-archives --value
  + This MDE setting can be enabled or disabled.
* Log level: log level set --level
  + Possible levels include error, warning, info, and verbose.
* Generate diagnostic logs: diagnostic create --path
* Health check: mdatp health
* Path scan: scan custom --path
  + This scan can ignore given exclusions.
* Scans: mdatp scan [full|quick|cancel]
* Security intelligence update: definitions update
* Remove files all quarantined files: threat quarantine remove-all
* Manage files detected as a threat to the quarantine: threat quarantine [add/remove --id
  + The --id value ought to be the threat ID for the file.
* Restore a quarantined file: threat quarantine restore --id --path
* Set early preview: edr early-preview
  + This MDE setting can be enabled or disabled.
* Set group-id: edr group-ids --group-id
* Manage tags: edr tag [set/remove] --name GROUP --value
  + The input --value is the tag.
* List exclusions from root: edr exclusion list [processes|paths|extensions|all]

# Firewall configuration

Firewall configuration relies on a command line utility called *Iptables*. Armed with this command, it is possible to alter the firewall rules that a device refers to whenever a connection is attempted.

It is important to note that the order of rules within iptables is important and are typically added at the beginning of the rules list unless otherwise specified with -I. The most common rules involve three values (ACCEPT, REJECT, and BLOCK) and determine how a device handles different kinds of connections. The primary difference between BLOCK and REJECT is that REJECT sends an error response back to the sender before disallowing the connection whereas BLOCK does not.

Another important characteristic of firewall configuration is port designation. Port assignments tend to be the same across devices, with certain ports used only for specific kinds of connections or functions. More information on the port database and the uses of each port can found at: <https://www.speedguide.net/ports.php>.

* Loopback Connections: each device uses a loopback interface to send itself connections (essentially, devices can ping themselves). The server may also use it if your application server is configured to connect to the “localhost” address.
* Established and Related Incoming/Outgoing Connections: allowing, blocking or rejecting incoming and/or outgoing connections from established and related servers.
* Internal to External: the primary purpose of configuring this setting is to give internal access to the external within a device. The default external network is eth0 and the default internal network is eth1.
* Dropping Invalid Packets: it is possible for packets to be labeled as invalid. As such, it is also possible to drop any packets that are considered invalid.
* SSH: SSH becomes relevant if the device is using a cloud server. In this case, the firewall would need to allow incoming SSH connection to port 22 in order to be able to connect to and manage the server.
* Web Server: HTTP connections are typically received on port 90 while HTTPS typically come on port 443. Web servers utilize these ports to “listen in” and catch connection requests. In order for this to be possible, the firewall needs to set rules that allow the network traffic through.
* MySQL: when using a database server, it is possible that the client may utilize a remote server in order to access the given database. This client subsequently needs the firewall to allw incoming traffic from the relevant port, which is typically port 3306.
* PostgreSQL: the reasons behind enabling PostgreSQL traffic are the same as for MySQL, but PostgreSQL uses a different port. Connections for PostgreSQL typically come from port 5432.
* Mail: depending on the device protocol, mail servers can receive and send connections on a number of different ports. As such, it is important to identify the protocol in place and allow, block, or reject connections according to that protocol.
  + SMTP uses port 25.
  + IMAP uses port 143.
  + IMAPS uses port 993.
  + POP3 uses port 110.
  + POP3S uses port 996.
* Other: there are other related firewall settings that can be altered. For example, limit bursts or traffic limitations can be important in preventing a DoS attack or protecting the server from being overwhelmed with connections. These settings are configured according to the policy, protocols, and security standards an organization wishes to maintain and as such, are unique to the organization and devices in question.

# Patch update configuration

Patch updates typically involve small pieces of software that are used for error correction within computer software, resolution of security vulnerabilities, or small feature updates that improve a program’s functionality or defense against attack. For this reason, they can be incredibly important and might require regular management.

This script serves three purposes related to patch updates: 1) it updates, upgrades, and otherwise runs patch updates for the software and repositories it identifies on a given device; 2) it alters automatic update settings to enable daily automatic updates and periodic software cleanup/removal; and 3) it acts as a resource for downloading new or updated repositories.

* /etc/apt/apt.conf.d/50unattended-upgrades: this file contains an important piece of code that is, by default, commented out. This line enables automatic updates through the unattended-upgrade package and needs to be un-commented in order to allow automatic update settings to be altered or enabled.
* /etc/apt/apt.conf.d/20auto-upgrades: used in conjunction with the file above, this file contains the configuration for periodic update packages. As such, it is possible to set settings to a value of “1” to tell the system to check for updates and upgrade necessary repositories on a daily basis (and to clean out unnecessary software every 30 days). For this reason, the script overwrites the file in order to ensure that daily automatic updates are enabled.
* Any repository URL can be added to the REPO\_URL variable and then downloaded onto the device using the script. This repository will be added to the device’s repository list, which can be found via “/etc/apt/sources.list”
* Similarly, a particular patch update file can be merged with existing software via the PATCH\_FILES variable.

# Certificate deployment

This script serves as an easy method by which certificates can be copied from agent packages and installed. Certificates allow a device or an identity to be identified via a digitally signed document that can be sent back and forth between a device and the agent/organization that requested it.

The script itself relies on the administrator to send out the certificate files that are to be deployed and for those certificate files to already be downloaded on the device. From there, the script will copy the desired certificate files into the device’s certificate directory, update the list of certificates (adding if needed), and ensure certificate permissions are not locked. At that time, the certificate is then deployed on the device.